В.А.Дронов

JAVASCRIPT В WEB-ДИЗАЙНЕ

Книга посвящена языку сценариев JavaScript и его использованию в Web-дизайне. Дан вводный курс по языку HTML. Рассматривается работа JavaScript-сценариев в составе Web-страниц: общие принципы написания сценариев; управление элементами страниц на основе объектной модели документа (DOM); организация взаимодействия с пользователем на основе событий; ввод-вывод при помощи Web-форм, диалоговых HTML-окон, Буфера обмена и технологии drag-n-drop; работа с базами данных; применение мультимедийных эффектов. Изложение сопровождается большим количеством примеров HTML-страниц с использованием JavaScript-сценариев. В приложениях к книге содержатся справочные сведения о HTML, CSS и DOM.

Содержание

Теги

Методы

События

Команды ExecCommand

Объекты

Коллекции

Предметный указатель

Введение

Для кого предназначена эта книга

Эта книга для тех, кто интересуется современными интернет-технологиями, кто собирается заняться или уже занимается разработкой Web-страниц, кто хочет создать в Интернете что-то свое, новое, а не просто уныло созерцать чужое, сделанное другими. В общем, это книга для создателей, творцов, если хотите.

Конечно, она не охватывает всех аспектов Web-строительства. Чтобы описать все возможности, предоставляемые одной только *WWW* (World Wide Web — Всемирной паутиной), нужно много десятков толстых книг. Я и не ставил себе цели объять необъятное. Расскажу только о *языке создания сценариев JavaScript* и его использовании в Web-дизайне. Поверьте, JavaScript — это целая вселенная, которая вполне может стать вашей.

Что такое JavaScript и зачем он нужен

Вначале — немного отвлеченных материй.

Если вы — бывалый "интернетчик", то должны были заметить, что Всемирная Сеть становится все более и более динамичной. На смену унылым статичным страничкам пришли "живые", реагирующие на действия пользователя и изменяющие в реальном времени свое содержимое (причем вам даже не нужно перезагружать их, чтобы увидеть все изменения). В Интернете появились звук и видео. Да и сами странички стали намного более приятными для глаза; теперь они зачастую больше походят на документы, сверстанные в настольных издательских программах. (Конечно, не все странички стали такими: кое-кто до сих пор никак не вылезет за рамки текста и пары корявых рисунков, но это уже зависит от фантазии и/или предпочтений Web-дизайнера.) Почему так случилось? Первое и главное — в Сеть

пришли рядовые граждане планеты Земля, которые хотели не только работать в ней, но и развлекаться с ее помощью. А во-вторых, появилось несколько новых технологий, позволивших привнести в Сеть чуточку жизни. Появились новые программы. Наконец, стали доступны довольно "широкие" каналы связи, по которым возросшие потоки информации могли проходить достаточно свободно. Одним словом, новые потребности вызвали к жизни новые предложения от поставщиков программного обеспечения для Интернета.

Все Web-страницы пишутся на языке *HTML* (HyperText Markup Language — язык гипертекстовой разметки). Это довольно странный язык программирования, описывающий не то, что должна сделать программа, дабы получить определенный результат, а сам результат, получаемый после выполнения программы. А что за действия необходимы для получения этого результата, остается на совести программы. То есть HTML описывает представление Web-страницы на экране, которое программа *Web-обозревателя* должна предоставить пользователю. В большинстве случаев, это представление статично; в самом деле, что еще нужно этим ученым занудам (см. историю HTML ниже). Чтобы добавить в Web-страницу динамические элементы, нам необходимо описать, как сделать их динамичными. То есть, нужен еще один язык программирования.

Одним из таких языков и стал JavaScript. Его-то мы и будем изучать.

Небольшие программы — *скрипты, или сценарии* — пишутся на JavaScript и особым образом внедряются в HTML-код Web-страниц. Эти программы получают доступ к отдельным элементам страницы с помощью так называемой *объектной модели документа* (DOM — Document Object Model) — набора специальных программных *интерфейсов*, иначе говоря, "рычагов" воздействия на HTML-документ.

Для тех, кто не знает HTML, в начале книги будет приведен небольшой вводный курс. Основное время будет посвящено исключительно использованию JavaScript и DOM. Мы узнаем, как создавать динамические Web-страницы, реагирующие на действия пользователя изменением своего содержимого. Более того, JavaScript предоставляет также некоторые возможности анимации: вообразите себе этакие мультики на вашей странице! Ну и в самом конце я расскажу о доступе к базам данных, о возможности, мало кем используемой.

Уже интересно? Минутку терпения.

Инструменты

Теперь о том, какое программное обеспечение нам понадобится для работы. Прежде всего, это, конечно, программа для просмотра Web-страниц, Web-обозреватель (или Web-браузер, как часто говорят и пишут). JavaScript поддерживают Web-обозреватели Microsoft Internet Explorer 3.0 и Netscape Navi-

gator 3.0, а также их более новые версии. Следует знать, что реализации DOM в этих двух программах сильно отличаются; Microsoft трудилась над своим Web-обозревателем в поте лица, в то время как Netscape явно отдыхала. И если программист, создающий JavaScript-программы для Internet Explorer, чувствует себя свободно и волен делать очень многое, то приверженец Netscape ограничен почти во всем. Подробнее об этом — далее в книге, но уже сейчас уясните, что при желании создавать Web-страницы, одинаково хорошо работающие на обоих Web-обозревателях, вам придется установить оба браузера на свой компьютер и тестировать каждую страницу на каждом из них отдельно.

Кроме Web-браузеров, для работы понадобятся программы редактирования HTML. Сейчас на рынке хватает HTML-редакторов разных типов: от сложнейших, мощнейших пакетов типа Dreamweaver или FrontPage до небольших бесплатных программ, написанных левой задней ногой в перерыве между пивом и Quake. Какой выбрать? Рекомендую, стандартный Блокнот, поставляемый в составе Windows (или аналогичный текстовый редактор, если вы пользуетесь другой системой). И знаете, почему? Дело в том, что именно программы, подобные Блокноту, позволят вам "почувствовать" HTML, посмотреть, как устроена та или иная страница. Конечно, "тяжелые" *WYSIWYG*-пакеты (What You See Is What You Get — что вы видите, вы и получаете, стандартная аббревиатура для обозначения программ, показывающих документ таким, какой он есть) ускорят создание Web-страницы и избавят от необходимости задумываться о каком-то там HTML, но ведь наша цель не в этом. К тому же, сейчас пока никто не создал достаточно мощный пакет для написания JavaScript-программ и внедрения их в HTML-код (во всяком случае, я таких пакетов еще не видел). Так что вам лучше пользоваться Блокнотом или аналогичным текстовым редактором; для Windows я лично рекомендую Bred 2 (http://www.gladiators.nm.ru), т. к. он поддерживает все русские текстовые кодировки и позволяет редактировать файлы размером более 64 Кбайт.

Вот, вроде, и все. Теперь можно и начинать. Но сначала — небольшой экскурс по Интернету, WWW и HTML. Заранее прошу прощения за то, что буду говорить вещи, которые знают все бывалые "интернетчики". Просто эту книгу могут читать самые разные люди, в том числе те, кто не знаком с основополагающими концепциями Интернета. Поэтому с ними нужно провести небольшой ликбез. Вы позволите, господа крутые Web-мастера?

Краткая история Интернета

Каменный век. ARPAnet. Становление Интернета

Впервые нечто подобное современной сети Интернет было создано в самом начале семидесятых годов в Соединенных Штатах по инициативе тамошнего Министерства Обороны. Это "нечто" называлось *ARPAnet* и было, прежде

всего, экспериментом по созданию устойчивых компьютерных коммуникаций, функционирующих даже в случае глобальной ядерной войны. Сеть была спроектирована таким образом, что обеспечивалась ее стабильная работа даже при выходе из строя значительного числа составляющих ее рабочих станций. И, надо сказать, этот давний эксперимент удался.

Для того чтобы компьютеры, объединенные в сеть, могли нормально общаться друг с другом, они должны "говорить" на одном и том же "языке", иными словами, пользоваться одинаковым стандартом передачи данных, не зависимым ни от архитектуры компьютера, ни от операционной системы, ни от географического местоположения. Такой стандарт называется *протоколом*. Сеть ARPAnet базировалась на протоколе *IP* (Internet Protocol — интернет-протокол). Он был специально создан для общения компьютеров в "ненадежной" сети.

Надо заметить, что, хотя проект в целом и поддерживался Министерством Обороны США, но реально все делалось энтузиастами буквально "на коленке". Даже *Международная организация по стандартизации* (International Organization for Standardization — ISO) далеко не сразу стандартизировала протокол IP. Зато сама сеть росла как на дрожжах. Все больше и больше энтузиастов устанавливало на свои разношерстные компьютеры программное обеспечение для поддержки IP и включалось в растущее сообщество себе подобных.

Именно сеть ARPAnet стала предшественником Интернета.

Какие же *сервисы* предлагал тогдашний прото-Интернет? Немного. Всего-то FTP (File Transfer Protocol — протокол передачи файлов) и электронная почта. Где-то в восьмидесятых появились Usenet (всемирное сообщество групп новостей) и Gopher (своего рода предшественник WWW, нечто вроде систематизированного архива документов). До грядущего интернет-бума оставалось еще ох как долго...

В начале восьмидесятых появились локальные сети (*Ethernet*, *TokenRing* и пр.). Крупные организации, владеющие такими сетями, также подключались к ARPAnet. В самом деле, в современном мире главное — это устойчивая и быстрая связь, а какая связь может быть быстрее электрической. Однако, поскольку сеть ARPAnet работала не на самом лучшем и быстром оборудовании (например, часто использовались телефонные линии даже в качестве опорных магистралей для передачи больших потоков данных — представьте себе!), очень скоро возникла потребность технического перевооружения. Министерство Обороны тем временем создало свою собственную сеть и полностью открестилось от ARPAnet, так что помощи от вояк было ждать бесполезно. А число ARPAnet-чиков все росло и росло...

На помощь пришел Национальный научный фонд США (NSF — National Science Foundation). К тому времени под его началом числилось пять суперкомпьютерных центров, которым была жизненно необходима быстрая сеть

для обмена информацией и общения с пользователями. Она и была создана. Организации, непосредственно связанные с суперкомпьютерными центрами, подключали к себе других желающих, те — третьих, и так до бесконечности. Подобные организации, предоставляющие услуги по доступу в Интернет, стали называться интернет-провайдерами (или поставщиками услуг Интернета). В конце концов новая сеть перестала справляться с возросшими потребностями интернетчиков.

Я сказал "интернетчиков", и это не ошибка. Получившаяся сеть уже не имела ничего общего с проектом ARPAnet. Это была сеть Интернет. Ибо от старой ARPAnet там не осталось ровно ничего.

Чтобы вместить резко возросший *трафик* (поток данных), была проложена новая сеть, заменены устаревшие серверы. Это продолжается и по сей день: что-то обновляется, что-то наращивается, что-то делается заново.

Да, Великая Сеть и сейчас растет и совершенствуется. Почему же тогда, подключившись к интернет-провайдеру и набрав адрес в строке Web-обозревателя, вы не получаете сообщения: "Закрыто на ремонт" или что-нибудь еще более глупое? А все потому, что умные головы еще в далеких семидесятых сделали Интернет сетью помехо- и сбоеустойчивой. И если где-то что-то чинят, интернетчики не испытывают никаких неудобств.

Железный век. HTML и WWW

Я уже говорил, что первыми пользователями Интернет были ученые. Именно они и их нужды подхлестнули на первых порах развитие Всемирной сети. Так случилось и с самым популярным на сегодняшний момент сервисом — WWW (World Wide Web — Всемирная паутина).

Все началось с того, что ученым, работавшим в *Европейской лаборатории элементарных частиц* (CERN), расположенной в Швейцарии, понадобилось обмениваться между собой различными электронными документами таким способом, чтобы на любом компьютере они выглядели одинаково. Естественно, в качестве среды существования этих документов предполагалось использовать Интернет: CERN была одним из оживленнейших мест в Сети. Этой проблемой занялся сотрудник Лаборатории по имени Тим Бернерс-Ли.

Надо сказать, что американский ученый Теодор Хольм Нельсон занимался подобной проблемой еще в 1960 году. Он поставил перед собой крайне амбициозную цель: объединить в особой компьютерной сети все более или менее значимые текстовые документы, созданные к тому времени человечеством, и логично связать их между собой. При этом читатель мог из любого места одного документа перейти к другим, содержащим дополнительную или поясняющую информацию. В 1965 году Нельсон назвал такой метод организации текстовой информации *гипертекстом*. А свой так и не осуществленный проект он именовал *Xanadu*.

Именно идеи, заложенные Нельсоном в Xanadu, и дали жизнь WWW.

Бернерс-Ли не стал брать на себя такие глобальные обязательства: он просто хотел помочь коллегам. Свое творение — язык создания платформенно-независимых, связанных друг с другом текстовых документов — он назвал HTML. Связывались эти документы друг с другом посредством *гиперссылок* — специально выделенных слов, активизировав которые читатель попадал в связанный документ, где мог получить дополнительные сведения по описанной в документе проблеме. (Теоретически, конечно — на практике гиперссылка может завести куда угодно...) Для просмотра гипертекстовых документов Бернерс-Ли написал программу, получившую название Web-обозревателя. Происходило все это в 1989 году.

Ученому люду затея Бернерса-Ли понравилась настолько, что вскоре в CERN-овском сегменте Интернета выросла целая паутина из гипертекстовых документов, связанных друг с другом самым причудливым образом. Название нового интернет-сервиса вертелось, вероятно, не на одном языке — Всемирная паутина, World Wide Web. Она все росла и росла и вскоре вырвалась за пределы Лаборатории, раскинулась по всей Европе, перебралась за океан, опутывая своими тенетами множество новых поклонников. Это был настоящий WWW-взрыв.

Этот взрыв продолжается и до сих пор. Дошло до того, что сейчас многие считают, что Интернет и WWW — это одно и то же.

В 1993 году молодой и удалой американский студент Марк Андриессен напишет Web-обозреватель Mosaic. Эта программа первой в своем классе получит графический интерфейс и станет работать с мышью. Именно она будет индустриальным стандартом; все Web-обозреватели, даже самые новейшие, супернавороченные, до сих пор используют ее наработки. Андриессен станет мировой знаменитостью. Несколько позже он организует компанию Netscape, и она станет одной из самых успешных компьютерных компаний за всю историю этого сектора рынка.

Странное совпадение: как раз в 1993 году окончательно, после долгой агонии умрет проект Нельсона Xanadu...

WWW и технологии, на которых она основана, в настоящее время регулируются специальной некоммерческой организацией под названием WWW Consortium (W^3C). Все предложения от разработчиков собираются в так называемые проекты стандартов, которые после долгого (иногда слишком долгого) обсуждения становятся или не становятся стандартами. Последняя версия HTML, принятая 24 декабря 1999 года, имеет номер 4.01. Конечно, она далеко ушла от той, что предложил в 1989 году Тим Бернерс-Ли.

В Web-дизайн давным-давно пришла графика. Предприимчивые интернетчики создали фреймы — оригинальный способ разделения Web-страницы на несколько независимых частей. А сколько новых возможностей форматирования текста, размещения элементов появилось — не счесть! Мы подробно рассмотрим их в книге, в основном, в главе 1, посвященной HTML.

И, конечно, развивалось программное обеспечение. В мир интернет-программ пришла великая и ужасная Microsoft со своим Internet Explorer и задвинула Netscape в глубокую тень. И она же дала Web-дизайну новый толчок.

Но об этом — в главе 1.

Новейшая история. Java, JavaScript, Dynamic HTML. Современное положение дел

Со временем в Интернете произошло то, что должно было произойти при таких темпах роста: в Сеть пришел рядовой гражданин. Он не был искушен в науках, он не собирался искать труды по ядерной физике, общаться со всякого рода оригиналами и чудаками, наводнившими к тому времени Интернет, не жаждал запретных удовольствий, которых, чего греха таить, там тоже хватало. Он желал читать новости, следить за биржевыми котировками, искать новые товары в магазинах, смотреть фильмы и слушать музыку. Это был добропорядочный обыватель со всеми своими достоинствами и недостатками. И его следовало опекать, ведь именно обыватель является опорой нашей экономики.

Обывателю, прежде всего, следовало сделать красиво. Да и самим обитателям Сети к тому времени до смерти надоели унылые статичные странички.

Но вся проблема заключалась в том, что HTML не предполагает никакой динамики в Web-страницах. Он плосок и уныл, как предвыборная листовка, брошенная в ваш почтовый ящик. Это просто текст, а много ли в тексте жизни. И какими бы уловками не пользовались писатели и дизайнеры, они не в силах преодолеть эту плоскость и это уныние голого текста. (Кстати, Web-дизайнеры крайне ограничены в выразительных средствах, не то что дизайнеры, работающие в полиграфии.) Требовалось что-то еще...

Первый шаг сделала фирма *Sun* в 1995 году. Он назывался Java.

Java — это язык разработки кроссплатформенных приложений. Поясняю: *кроссплатформенные*, значит, не зависимые от платформы. Приложения пишутся на Java, компилируются в некий промежуточный код, называемый *байт-кодом* Java (он не имеет ничего общего с командами процессора), и исполняются на компьютерах клиентов с помощью *виртуальной машины* Java, понимающей байт-код. Java-приложения очень компактны и безопасны для пользователя, т. к. вследствие особой организации виртуальной машины они не могут натворить бед на клиентском компьютере, даже если очень захотят. Небольшие Java-приложения можно встраивать прямо в Web-страницы; при их просмотре Web-обозреватель автоматически загрузит само приложение, запустит виртуальную машину, а остальное — дело техники, т. е. самого приложения. Такие компактные приложения назвали *аплетами*.

Теперь предположим, что вы хотите написать Java-апплет, показывающий на Web-странице какой-то график. Вы изучили язык Java (а если вы знаете C++, то это совсем несложно), написали апплет, откомпилировали, а он не работает. В код вкралась ошибка! Вы переписываете код, опять компилируете, потом еще раз переписываете и компилируете... не слишком ли много шагов? Не лучше ли просто вставить Java-код прямо в HTML-страницу?!

Такая возможность появилась через год. Netscape на основе Java был создан язык JavaScript, программы на котором (скрипты) могли встраиваться прямо в HTML-код. (Говорят, что JavaScript — *скриптовая* версия Java.) В отличие от своего предка, JavaScript был языком интерпретируемым, а не компилируемым, т. е. виртуальная машина Java выполняла не полученный в результате компиляции байт-код, а сам JavaScript-код. Таким образом была исключена потребность в компиляторе и ускорен процесс разработки программ.

Но разработчики Web-обозревателей пошли еще дальше. Примерно в то же время компаниями Microsoft и Netscape был предложен ряд добавлений в HTML, получивший название *Dynamic HTML*. Когда-то эта аббревиатура не сходила со страниц книг и периодических изданий и немудрено — эти две компании предложили нечто такое, что подняло на дыбы всю WWW. Фактически, с помощью Dynamic HTML можно было программировать Web-страницы, как программируют обычные приложения с оконным интерфейсом.

Dynamic HTML — это набор расширений HTML, специальных методик и соответствующих программных интерфейсов (к ним относится уже знакомая вам DOM), реализованных в новейших Web-обозревателях (на то время: Microsoft Internet Explorer 3.0 и Netscape Navigator 3.0), которые позволяют взаимодействовать JavaScript-программе и HTML-коду страницы. То есть, программа на JavaScript, внедренная в тело Web-страницы, может управлять всеми элементами этой страницы: двигать заголовки, менять цвет текста, прятать одни элементы и показывать другие и даже управлять поведением самого Web-обозревателя. Web-дизайнер, использующий средства Dynamic HTML, становится настоящим программистом. Более того, программисты могут создавать приложения, исполняемые на сервере и имеющие в качестве интерфейса Web-страницу. Это может произвести (и производит потихоньку) переворот в программировании!

Нет, Тим Бернерс-Ли такого не видал!

Еще раз повторю, что Dynamic HTML — это нестандартное расширение HTML, предложенное разработчиками двух самых популярных программ Web-обозревателей. И поэтому реализации Dynamic HTML в этих двух Web-обозревателях сильно различаются между собой. Когда же консорциум W3C наконец-то стандартизировал эти расширения и включил их в состав HTML 4.0, то аббревиатурой Dynamic HTML стали обозначать саму техно-

логию создания "живых" Web-страниц. W3C стандартизировал и объектную модель документов DOM; в настоящее время доступна и поддерживается версия 1.0, стандартизируется 2.0 и пока еще набросана начерно 3.0.

Из программ Web-обозревателей сейчас наиболее популярен Microsoft Internet Explorer 5.5, полностью удовлетворяющий спецификации HTML 4.0 и DOM 1.0. Последняя версия Netscape Navigator (4.76) выглядит в этом плане значительно бледнее. Правда, совсем недавно вышла версия 6.0; посмотрим, что это за штука. А там на подходе и Internet Explorer 6.0.

JScript и VBScript

Фирма Microsoft отличается привычкой (хорошей или плохой — не знаю) переделывать все писаные и неписаные стандарты под себя. Так она переделала под себя JavaScript и назвала его JScript. Эта "частная" версия всеми признанного языка отличается наличием нескольких дополнительных функций.

С VBScript история сложнее. Издавна Microsoft поддерживала языки программирования семейства Basic. Последним представителем этого славного семейства является Visual Basic 6.0, довольно мощная среда разработки Windows- и интернет-приложений, широко распространенная и отлично поддерживаемая как самим производителем, так и независимыми разработчиками. Естественно, Microsoft просто не могла не перенести свое любимое детище в мир Web-дизайна. Скриптовая версия Visual Basic получила название *VBScript*. Этот язык несколько проще в освоении, чем JavaScript (т. к. Visual Basic был проще Java), хотя и ограниченнее по возможностям. Главнейшим недостатком VBScript является то, что он работает только на Internet Explorer. Так что, если вы хотите создавать динамические страницы, одинаково хорошо работающие и на Internet Explorer, и на Navigator, вам придется забыть о VBScript.

VBScript нашел широкое применение и в областях, не связанных с Интернетом, рассмотрение которых выходит за рамки этой книги.

Как работает WWW

Продолжаем ликбез для новичков. Поговорим о том, как же работает это чудо XX века — Всемирная паутина. Рассмотрим ситуацию с неким сайтом, посвященным... какая разница, чему!

Прежде всего, все Web-страницы, организованные с помощью гиперссылок в единый сайт, находятся на жестком диске компьютера, подключенного к Сети по быстрому каналу. Этот компьютер может и не выглядеть так же, как ваш настольный ПК, может работать под управлением другой операционной системы, но, тем не менее, они близкие родственники. На этом ком-

пьютере исполняется программа, получающая запросы от пользователей Интернета, отыскивающая на диске нужные файлы — документов, картинок, мультимедийных данных, архивов, программ — и посылающая их пользователям. Web-страницы, картинки и некоторые мультимедийные файлы воспроизводятся непосредственно Web-обозревателем, для других загружаются дополнительные приложения, или же Web-обозреватель предлагает их сохранить на диске для дальнейшего использования. Так что красивые страницы, усеянные графикой, взрывающиеся звуком и анимацией — это всего лишь файлы, что хранятся где-то вдали на жестком диске.

Программа, принимающая запросы от пользователей и выдающая им файлы, называется *Web-сервером*. Этим термином также именуют компьютер, где она работает. Иногда, чтобы не было путаницы, дополнительно уточняют, что имеется в виду: компьютер или программа.

Пользователь запрашивает Web-страницы, набирая в поле адреса Web-обозревателя адрес страницы. Если вы не впервые в Интернете, то знаете наизусть уже не один такой адрес.

http://www.microsoft.com — адрес сайта Microsoft.

http://www.netscape.com — а это адрес Netscape.

http://www.aport.ru — моя любимая поисковая машина, сайт для поиска информации в Интернете.

http://www.music.ru — огромная база данных русских музыкальных исполнителей. Рекомендую!

http://windoms.sitek.net/~vladmur/ — а здесь лежат мои литературные опыты.

Когда вы набираете любой интернет-адрес в поле адреса, Web-обозреватель устанавливает соединение с нужным сервером, запрашивает у него необходимые файлы, получает и отображает их. Таков принцип функционирования WWW, правда, в очень приблизительном изложении. На самом деле, Web-сервер предоставляет много дополнительных возможностей, но мы их здесь перечислять не будем.

Вы сами можете создать Web-сервер на своем компьютере. Для этого даже не понадобится никакого дополнительного программного обеспечения — только Windows и Блокнот (если вы пользуетесь другой операционной системой, задействуйте иные текстовый редактор и Web-браузер). Откройте Блокнот и создайте в нем следующий текст (только не делайте ошибок):

```
<HTML>
<HEAD>
<TITLE>Пример Web-страницы</TITLE>
</HEAD>
<BODY>
<H1>Пример Web-страницы</H1>
```

```
<P>Это Web-страница, созданная только ради демонстрации того,
что стать Web-дизайнером может всякий. Для этого не нужна сложная в
настройке программа Web-сервера. В данном случае, ваша операционная
система с успехом заменяет ее.</P>
</BODY>
</HTML>
```

Сохраните этот документ под именем Sample.htm (только заключите имя в кавычки, чтобы Блокнот не добавил по простоте душевной расширение txt). Потом дважды щелкните на созданном файле мышью, чтобы открыть его в Web-обозревателе. Вы должны увидеть нечто, похожее на рис. 1.

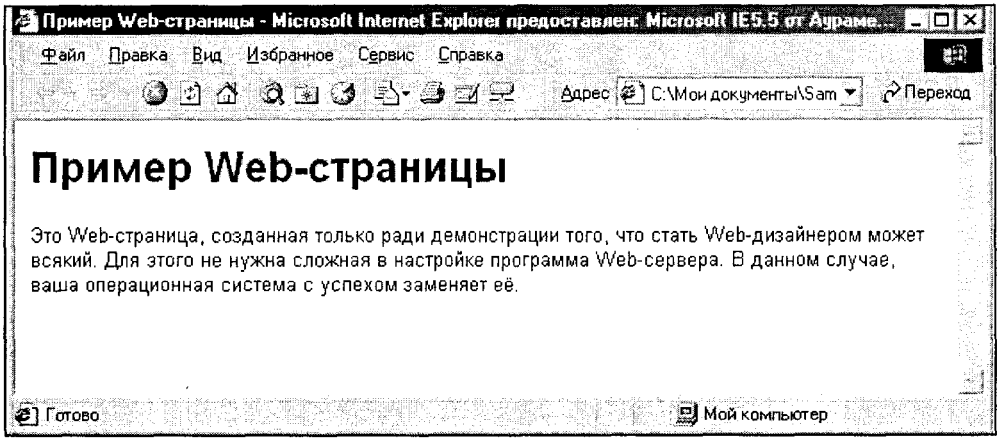


Рис. 1. Наш пример Web-страницы

В данном случае, операционная система, установленная на вашем компьютере, действительно работает как Web-сервер. Она принимает запрос от Web-обозревателя (он запрашивает файловую систему вашего ПК), отыскивает нужный файл и выдает его.

Вот и все! Теперь вы — настоящий Web-дизайнер. Конечно, пока вы умеете не очень много, но все когда-то начинают с малого.

Хотите знать больше?

Вы уже, наверно, жалуетесь на меня: "вот авторы пошли: ничего не объясняют толком". Читайте дальше. В главе 1 мы рассмотрим язык гипертекстовой разметки HTML. Вы научитесь создавать более сложные Web-страницы и вплотную подойдете к тому, чтобы приступить к изучению JavaScript и DOM.

Адрес: Россия, 199397, Санкт-Петербург, а/я 194; Web-сайт: www.bhv.ru

Уважаемые господа!

Издательство "БХВ-Петербург" приглашает специалистов в области компьютерных систем и информационных технологий для сотрудничества в качестве авторов книг по компьютерной тематике.

Если Вы знаете и умеете то, что не знают и не умеют другие,
Если Вам не нравится то, что уже написано,
Если у Вас много идей и творческих планов,

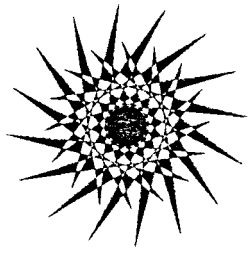
напишите книгу вместе с "БХВ-Петербург"

Ждем в нашем издательстве как опытных,
так и начинающих авторов,
и надеемся на плодотворную совместную работу.

С предложениями обращайтесь к
главному редактору Екатерине Кондуковой

тел.: (812) 251 4244, 251 6501
e-mail: kat@bhv.ru
факс: (812) 251 1295

ГЛАВА 1



Язык HTML

Сейчас мы займемся изучением языка HTML. Если вы уже его знаете, то можете пропустить главу и начать читать следующую. Но я уверен, что вы ее все-таки прочитаете, т. к. я постараюсь рассказать о вещах, которые известны мало кому из Web-дизайнеров. Но это потом, ближе к концу. А пока — вводный курс для новичков в Web-дизайне.

Сразу хочу предупредить, что я никоим образом не собираюсь рассматривать все возможности, все хитрости и тонкости HTML. Также не буду учить вас тонкостям и премудростям Web-дизайна, хотя все же дам некоторые понятия о создании Web-страниц. Для этого имеются другие книги. Более того, есть множество Web-сайтов, почтовых рассылок, групп новостей, где только и делают, что обсуждают HTML и тему "как сделать красиво в Интернете". Если вы желаете более глубоко изучить этот язык, поищите соответствующие книги в магазинах и материалы во Всемирной сети.

Ну а мы начнем, пожалуй...

Самое начало

Перед тем как, собственно, начать изучение языка HTML, условимся, что будем сохранять тексты примеров в файлах с расширением htm. Как правило, HTML-файлы имеют расширение htm или html; первое имя удовлетворяет соглашениям MS-DOS, но второе более логично. Вы можете использовать и первое, и второе расширение; просто я в данном случае выбрал первое.

Понятие тегов HTML

Вы уже, наверное, заметили, что HTML-документ представляет собой текст на нормальном человеческом языке, испещренный какими-то непонятными словами. Давайте еще раз посмотрим на текст пробной странички, написанной нами во введении.

```
<HTML>
<HEAD>
<TITLE>Пример Web-страницы</TITLE>
</HEAD>
<BODY>
<H1>Пример Web-страницы</H1>
<P>Это Web-страница, созданная только ради демонстрации того,
что стать Web-дизайнером может всякий. Для этого не нужна сложная
в настройке программа Web-сервера. В данном случае, ваша операционная
система с успехом заменяет ее.</P>
</BODY>
</HTML>
```

Большую часть текста странички занимает собственно текст, который мы видели в окне Web-обозревателя на рис. 1 во введении. Но что это за слова, обрамленные знаками ">" ("больше") и "<" ("меньше")? Зачем они нужны?

Эти слова называются *тегами*. Теги применяются для форматирования текста и вставки в текст различных нетекстовых элементов: графики, дополнительных объектов, Java-апплетов и пр. Одним словом, теги форматируют текст внутри HTML-документа.

Скажем, если вы хотите выделить какой-либо участок текста курсивом или оформить его в виде заголовка, то можете расположить его между соответствующими тегами, как показано ниже:

```
Этот текст будет выделен <I>курсивом</I>.
<H1>А это заголовок.</H1>
```

Теги `<H1>` и `</H1>` использовались в нашей пробной страничке — вы, вероятно, их уже узнали. Они обозначают *заголовок первого уровня*. Есть также теги, обозначающие заголовки второго, третьего, четвертого, пятого и шестого уровней.

В примерах мы помещаем текст между *парными* тегами. Всего парных тегов два: *открывающий* и *закрывающий*. В данном случае открывающими тегами являются `<I>` и `<H1>`, а закрывающими — `</I>` и `</H1>`. Обратная косая черта "/" — признак закрывающего тега.

В самом деле, это логично. Ведь для того, чтобы использовать в тексте какое-то форматирование, нужно указать, к какому именно фрагменту его применять. То есть, указать начало и конец текстового фрагмента, который будет отформатирован. Открывающий и закрывающий теги как раз и указывают такой фрагмент.

Парные теги могут быть *вложенными*. Рассмотрим пример.

```
<B>Это полужирный текст, <I>а этот еще и курсивный</I>.</B>
```

Тег `<B>` устанавливает полужирное начертание текста. Внутри этого тега вы видите еще один парный тег, устанавливающий курсивное начертание. Это

значит, что текст, находящийся в тегах <B> и <I> будет не только полужирным, но еще и курсивным. Говорят, что тег <I> вложен в <B>. Web-обозреватель прекрасно понимает, что к чему, и правильно отображает текст. Но если вы сделаете так:

```
<B>Это полужирный текст, <I>а этот еще и курсивный</B>.</I>
```

он может вас и не понять. Одним словом, каждому открывающему тегу должен соответствовать парный закрывающий. Путаница с тегами, похожая на ту, что мы только что видели, не допускается.

Теги различаются по *уровню вложенности*. В приведенном выше примере тег <B> имел первый уровень вложенности, а тег <I> — второй. (Наш пример очень прост, но в сложных HTML-документах уровень вложенности иных элементов может превышать 10.) При этом теги, находящиеся на предыдущем уровне вложенности по отношению к текущему, называются *родительскими*, а текущий тег является для них *дочерним*. В свою очередь, дочерний тег может быть родительским по отношению к другим тегам, вложенным в него. В нашем примере <I> — дочерний тег для <B>, а <B> является дочерним по отношению к тегу <BODY> (не указан), который является родительским и для <B>, и для <I>.

Бывают и *одиночные* теги. Например, тег, вставляющий в текст изображение — <IMG>. Ведь для того, чтобы вставить в необходимое место документа рисунок, не нужно указывать границы текстового фрагмента — достаточно задать только место, куда следует вставить рисунок. И все же, большинство тегов HTML — парные.

Вы спросите, а зачем нужны теги <P> и </P>? Они обозначают специальный текст, не относящийся ни к заголовкам, ни к листингам, ни к иным особым случаям. То есть, просто отдельный абзац текста, отделенный от других таких же абзацев пустым пространством.

Для примера давайте разобьем текст нашего единственного абзаца на два и посмотрим, что получится. См. листинг ниже (часть текста пропущена для экономии места). Сохраним этот пример в файле с именем 1.1.htm.

```
. . .
<BODY>
<H1>Пример Web-страницы</H1>
<P>Это Web-страница, созданная только ради демонстрации того,
что стать Web-дизайнером может всякий.</P>
<P>Для этого не нужна сложная в настройке программа Web-сервера.
В данном случае, ваша операционная система с успехом заменяет
ее.</P>
</BODY>
. . .
```

Web-обозреватель покажет нам картину, как на рис. 1.1.

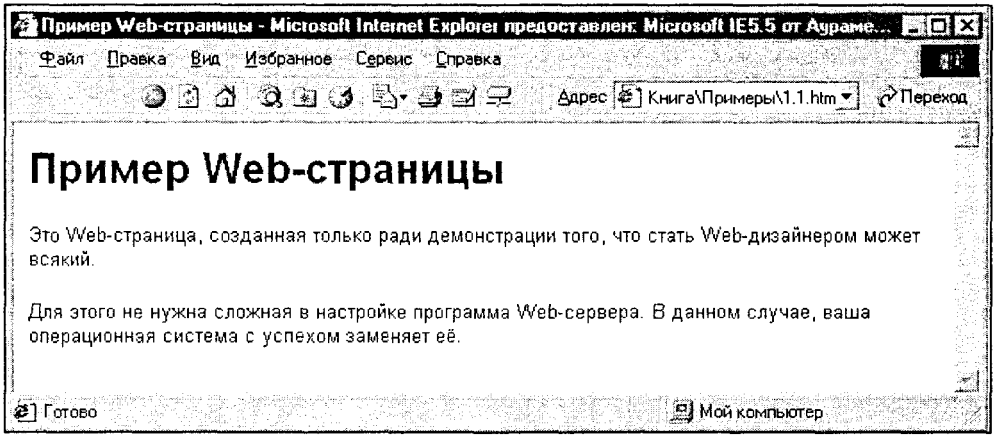


Рис. 1.1. Модифицированный пример Web-страницы

Вообще-то небольшой текст можно и не разбивать на абзацы, а просто печатать, как есть.

```
. . .
<BODY>
<H1>Пример Web-страницы</H1>
Это Web-страница, созданная только ради демонстрации того,
что стать Web-дизайнером может всякий. Для этого не нужна сложная
в настройке программа Web-сервера. В данном случае, ваша операционная
система с успехом заменяет ее.
</BODY>
. . .
```

В данном примере мы совсем исключили теги <P> и </P>. Web-обозреватель правильно отобразит эту страницу. Но для того, чтобы HTML-документ считался оформленным по всем правилам HTML 4.0, теги абзаца обязательно должны присутствовать, даже если HTML-документ очень мал. К тому же, эти теги позволят применить к тексту дополнительные возможности форматирования, предоставляемые HTML 4.0. Так что я советую вам все-таки использовать теги <P> и </P> в своих HTML-документах.

И еще немного о тегах. Дело в том, что спецификация HTML требует, чтобы Web-обозреватели, встретив любой неизвестный им тег, игнорировали его. Это открывает широкие возможности для Web-дизайнеров: от создания своих собственных "секретных" тегов (уж не знаю, зачем; возможно, это пригодится, если они когда-нибудь напишут свой Web-обозреватель) до скрытного размещения в теле HTML-документа каких-либо данных. Да и если Web-дизайнер случайно ошибется в написании какого-то тега, документ, по крайней мере, не будет содержать непонятного мусора.

Структура HTML-документа

Так, с тегами мы разобрались. Вы, наверное, желаете узнать сейчас как можно больше тегов, чтобы начать творить. Терпение!

Сначала поговорим о структуре типичного HTML-документа. Дело в том, что любой HTML-документ должен удовлетворять определенным правилам, чтобы его смог показать любой Web-обозреватель. Эта структура включает в себя три тега, которые мы здесь рассмотрим.

Но предварительно условимся о терминологии. HTML-документ, содержащий ошибки, но, все-таки, более или менее сносно показываемый Web-обозревателем, будем называть *хорошо оформленным*. (Не самый удачный термин, но общепринятый в индустрии...) Напротив, HTML-документ, не содержащий ошибок в оформлении, назовем *стандартным*.

Привередливый Navigator

Netscape Navigator имеет дурную славу привереды. Если в HTML-коде документа содержится малейшая ошибка или отклонение от стандартов (кроме тех, что поддерживаются самой фирмой Netscape), документ может быть искажен до неузнаваемости. Microsoft Internet Explorer более снисходителен к таким документам и в большинстве случаев отображает HTML-документ с ошибками нормально.

Взгляните еще раз на код нашей пробной странички. Вы увидите, что документ как бы разделен на две части двумя парами тегов и находится внутри третьей пары... Да что я тут говорю! Посмотрите сами! (Теги, о которых идет речь, выделены жирным шрифтом.)

```
<HTML>
  <HEAD>
    <TITLE>Пример Web-страницы</TITLE>
  </HEAD>
  <BODY>
    <H1>Пример Web-страницы</H1>
    <P>Это Web-страница, созданная только ради демонстрации того,
    что стать Web-дизайнером может всякий. Для этого не нужна сложная
    в настройке программа Web-сервера. В данном случае, ваша операционная
    система с успехом заменяет ее.</P>
  </BODY>
</HTML>
```

Вы видите, что весь HTML-документ заключен внутри парного тега <HTML> и </HTML>. Это первое правило оформления стандартных HTML-документов.

Второе правило заключается в том, что HTML-документ разделяется на две неравные *секции*.

Первая (и меньшая) секция — это *HTML-заголовок* (не путать с заголовками, обозначаемыми тегами <H1> и </H1>). HTML-заголовок выделяется парными

тегами <HEAD> и </HEAD>. Он не отображается в окне Web-обозревателя, а содержит служебную информацию, которую Web-обозреватель использует для своих нужд. В частности, в HTML-заголовке содержится *название* документа, выделенное парными тегами <TITLE> и </TITLE>. Это название появляется в заголовке окна и в списке "истории" Web-обозревателя. Еще раз повторяю, что нигде в самом документе эта информация не отображается.

Вторая (и большая) секция — это собственно документ, так называемое *тело* документа. Вот как раз оно-то и отображается в окне Web-обозревателя. Тело выделяется парными тегами <BODY> и </BODY>.

Отсюда следует второе правило оформления стандартных HTML-документов: каждый документ должен содержать секции HTML-заголовка и тела. И обе эти секции должны быть правильно оформлены.

Пролог HTML-документа

Вообще-то, для того чтобы HTML-документ считался стандартным, нужен еще и *пролог*. Пролог устанавливает, каким образом должен обрабатываться документ. Вы скажете, Web-обозреватель должен знать это без всяких прологов, и это будет правильно. Но буквоеды из W3C обязательно требуют пролог.

Как же выглядит это чудовище? Очень просто и совсем не страшно:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
```

Как видите, пролог представляет собой одиночный тег специального вида. Этот тег вставляется в самом начале HTML-документа перед открывающим тегом <HTML>. И обозначает он документ, оформленный в строгом соответствии со спецификацией HTML 4.0.

Вы можете вставлять в свои HTML-документы пролог или не вставлять его — на мой взгляд, это дело вкуса. В конце концов, подавляющее большинство HTML-документов во Всемирной сети не имеют пролога, и мир пока еще не перевернулся от такой дерзости по отношению к стандартам W3C.

Форматирование текста

После рассмотрения основ HTML пора переходить к описаниям конкретных тегов и предоставляемых ими возможностей.

Хочу сразу предупредить: я не буду описывать здесь все теги HTML. Опишу только те из них, что употребляются на Web-страницах наиболее часто. Полное описание тегов HTML, включая нестандартные и устаревшие, приведено в приложении 1 в конце книги.

Форматы символов

Сначала рассмотрим простое форматирование текста — возможности, аналогичные предлагаемым большинством текстовых процессоров.

Обычный текст достаточно поместить внутри парного тега <BODY> и </BODY>. Web-обозреватель просто покажет его в окне, как вы уже видели на рис. 1.1.

Если нужно отобразить отдельные символы другим цветом или в другом начертании, HTML предоставляет соответствующие теги. В табл. 1.1 рассмотрены главнейшие из них. Если вас интересуют все теги форматирования символов, см. приложение 1.

Таблица 1.1. *Теги форматирования символов*

Теги	Описание
<B>...</B>	Полужирный текст
<BIG>...</BIG>	Отображение текста шрифтом с увеличенным относительно стандартного размером
<BLINK>...</BLINK>	Мерцающий текст. Этот тег поддерживается только Netscape Navigator
<I>...</I>	Курсив
<SMALL>...</SMALL>	Отображение текста шрифтом с уменьшенным относительно стандартного размером
<STRIKE>...</STRIKE>	Зачеркнутый шрифт
_{...}	Верхний индекс
^{...}	Нижний индекс
<TT>...</TT>	Моноширинный шрифт. Моноширинными называют шрифты, у которых все символы имеют одинаковую ширину, например, "Courier New". Моноширинный шрифт создает иллюзию того, что текст набран на пишущей машинке или телетайпе
<U>...</U>	Подчеркнутый текст

Несколько примеров использования этих тегов:

```
<B>Полужирный текст</B>
<BIG>Великое</BIG> и <SMALL>малое</SMALL>
Химическая формула воды — H<SUP>2</SUP>O
```

Несколько особняком стоят теги <FONT> и </FONT>. Здесь мы столкнемся с таким понятием, как *атрибуты*.

Теги <FONT> и </FONT> позволяют настроить начертание, размер и цвет шрифта. Они имеют следующий вид:

```
<FONT [FACE="{Имя шрифта}"] [SIZE="{Размер шрифта}"]
{COLOR="#{Код цвета}"|"{Имя цвета}"]>
. . .
</FONT>
```

Вы видите, что этот тег имеет дополнительные параметры, так называемые атрибуты. Атрибутов тега <FONT> довольно много (см. приложение 1, где приводится полное описание HTML), но пока мы рассмотрим три главнейших. Эти атрибуты и задают нужные параметры шрифта. Заметьте, что они указываются только в открывающем теге, но не в закрывающем.

Атрибут FACE задает начертание шрифта, например, "Arial" или "Courier New". Имя начертания подставляется в кавычках после знака равенства. Можно задавать сразу несколько начертаний через запятую; в этом случае Web-обозреватель при отсутствии шрифта, указанного в шрифте первым, подставит один из последующих. Если же на компьютере клиента вообще нет шрифта, заданного в атрибуте FACE, Web-обозреватель использует шрифт по умолчанию.

Отсюда следует правило: не употребляйте в теге <FONT> редких и экзотических шрифтов, имеющихся только на вашем компьютере.

Атрибут SIZE задает размер шрифта. Но не в пунктах! Дело в том, что Web-обозреватель может отображать всего семь размеров любого шрифта; эти размеры пронумерованы от 1 до 7 в порядке увеличения. Размер по умолчанию имеет номер 3. Вы можете задать либо номер размера (без кавычек), либо относительный размер в виде +2 (т. е., шрифт с размером, большим на две позиции, чем размер по умолчанию) или -3 (соответственно, меньше на три позиции).

Атрибут COLOR позволяет расцветить ваш текст... нет, не всеми цветами радуги. Дело в том, что клиенты, которые будут просматривать ваши Web-страницы, могут использовать разную аппаратуру с совершенно различными параметрами. В том числе, у них могут быть разные видеокарты. И для того, чтобы Web-документ одинаково сносно просматривался на всех компьютерах, W3C сформулировал так называемую *безопасную таблицу цветов*. Эта таблица является частью спецификации HTML и содержит 216 цветов, рекомендованных к использованию Web-дизайнерами. Применяя безопасные цвета, вы можете быть уверенными, что они не сведут с ума ни одну видеокарту. Познакомиться с таблицей безопасных цветов можно в приложении 1.

Как задаются цвета? Либо RGB-кодом, либо *символическим именем*. RGB-код представляет собой шестнадцатеричное число, задающее долю в цвете соответственно красной, зеленой и голубой составляющей. Например, крас-

ный цвет будет иметь RGB-код #FF0000 (знак "решетки" "#" обязателен), белый — #FFFFFF, а черный — #000000. Те же цвета можно задать символическими именами: red, white и black. Значения цветов задаются в кавычках.

И еще. Ни один из этих атрибутов не является обязательным. То есть, хотя бы один из них должен присутствовать. Но указывать их все совершенно не надо — только те, которые действительно вам нужны.

Теперь пример:

```
<FONT SIZE=7>Большой <FONT COLOR="#00FF00">зеленый</FONT> текст</FONT>
```

Или так:

```
<FONT SIZE=7>Большой <FONT COLOR="green">зеленый</FONT> текст</FONT>
```

Здесь мы вложили одну пару тегов <FONT> и </FONT> в другую. В результате Web-обозреватель отобразит весь текст самым большим размером шрифта, а слово зеленый — еще и зеленым цветом.

Другие форматы символов

Все вышеприведенные теги указывают, как нужно сформатировать те или иные символы в вашем документе. Web-обозреватель просто отображает соответствующую часть текста полужирным шрифтом или зачеркивает его, не утруждая себя догадками, что этим хотел сказать Web-дизайнер. Но есть и другие теги форматирования текста; с их помощью Web-дизайнер может сказать Web-обозревателю, как следует понимать тот или иной фрагмент. Скажем, является ли текст цитатой или примером исходного текста программы.

Основная проблема заключается в том, что Web-дизайнер не может управлять отображением текста, выделенного такими тегами. Web-обозреватель может показать его, как ему заблагорассудится, или не отобразить вообще.

В табл. 1.2 приведены эти теги.

***Таблица 1.2.** Дополнительные теги форматирования символов*

Теги	Описание
<ACRONYM>...</ACRONYM>	Акроним (сокращение)
<CITE>...</CITE>	Цитата. Отображается курсивом
<CODE>...</CODE>	Пример исходного текста программы. Отображается моноширинным шрифтом
<DFN>...</DFN>	Определение термина. Отображается курсивом и поддерживается только Internet Explorer

Таблица 1.2 (окончание)

Теги	Описание
<EM>...</EM>	Выделенный текст. Отображается курсивом
<KBD>...</KBD>	Пример ввода пользователя. Отображается моноширинным шрифтом
<SAMP>...</SAMP>	Пример исходного кода программы. Отображается моноширинным шрифтом
<STRONG>...</STRONG>	Выделенный текст. Отображается полужирным шрифтом
<VAR>...</VAR>	Имя переменной, процедуры или функции в тексте Отображается курсивом

Пример использования:

```
<ACRONYM>WWW</ACRONYM> — самый распространенный сервис
<DFN>Интернета</DFN>
```

Зарезервированные символы HTML

Вы уже знаете, что теги HTML выделяются символами "<" ("меньше") и ">" ("больше"). Такие символы называют *зарезервированными*. Когда Web-обозреватель встречает в тексте эти символы, он предполагает, что между ними находится тег HTML, и соответственно пытается его распознать. Так что если вы вставите в текст одиночный символ "<" или ">", например, как часть математического выражения, Web-обозреватель будет сконфужен. Что он в этом случае покажет на экране, не знает никто, даже написавшие его программисты.

Что же делать, если вам в тексте нужно вставить один из таких зарезервированных символов? Используйте символические имена.

Наиболее общеупотребительные символические имена перечислены в табл. 1.3. Полный список символических имен находится в приложении 1. Заметьте, что всякое символическое имя начинается с амперсанда и заканчивается точкой с запятой.

Таблица 1.3. Символические имена

Зарезервированный символ	Символическое имя
Кавычка (")	"
Амперсанд (&)	&

Таблица 1.3 (окончание)

Зарезервированный символ	Символическое имя
Меньше (<)	`<`
Больше (>)	`>`
Неразрывный пробел (пробел, по которому текст не будет разрываться на строки)	` `
Права (©)	`&copy;`
Торговая марка (®)	`&reg;`

По традиции приведем несколько примеров:

```
&quot;Текст в кавычках&quot;
2&lt;3
Microsoft&reg;
```

Форматирование заголовков и абзацев

Мы рассмотрели форматирование отдельных символов и слов в HTML. Пришло время поработать по-крупному, рассмотреть, как форматируются целые блоки текста.

Некоторые парные теги, ответственные за блоки текста, мы уже знаем. Это теги форматирования разноуровневых заголовков `<H1>`—`<H6>` и соответствующие им закрывающие теги. И тег простого абзаца `<P>`.

Давайте подытожим наши знания.

Тег `<Hx>...</Hx>`, где `x` — число от 1 до 6, обозначающее уровень заголовка, преобразует заключенный в нем текст в заголовок соответствующего уровня. Web-обозреватель отображает заголовок крупным шрифтом, отличающимся от шрифта обычного текста.

Тег `<P>...</P>` разделяет текст на отдельные абзацы. Он может содержать атрибут `ALIGN="left|center|right"`, задающий способ выравнивания текста абзаца. Допустимо одно из трех значений: `left`, `center` и `right`, задающие соответственно левое, центральное или правое выравнивание.

```
<P ALIGN="center">Этот текст выровнен по центру</P>
```

В дополнение к вышеприведенным тегам HTML определяет набор тегов для специального форматирования текстовых абзацев. Они приведены в табл. 1.4.

Таблица 1.4. *Теги специального форматирования абзацев*

Теги	Описание
<ADDRESS>...</ADDRESS>	Адрес. Отображается курсивом, возможно, с отступом
<BLOCKQUOTE>...</BLOCKQUOTE>	Цитата. Отображается с отступом слева
<CENTER>...</CENTER>	Центрированный текст. Аналогичен тегу <P ALIGN="center">
<PRE>...</PRE>	Текст в заданном формате. Позволяет вывести текст на экране в таком виде, в каком он набран в HTML-коде. Отображается моноширинным шрифтом

Здесь стоит дать некоторые комментарии относительно тегов <PRE>...</PRE>. На первый взгляд непонятно, что он дает и зачем вообще нужен.

Сначала расскажем, как Web-обозреватель отображает на экране текст. Он руководствуется двумя простыми правилами:

- один или несколько подряд стоящих пробелов заменяются одним пробелом;
- возврат каретки преобразуется в пробел.

Это значит, что вы можете как угодно разбивать HTML-текст в своем документе, можете даже разместить каждое слово в отдельной строке или, наоборот, растянуть весь документ в одну длинную строку, и все равно Web-обозреватель разобьет его на строки правильно. Рассмотрим, например, такой листинг.

```
<HTML>
<HEAD>
<TITLE>Пример Web-страницы</TITLE>
</HEAD>
<BODY>
<H1>Пример
Web-страницы</H1>
<P>Это
Web-страница,
созданная
. . .
с
успехом
заменяет
ее.</P>
</BODY>
</HTML>
```

Узнали? Это наша многострадальная тестовая страница, только сейчас искаженная до неузнаваемости (часть текста пропущена). И все равно Web-обозреватель отобразит ее правильно (см. рис. 1 во введении).

Теперь заменим теги <P>...</P> на <PRE>...</PRE>. Новый файл примера будет иметь имя 1.2.htm.

```
<HTML>
<HEAD>
<TITLE>Пример Web-страницы</TITLE>
</HEAD>
<BODY>
<H1>Пример
Web-страницы</H1>
<PRE>Это
Web-страница,
созданная
. . .
с
успехом
заменяет
ее.</PRE>
</BODY>
</HTML>
```

Результат — на рис. 1.2. Видно, что в этом случае Web-обозреватель сохранил заданное нами разбиение на строки.

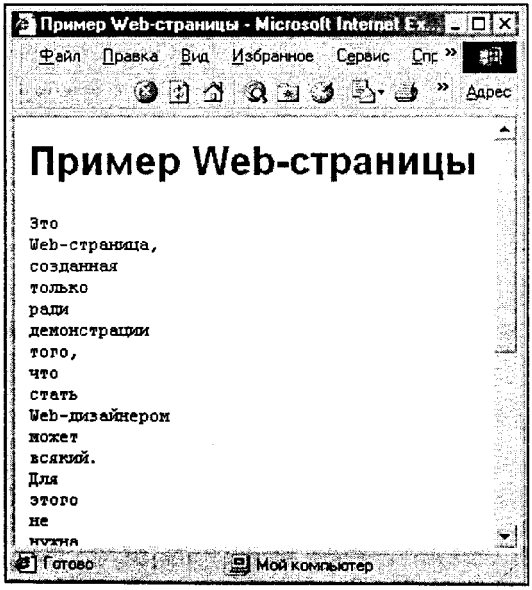


Рис. 1.2. Пробная страница, отформатированная с использованием тегов <PRE>...</PRE>

Эти теги можно использовать, например, для быстрого преобразования в формат HTML старых текстовых документов с разнообразным форматированием. Просто в начале текстового документа добавляется HTML-заголовок, теги <HTML> и <PRE>, меняется расширение файла и все!

Управление разрывами текста

Хорошо, что Web-обозреватель думает за нас, как ему разорвать строку, чтобы она полностью уместилась в его окне! Но иногда нам нужно, чтобы строка в некотором месте не разрывалась или, наоборот, разрывалась именно в этом месте. На такой случай HTML предлагает нам теги *управления разрывами.*

Все изученные нами до этого теги были парными. Сейчас мы ради разнообразия изучим пару единичных тегов.

Теги управления разрывами перечислены в табл. 1.5.

Таблица 1.5. *Теги управления разрывами*

Теги	Описание
 	Вставляет разрыв строки текста. Последующие символы будут отображаться с начала следующей строки
<HR>	Горизонтальная линейка
<NOBR>...</NOBR>	Текст без разрывов. Аналогичен тегам <PRE> и </PRE>, но не отображает текст моноширинным шрифтом. Если вы хотите, чтобы текст разрывался в каком-то месте, используйте тег

Здесь стоит подробнее рассказать о теге <HR>. Он имеет богатый набор атрибутов — целых пять.

```
<HR [SIZE="{Толщина}"] [WIDTH="{Ширина}"] [ALIGN="left|center|right"]
[NOSHADE] [COLOR="{Цвет}"]>
```

Атрибут SIZE задает толщину линейки.

Атрибут WIDTH задает ширину линейки. Можно задать как абсолютное значение ширины в пикселах, так и относительное в процентах. В последнем случае значение ширины имеет вид "{Ширина}%" и вычисляется относительно ширины окна Web-обозревателя.

Атрибут ALIGN задает выравнивание горизонтальной линейки. Допустимо одно из трех значений: left, center и right, задающие соответственно левое, центральное или правое выравнивание. Естественно, этот атрибут имеет значение только тогда, когда ширина линии меньше ширины окна Web-обозревателя, а иначе пользователь не заметит вашего выравнивания.

Атрибут NOSHADE стоит несколько особняком. Дело в том, что у него нет никакого значения — одним своим присутствием он указывает Web-обозревателю, что линейка не должна иметь тени. Все современные Web-обозреватели по умолчанию отображают горизонтальные линейки в трехмерном виде, с тенью. Атрибут NOSHADE отменяет это поведение.

Атрибут COLOR задает цвет линейки. Здесь все понятно. Но учтите, что этот атрибут поддерживается только Internet Explorer.

Под конец — пара примеров:

```
<P>Эта первая строка параграфа.<BR>А это вторая строка параграфа.</P>
<P>Ниже находится горизонтальная линейка синего цвета, занимающая 40%
ширины окна Web-обозревателя, выровненная по правому краю и
лишенная тени.<P>
<HR COLOR="#0000FF" WIDTH="40%" ALIGN="right" NOSHADE>
```

Списки

В заключение рассмотрим форматирование *списков* в HTML. Список — это набор упорядоченных абзацев текста, выделенных отступами, помеченных специальными значками *(маркированных)* или *нумерованных.* Вот пример маркированного списка:

- один;
- два;
- три.

А это — нумерованный список:

1. Один.
2. Два.
3. Три.

HTML предоставляет возможность создания как маркированных списков, так и нумерованных. Оба этих вида списков создаются практически одинаково с использованием почти одних и тех же тегов.

Маркированный список целиком помещают внутри пары тегов <UL> и </UL>, нумерованный — <OL> и </OL>. Каждая строка списка предваряется тегом <LI>.

```
<UL>
<LI>один;
<LI>два;
<LI>три.
</UL>
```

Тег <LI> очень странный. Очевидно, что ему нужен закрывающий парный тег </LI>. Но спецификация HTML почему-то считает его необязательным.

Тег <UL> может иметь атрибуты, управляющие отображением списка:

```
<UL [COMPACT] [TYPE="disc|circle|square"]>. . .</UL>
```

Атрибут COMPACT заставляет Web-обозревателя отображать список более компактно. Как именно, зависит от конкретного Web-обозревателя.

Атрибут TYPE позволяет задать значок, которым будут помечаться строки списка; допустимо одно из трех значений: disc, circle и square, обозначающие соответственно круг с заливкой, круг без заливки и квадрат. По умолчанию строки маркируются кругом с заливкой.

Этот же атрибут поддерживается тегом <LI>; таким образом, можно оформить отдельные строки списка особенным образом. Вот только зачем это может понадобиться, я не знаю.

Примечание

Учтите, что атрибут TYPE в тегах <UL> и вложенных в него <LI> поддерживается только Navigator.

Перейдем к тегам <OL> и </OL>. Собственно, о них и так все сказано: нумерованный список задается так же, как и маркированный. Различие проявляется (помимо внешнего облика готового списка) в наборе поддерживаемых атрибутов. Их существенно больше.

```
<OL [COMPACT] [TYPE="A|a|I|i|1"] [START="{Номер}"]>
. . .
</OL>
```

Прежде всего, тег нумерованного списка также поддерживает атрибут TYPE, но теперь он значит совсем другое. Формат этого атрибута таков: TYPE="A|a|I|i|1".

- A — нумерует строки большими латинскими буквами.
- a — малыми латинскими буквами.
- I — большими римскими цифрами.
- i — малыми римскими цифрами.
- 1 — арабскими цифрами.

По умолчанию строки списка нумеруются арабскими цифрами. В отличие от <UL>, атрибут TYPE в теге <OL> (и вложенных в него <LI>) поддерживается и Navigator, и Internet Explorer.

Также Internet Explorer в <OL> поддерживает атрибут COMPACT. Новый атрибут START тега <OL> задает номер, с которого будет начинаться нумерация строк в списке.

В теге <LI> также поддерживаются уже знакомый вам атрибут TYPE (естественно, с поправкой на нумерованный список) и новый атрибут VALUE. Последний задает номер, которым будет помечена конкретная строка.

Теперь — традиционный пример. Это будет файл 1.3.htm.

```
<HTML>
<HEAD>
<TITLE>Пример Web-страницы со списками</TITLE>
</HEAD>
<BODY>
<H1>Пример Web-страницы со списками</H1>
<P>Простой маркированный список.</P>
<UL>
<LI>Первая строка.
<LI>Вторая строка.
</UL>
<P>Список, нумерованный римскими цифрами.</P>
<OL TYPE="I">
<LI>Первая строка.
<LI>Вторая строка.
<LI>Третья строка.
</OL>
<P>Продолжение — но уже маленькими латинскими буквами.</P>
<OL TYPE="a" START="4">
<LI>Четвертая строка.
<LI>Пятая строка.
</OL>
</BODY>
</HTML>
```

На рис. 1.3 показан результат.

Приведем еще одну разновидность списков, предоставляемых HTML. Это *списки определений*. Они, в частности, прекрасно подходят для создания списков каких-либо терминов и их описаний. Они создаются с помощью тегов <DL> и </DL>, между которыми помещаются собственно термины и их описания. Термины предваряются тегами <DT>, а определения — <DD>. Два последних тега, как и <LI>, — "сироты" без закрывающего парного тега. В теге <DL> поддерживается атрибут COMPACT.

Вместо того, чтобы долго описывать, как будет отображаться список определений, приведем пример. Сохраним его под именем 1.4.htm и посмотрим, что получится.

```
<HTML>
<HEAD>
```

```
<TITLE>Пример списка определений</TITLE>
</HEAD>
<BODY>
<H1>Пример списка определений</H1>
<DL>
<DT>Internet
<DD>Всемирная компьютерная сеть. Допустима также русскоязычная калька
 - Интернет.
<DT>HTML
<DD>Язык создания документов, предназначенных для публикации в сети
Интернет.
<DT>WWW
<DD>Общее название массива взаимосвязанных электронных документов,
размещенных в сети Интернет. Самый популярный интернет-сервис.
</DL>
</BODY>
</HTML>
```

Результат, отображаемый в окне Web-обозревателя, показан на рис. 1.4. Вы видите, что определения терминов отображаются с отступом слева, и сразу ясно, что есть что.

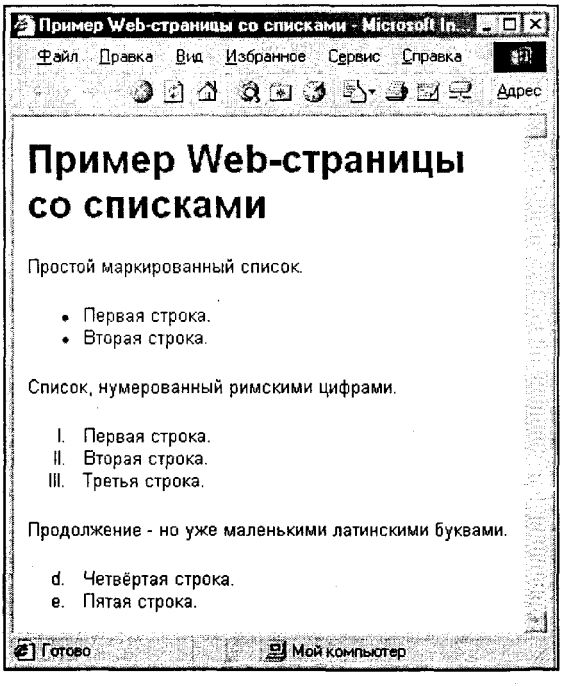


Рис. 1.3. Пример страницы со списками

Рис. 1.4. Список определений

Гиперссылки

Хватит о тексте! Давайте поговорим о чем-нибудь более волнующем. Например, о гиперссылках.

Именно гиперссылки связали все HTML-документы в единую электронную паутину. Благодаря этому мы можем с такой легкостью путешествовать по миру, не вставая из-за компьютера. Гиперссылки сделали HTML тем, чем он является теперь — стандартом для переносимых документов, используемых как в Интернете, так и в других областях. О, гиперссылки, вы наш и без того маленький мир сделали еще меньше!

Но прежде, чем мы начнем рассматривать гиперссылки и способы связи Web-документов друг с другом, нам нужно понять, что такое интернет-адрес. Здесь я немного отойду от HTML и углублюсь в дебри сетевых протоколов, в частности, IP.

Немного об интернет-протоколах

Как вы знаете, для того чтобы два (или более) компьютера, объединенных в сеть, понимали друг друга, они должны "разговаривать" на одном языке, т. е., обмениваться сигналами строго определенного формата. Такой строго определенный и зафиксированный в стандартах формат сигналов называется *протоколом*. Протоколы могут описывать *физические* (уровни и формы сигналов в линии, параметры самой линии, приемопередающих устройств: сетевых карт, модемов и т. п.) и *логические* (формат данных, пересылаемых по сети, алгоритмы сжатия информации, защиты от помех и обеспечения

секретности, способы именования компьютеров) характеристики сети. Существуют протоколы *низкого* и *высокого* уровня; в частности, физические протоколы — низкого уровня, а логические — высокого. Высокоуровневые протоколы базируются на низкоуровневых.

Так вот, протокол IP — низкоуровневый. Он описывает электрические параметры Всемирной сети и самые общие логические параметры передаваемых по ней сигналов. Протокол IP определяет, что информация передается небольшими порциями — *пакетами*, и что каждый пакет имеет своих *отправителя* и *получателя*.

На протоколе IP базируется более высокоуровневый протокол *TCP* (Transfer Control Protocol — протокол передачи данных). Он определяет способы защиты от помех и кое-какую дополнительную информацию, передаваемую в пакете вместе с данными.

Часто говорят, что Интернет базируется на связке протоколов TCP/IP. Эта "сладкая парочка" низкоуровневых протоколов служит основой для всех протоколов более высокого уровня, которые и обеспечивают передачу разнообразнейших данных.

HTML-документы передаются с использованием протокола *HTTP* (HyperText Transfer Protocol — протокол передачи гипертекста). Это протокол высокого уровня, манипулирующий значительно более крупными структурами данных, нежели рассмотренная нами "сладкая парочка". Он оперирует уже целыми документами.

С помощью этого протокола и общаются между собой Web-обозреватель и Web-сервер. Web-обозреватель принимает от пользователя адрес Web-сервера, с которого нужно получить документы, посылает на сервер специальный HTTP-запрос: блок текста, где указывает свой адрес, имя файла документа и еще множество параметров, включая характеристики клиентского компьютера, операционной системы и самого себя. Web-сервер ищет на дисках нужный файл (или выбирает некий файл по умолчанию, если имя файла в HTTP-запросе не указано) и отправляет его по указанному адресу.

Хитро составленное программное обеспечение — драйверы TCP/IP-протокола — незаметно для Web-сервера и Web-обозревателя разбивает документ на пакеты и впоследствии собирает его воедино. Также оно может исправить ошибки передачи, просто заново запросив у сервера искаженный ошибками пакет. Ни Web-обозреватель, ни Web-сервер об этом не беспокоятся, оперируя целыми файлами.

Интернет-адреса. IP-адрес и доменное имя

Выше я сказал, что протокол IP определяет формат пакета передачи данных. В числе прочих параметров (и, собственно, полезных данных, ради которых и затевалась вся эта кутерьма) он передает адреса источника и получателя

пакета. То есть, каждый компьютер, подключенный к Интернету, должен иметь уникальный адрес, чтобы посланный ему IP-пакет дошел по назначению.

IP-адрес — это четырехбайтовое значение, однозначно идентифицирующее компьютер. Оно имеет следующий вид: 123.45.67.89 и именно так записывается в диалоговых окнах настройки драйверов TCP/IP на вашем компьютере. В последнее время, правда, выработан новый стандарт IP-адреса — шестибайтный. (Для краткости старый, четырехбайтный формат IP-адреса называют IPv4, а новый — IPv6.) Связано это с тем, что четырехбайтных значений уже сейчас на всех не хватает, а в связи с тем, что в ближайшем будущем в Интернет выйдут, кроме обычных персональных компьютеров, еще и мобильные телефоны, "умные" холодильники, "интеллектуальные" утюги и прочая бытовая техника, доступных IP-адресов не останется совсем. Ну а шестибайтных адресов на всех хватит, во всяком случае, должно хватить...

Увидев в начале предыдущего абзаца пример IP-адреса, вы, наверно, воскликнули: "Как же так? А я в строке адреса Web-обозревателя набираю не 123.45.67.89, а http://www.aport.ru! Почему же у меня все работает?! Вероятно, какой-то фокус..." Ну, фокус не фокус, но придумано отлично. Вам не нужно запоминать умопомрачительные IP-адреса. Их помнят машины.

Эта система, превращающая символьные адреса вида http://www.aport.ru в IP-цифирь, называется *DNS* (Domain Name System — система доменных имен), а символьные адреса, соответственно, *доменными адресами* или *доменными именами*. Преобразованием занимаются специальные программы — DNS-серверы; они работают на компьютерах, постоянно подключенных к Сети. Web-обозреватель сам определяет, набрал пользователь доменное имя или IP-адрес, и посылает запрос на DNS-сервер, указанный в настройках драйвера TCP/IP. DNS-сервер отыскивает в своих базах данных имя и соответствующий ему IP-адрес и возвращает его Web-обозревателю.

Собственно, в приведенном выше примере доменным именем является только www.aport.ru. http:// — это обозначение протокола HTTP, которое указывать не обязательно.

Из чего состоит интернет-адрес

Когда вы набираете в строке адреса Web-обозревателя что-то вроде http://www.coolsite.ru, Web-сервер выдает вам страницу по умолчанию. Эта страница определяется в настройках программы Web-сервера для выдачи в случаях, если в интернет-адресе не указан файл документа, который хочет увидеть пользователь. Если же пользователю нужен какой-то конкретный документ, он может набрать адрес вида http://www.coolsite.ru/folder1 /folder2/file.html. При этом Web-серверу www.coolsite.ru будет послан запрос

на файл file.html, расположенный по пути /folder1/folder2. Таким образом, интернет-адрес может включать в себя не только адрес Web-сервера (IP- или доменный адрес), но и имя HTML-документа.

Теперь мы знаем три составные части интернет-адреса:

1. Необязательное обозначение протокола (http://).
2. Доменный или IP-адрес компьютера, на котором работает программа Web-сервера (www.coolsite.ru).
3. Имя файла, включающее или не включающее путь (/folder1/folder2 /file.html). Если имя файла отсутствует, Web-сервер выдает страницу по умолчанию, заданную в его настройках.

Вообще-то, интернет-адрес может содержать гораздо больше составляющих, но мы рассмотрим их позднее, когда вплотную займемся гиперссылками и Web-формами.

Собственно гиперссылки

Из всего вышесказанного следует, что для того, чтобы получить у Web-сервера какой-либо HTML-документ, нужно отправить ему HTTP-запрос, содержащий имя файла документа и адрес клиентского компьютера. Также выяснилось, что HTTP-запросы отправляют сами Web-обозреватели после того, как пользователь введет адрес нужного сервера. И уже давно известно, что в HTML-документах есть какие-то гиперссылки, с помощью которых можно перемещаться из одного документа в другой.

Попробуем свести все эти разрозненные данные воедино.

Гиперссылка — это специальный тег HTML, содержащий в качестве параметра адрес нужного документа. При активизации гиперссылки (в современных графических Web-обозревателях это производится щелчком левой кнопки мыши) на необходимый Web-сервер посылается HTTP-запрос с именем нужного документа. Как говорится, остальное — дело техники.

Ниже показан формат тега гиперссылки:

```
<A HREF="{Интернет-адрес}">. . .</A>
```

Прежде всего, заметьте, что адрес нужного документа указывается в атрибуте HREF. И еще: <A> — парный тег. Все, что помещено внутри него, и станет визуальным представлением гиперссылки в окне Web-обозревателя, тем, по чему должен щелкнуть мышью пользователь. Рассмотрим следующий пример.

```
<A HREF="http://www.coolsite.ru/folder1/folder2/file.html">Это
↳гиперссылка</A>
```

Этот тег форматирует слова "Это гиперссылка" как гиперссылку, ссылающуюся на документ, расположенный по адресу http://www.coolsite.ru/folder1

/folder2/file.html. Если пользователь подведет к этим словам курсор мыши, последний превратится в "указующий перст" — первый признак гиперссылки. Щелкнув по этим словам, пользователь получит на экране документ, на который ссылается гиперссылка.

Тег гиперссылки может иметь и такой вид:

```
<A HREF="/folder1/folder2/file.html">Это гиперссылка</A>
```

Это ссылка на документ /folder1/folder2/file.html, расположенный на том же сайте. То есть адрес сайта можно безболезненно опускать, если гиперссылка указывает куда-то в пределах текущего сайта. Также можно опустить путь и указать только имя файла страницы, если он находится в той же папке.

А теперь — традиционный пример. Давайте создадим страницу с гиперссылками, указывающими на уже созданные нами страницы-примеры. Это будет весьма поучительно. Сохраним страницу в файле 1.5.htm.

```
<HTML>
<HEAD>
<TITLE>Список страниц-примеров</TITLE>
</HEAD>
<BODY>
<H1>Список страниц-примеров</H1>
<UL>
<!-- Скорректируйте имена файлов, если нужно -->
<LI><A HREF="0.1.htm">Простейшая Web-страница</A>
<LI><A HREF="1.1.htm">Web-страница с двумя параграфами</A>
<LI><A HREF="1.2.htm">Web-страница с тегом &lt;PRE&gt;</A>
<LI><A HREF="1.3.htm">Списки</A>
<LI><A HREF="1.4.htm">Список определений</A>
</UL>
</BODY>
</HTML>
```

Результат показан на рис. 1.5.

Обратите внимание, что в этом примере мы ввели новый тег — `<!-- ... -->`. Этот тег обозначает комментарий, текст, не обрабатываемый Web-обозревателем и не отображаемый в его окне. При этом текст комментария указывается внутри тега — ситуация, не часто встречающаяся в HTML. Комментарии могут очень помочь тем, кто будет работать с HTML-кодом после вас, да и вам они могут пригодиться, когда вы вернетесь к своим документам по прошествии значительного времени.

Internet Explorer также поддерживает теги `<COMMENT>` и `</COMMENT>`. Здесь текст комментария указывается между парными тегами.

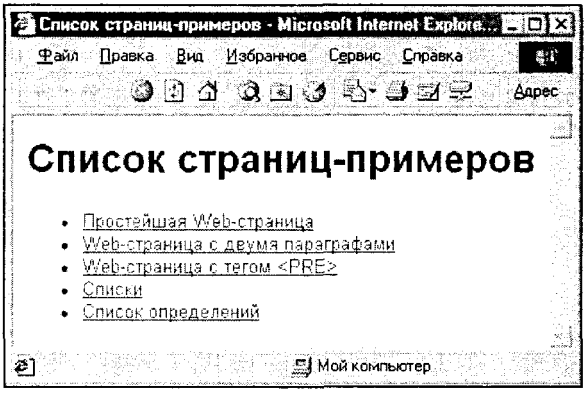


Рис. 1.5. Страница с гиперссылками

Перемещение к нужному фрагменту документа. "Якоря"

Иногда бывает очень нужно переместиться не к другому документу, а к другой части текущего документа. И в этом случае вам могут помочь гиперссылки, правда, специального вида.

Но сначала рассмотрим так называемые *"якоря"*. На редкость неудачное название, калька с английского "anchors". "Якорь" — это гиперссылка специального вида, помечающая какое-либо место в текущем документе, на которое будут ссылаться другие гиперссылки.

```
<A NAME="{Имя "якоря"}">. . .</A>
```

"Якорь", как вы видите, обозначается тем же парным тегом `<A>`, что и обычная гиперссылка. Атрибут `NAME` задает имя "якоря", на которое потом будут ссылаться другие гиперссылки.

```
<A NAME="chapter_1">Глава 1</A>
```

Чтобы сослаться на "якорь", указанный выше, напишем такую гиперссылку:

```
<A HREF="#chapter_1">Перейти к главе 1</A>
```

При щелчке по этой гиперссылке Web-обозреватель будет искать в текущем документе "якорь" по имени `chapter_1` и, если найдет, переместится к той части документа, где он определен. Как вы видите, это может очень пригодиться, если в начале каждой страницы вы делаете оглавление; тогда каждый пункт оглавления будет представлять собой гиперссылку, указывающую на нужный "якорь".

```
<HTML>
<HEAD>
<TITLE>Список страниц-примеров</TITLE>
</HEAD>
```

```
<BODY>
<H1>Статья</H1>
<P>Оглавление</P>
<P></P>
<P><A HREF="#chapter_1">Глава 1</A></P>
<P><A HREF="#chapter_2">Глава 2</A></P>
<P><A HREF="#chapter_3">Глава 3</A></P>
<HR>
<H2><A NAME="chapter_1">Глава 1</A></P>
. . .
<H2><A NAME="chapter_2">Глава 2</A></P>
. . .
<H2><A NAME="chapter_3">Глава 3</A></P>
. . .
</BODY>
</HTML>
```

Пример, приведенный выше, вы можете использовать как шаблон для создания страниц с оглавлениями.

Заметьте, что мы использовали пустой тег абзаца <P></P> для вставки в документ пустой строки. Вы можете также применить два тега разрыва строки <BR>, идущие подряд, но вариант с пустым <P></P> несколько изящнее.

Пользуясь "якорями", можно перемещаться к нужной части другого документа. В этом случае ваша гиперссылка будет иметь такой вид:

```
<A HREF="second.html#chapter_2">Перейти к главе 2 второго раздела</A>
```

Мы уже изучили три составные части интернет-адреса.

1. Необязательное обозначение протокола (http://).
2. Доменный или IP-адрес компьютера, на котором работает программа Web-сервера (www.coolsite.ru).
3. Имя файла, включающее или не включающее путь (/folder1/folder2 /file.html).

Теперь к ним добавилась четвертая:

4. Имя "якоря" (#chapter_2). Обозначает "якорь" в документе, к которому следует переместиться. Всегда предваряется символом "решетки" "#".

Ссылки на другие сервисы Интернета

Выше мы рассмотрели гиперссылки на другие Web-страницы. Но атрибут HREF тега <A> может содержать адрес, ссылающийся на иные сервисы Интернета, в частности на FTP-сервер или адрес электронной почты.

Как вы помните, любой интернет-адрес должен начинаться с обозначения протокола. Протокол HTTP, при помощи которого передаются Web-стра-

ницы, обозначается строкой "http://", и это обозначение является необязательным. Если вы его пропустите, Web-обозреватель по умолчанию поставит в интернет-адрес строку "http://".

Другие сервисы Интернета используют иные протоколы. И, соответственно, обозначения протоколов в интернет-адресах, указывающих на такие сервисы, должны быть другими. И — внимание — в этом случае обозначение протокола обязательно!

Давайте перечислим наиболее часто используемые сервисы Интернета и соответствующие им шаблоны гиперссылок. На самом деле интернет-сервисов, конечно, намного больше, но мы укажем только три самые популярные.

Электронная почта

Адрес электронной почты имеет следующий вид:

```
mailto:{Почтовый адрес}[?subject={Строка темы}][&cc={Адрес копии}]
↳[&bcc={Адрес "слепой" копии}][&body={Текст тела письма}]
```

Все это берется в кавычки и указывается в атрибуте HREF. При активизации такой гиперссылки откроется программа почтового клиента, установленная по умолчанию; причем поля адресов и темы, а возможно, и тело письма уже будут заполнены. Как видите, с HTML не соскучишься — где еще можно одним щелчком мыши написать целое письмо.

```
mailto:vasya@mail.sterver.ru?subject=Letter&cc=vasya@mail.home.ru&
↳bcc=chief@mail.sterver.ru@&body=Hi!
```

Поля subject, cc, bcc и body не являются обязательными. Возможен следующий вариант:

```
mailto:vasya@mail.sterver.ru
```

Новости

Ссылка на сервер новостей не представляет из себя ничего особенного:

```
nntp://{Адрес}
```

Ничем, кроме обозначения протокола, он от Web-адреса не отличается. При активизации такой гиперссылки запустится программа клиента новостей, подключится к заданному серверу и начнет скачивать с него списки доступных групп новостей.

FTP

Ссылка на FTP-сервер имеет такой вид.

```
ftp://[{Имя пользователя}@]{Адрес}
```

FTP — это протокол передачи файлов, один из самых старых сервисов Интернета наряду с почтой. FTP-серверы используются до сих пор в качестве легко развертываемых и поддерживаемых свалок файлов. Для просмотра содержимого такого сервера вы можете использовать свой Web-обозреватель, но если хотите что-то поместить на сервер, вам придется воспользоваться специальной программой FTP-клиента.

FTP-сервер допускает *анонимное* или *именное* подключение. Чтобы подключиться анонимно, вы просто указываете адрес FTP-сервера:

```
ftp://ftp.pomoika.ru
```

Чтобы подключиться под каким-либо зарегистрированным именем, придется указать имя пользователя:

```
ftp://vasya@ftp.pomoika.ru
```

Почти всегда при именном подключении FTP-сервер запрашивает пароль. При этом Web-обозреватель выводит на экран диалоговое окно, где вы будете должны ввести имя пользователя и пароль, причем имя пользователя уже может быть подставлено.

Нетекстовые элементы страниц

В этой главе мы рассмотрим все то, что не укладывается в рамки текста: графические изображения, звуки, видеофильмы и дополнительные элементы Web-страниц. Поскольку они не являются текстом, то сохраняются на дисках Web-сервера в виде отдельных файлов и запрашиваются Web-обозревателем отдельно от собственно Web-страниц.

Графические форматы

Сначала дадим некоторые разъяснения насчет форматов графических файлов, поддерживаемых Web-обозревателями.

Все Web-обозреватели, доступные на данный момент, поддерживают графические форматы *GIF* (Graphics Interchange Format — формат обмена графикой) и *JPEG* (Joint Picture Encoding Group — группа кодирования неподвижных изображений). Это стандартные форматы распространения графики в Интернете. Изображения при кодировке в одном из этих форматов сильно сжимаются и в результате имеют очень маленький размер при вполне сносном качестве изображения. Практически все современные графические программы поддерживают оба эти формата. GIF идеален для штриховых изображений (рисунки, состоящие из штрихов, схемы, графические элементы оформления Web-страниц), а в JPEG обычно кодируют полутоновую графику (фотографии, картины и т. п.). Более того, GIF-файл может содержать несколько графических изображений, показываемых на экране как

анимационный фильм. Такие анимированные GIF'ы можно часто встретить в Сети.

Internet Explorer также поддерживает форматы *PNG* (Portable Network Graphics — переносимая сетевая графика) и *BMP* (BitMaP — битовая матрица). PNG был недавно разработан с целью заменить и GIF, и JPEG и объединяет достоинства обоих этих форматов. К сожалению, пока он не получил должного распространения. Ну а BMP — стандартный формат хранения изображений в системе Windows.

Запомните расширения файлов различных графических форматов. GIF-файлы имеют расширение gif, JPEG-файлы можно встретить с расширениями jpg, jpe и jpeg, PNG-файлы — с расширением png, а BMP-файлы — bmp.

Теги вставки графики

Графические изображения вставляются в Web-страницы с помощью одинарного тега <IMG>. Ниже приведен его формат.

```
<IMG SRC="{Адрес файла изображения}" [WIDTH="{Ширина}"]
[HEIGHT="{Высота}"] [ALT="{Альтернативный текст}"]
[BORDER="{Толщина границы}"]
[ALIGN="left|right|top|texttop|middle|absmiddle|baseline|bottom|
➥absbottom"]
[VSPACE="{Расстояние до текста страницы по вертикали}"]
[HSPACE="{Расстояние до текста страницы по горизонтали}"]>
```

Как видите, у этого тега очень много атрибутов. Но не пугайтесь, обязательным является только атрибут SRC, задающий интернет-адрес файла графического изображения. Этот адрес может быть как полным, включающим в себя адрес Web-сервера (если файл находится на другом сервере), так и сокращенным, состоящим только из имени файла.

```
<IMG SRC="www.graphics-for-all.net/cat1/cat2/cat3/file1.gif">
<IMG SRC="/folder1/folder2/file2.gif">
```

Атрибуты WIDTH и HEIGHT позволяют задать соответственно ширину и высоту рисунка в пикселах. В этом случае Web-обозреватель еще до начала получения файла рисунка установит его правильные размеры, и дизайн вашей странички не исказится. Вы можете также специально изменить пропорции рисунка — например, чтобы получить комический эффект.

Атрибут ALT задает так называемый "альтернативный текст" — строку текста, которая будет выводиться в том месте, где должен появиться графический образ. Дело в том, что пользователь может отключить в Web-обозревателе показ графических изображений для ускорения "скачивания" страниц. И вместо графики на странице будут отображены пустые места. Поэтому нужно предусмотреть что-то, что будет эти пустые места заполнять: в дан-

ном случае, некий текст, описывающий, что увидит на этом месте пользователь, если все-таки включит показ графики.

Атрибут BORDER задает толщину рамки вокруг изображения. Задается она в пикселах.

Атрибут ALIGN позволяет нам управлять относительным местоположением изображения и "обтекающего" его текста. Он может принимать следующие значения:

- left — изображение смещается влево, а текст обтекает его справа;
- right — изображение смещается вправо, а текст обтекает его слева;
- top — изображение выравнивается по верху текущей строки;
- texttop — изображение выравнивается по вершине самого высокого символа текущей строки;
- middle — центр изображения выравнивается по базовой линии текущей строки;
- absmiddle — центр изображения выравнивается точно по центру текущей строки;
- baseline — нижний край изображения выравнивается по базовой линии текущей строки;
- bottom — нижний край изображения выравнивается по низу текущей строки;
- absbottom — нижний край изображения выравнивается по низу самого низко сидящего символа текущей строки.

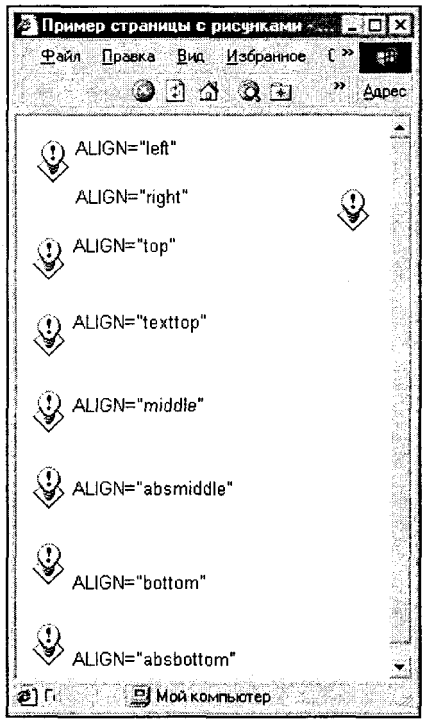
Ниже мы приведем пример с использованием различных значений атрибута ALIGN. А сейчас рассмотрим последние два атрибута — HSPACE и VSPACE. Эти атрибуты задают промежуток между изображением и "обтекающим" его текстом. Значение промежутка задается в пикселах.

А теперь — обещанный пример. Сохраним его в файле под именем 1.6.htm.

```
<HTML>
<HEAD>
<TITLE>Пример страницы с рисунками</TITLE>
</HEAD>
<BODY>
<P><IMG SRC="Tips.gif" ALIGN="left">ALIGN=&quot;left&quot;</P>
<P><IMG SRC="Tips.gif" ALIGN="right"> ALIGN=&quot;right&quot;</P>
<P><IMG SRC="Tips.gif" ALIGN="top"> ALIGN=&quot;top&quot;</P>
<P><IMG SRC="Tips.gif" ALIGN="texttop"> ALIGN=&quot;texttop&quot;</P>
<P><IMG SRC="Tips.gif" ALIGN="middle"> ALIGN=&quot;middle&quot;</P>
<P><IMG SRC="Tips.gif" ALIGN="absmiddle"> ALIGN=&quot;absmiddle&quot;</P>
<!-- Значение baseline пропущено -->
```

```
<P><IMG SRC="Tips.gif" ALIGN="bottom"> ALIGN=&quot;bottom&quot;</P>
<P><IMG SRC="Tips.gif" ALIGN="absbottom"> ALIGN=&quot;absbottom&quot;</P>
</BODY>
</HTML>
```

Графический файл вы можете найти в папке Web, расположенной там, где у вас установлена Windows. Эту вложенную папку создает при инсталляции Internet Explorer 4.0 или более поздняя его версия.



Результат вы можете увидеть на рис. 1.6.

Вообще-то, для того чтобы получить удовлетворительный результат, придется поэкспериментировать со значениями атрибутов ALIGN, VSPACE и HSPACE. К тому же, различные программы Web-обозревателей могут реагировать на эти атрибуты по-разному.

Рис. 1.6. Web-страница с рисунками

Изображения-гиперссылки

Графическое изображение можно очень просто превратить в гиперссылку. Для этого используйте все тот же тег <A>.

```
<A HREF="www.coolsite.ru">
<IMG SRC="www.coolsite.ru/images/banner.gif">
</A>
```

Изображения-карты

Создавать изображения-гиперссылки мы научились. Следующий шаг — *изображения-карты*.

Изображения-карты — это графические изображения, различные части которых представляют собой отдельные гиперссылки. Вы, наверно, видели в Сети такие штуковины: большой рисунок, отдельные части которого позволяют перейти к разным документам. Создаются изображения-карты немного сложнее, чем простые изображения-гиперссылки, но выглядят очень эффектно.

Выше я перечислил несколько атрибутов тега <IMG>. Теперь добавляю еще один — USEMAP={[Адрес страницы]}#{Имя карты}. Это атрибут одним своим присутствием превращает обычный рисунок в изображение-карту. А в качестве значения он задает адрес и имя списка областей, в котором описаны форма и расположение "горячих" областей, находящихся на изображении, и гиперссылки, привязанные к этим областям.

Сам список областей задается с помощью пары тегов <MAP> и </MAP>. Внутри них перечисляются "горячие" области, обозначенные тегами <AREA>.

```
<MAP NAME="{Имя карты}">
<AREA [SHAPE="rect|circle|poly"] COORDS="{x},{y},..."
HREF="{Адрес}"|NOHREF>
. . .
</MAP>
```

Атрибут NAME тега <MAP> задает имя списка областей, то самое имя, на которое ссылается атрибут USEMAP изображения-карты. Это понятно и не требует дополнительных разъяснений.

Поэтому рассмотрим подробнее атрибуты тега <AREA>.

Атрибут SHAPE задает форму "горячей" области. Атрибут COORDS перечисляет координаты, необходимые для построения этой области. SHAPE может принимать следующие значения:

- rect — задает область в виде прямоугольника. Соответственно, атрибут COORDS имеет вид COORDS="{X1},{Y1},{X2},{Y2}", где X1 и Y1 — координаты верхнего левого, а X2 и Y2 — правого нижнего угла прямоугольника. Это значение атрибута SHAPE по умолчанию;
- circle — задает круглую область. В этом случае атрибут COORDS имеет вид COORDS="{X центра},{Y центра},{Радиус}";
- poly — задает область в виде сложного многоугольника. Атрибут COORDS имеет вид COORDS="{X1},{Y1},{X2},{Y3},{X3},{Y3}...", где Xn и Yn — координаты соответствующей точки.

Атрибут HREF, как вы уже поняли, задает интернет-адрес. Вместо атрибута HREF может быть задан атрибут NOHREF. Он задает область, не связанную ни с каким интернет-адресом. Это позволяет создавать оригинальные изображения-карты, например карту в виде бублика, "дырка" которого никуда не указывает.

И в заключение — пример:

```
<IMG HREF="map.gif" USEMAP="#map1">
. . .
<MAP NAME="map1">
<AREA SHAPE="circle" COORDS="50,50,30" HREF="page1.html">
<AREA SHAPE="circle" COORDS="50,150,30" HREF="page2.html">
<AREA SHAPE="circle" COORDS="50,50,10" NOHREF>
<AREA SHAPE="circle" COORDS="50,150,10" NOHREF>
<AREA SHAPE="rect" COORDS="0,90,200,100" HREF="appendix.html">
</MAP>
```

Вот и все об изображениях-картах.

Изображения-карты, рассмотренные нами, называются *клиентскими*, т. к. обрабатываются на стороне клиента. На самом деле, существуют еще *серверные* изображения-карты, за обработку которых отвечает Web-сервер. Но мы не будем их рассматривать, поскольку это выходит за рамки книги.

Фильмы

Вы научились помещать на Web-страницы изображения. Логично сделать следующий шаг — заняться видеороликами. Представьте себе: заходит на сайт посетитель и смотрит кино...

Конечно, не все будет столь радужно. Все более-менее приличные фильмы имеют огромный размер, и на Web-страницах реально размещать только совсем маленькие ролики, чтобы посетитель не успел потерять терпение, ожидая окончания их загрузки. Также существует проблема авторских прав, игнорировать которые зачастую просто опасно.

Вообще-то, анимированные GIF-файлы тоже можно рассматривать как видеофильмы (да это, собственно, и есть фильмы). А они размещаются на страницах с помощью уже знакомого вам тега `<IMG>`. Но дело в том, что на страницах также можно разместить и фильмы в формате *AVI* (Audio-Video Interlaced — перемежающиеся аудио-видео), являющемся стандартом в системе Windows. И делается это с помощью того же тега `<IMG>`. А точнее, его атрибута `DYNSRC` (и еще некоторых атрибутов, которые мы также рассмотрим). Беда лишь в том, что этот атрибут (и все сопутствующие ему атрибуты) поддерживает только Internet Explorer.

```
<IMG SRC="{Адрес файла статичного изображения}"
[DYNSRC="{Адрес файла видеофильма}"]
[LOOP="-1|0|{Количество повторений}"]
```

Сначала Web-обозреватель запрашивает у Web-сервера файл изображения, адрес которого указан в атрибуте `SRC`, и отображает его на странице. Потом Web-обозреватель, если он — Internet Explorer, запрашивает файл фильма,

адрес которого указан в атрибуте DYNSRC, и выводит его вместо статичного изображения. В противном случае атрибут DYNSRC просто игнорируется — любой Web-обозреватель обязан игнорировать теги и атрибуты, которые не понимает.

Атрибут LOOP устанавливает количество повторений фильма. Если в качестве значения задано -1 или 0, то фильм будет повторяться бесконечно. По умолчанию фильм демонстрируется один раз.

И, само собой, не забывайте обо всех уже изученных нами атрибутах тега <IMG>: ALIGN, WIDTH, HEIGHT и пр.

Традиционный пример:

```
<IMG SRC="SantaBarbaraLogo.jpe" DYNSRC="SantaBarbara.avi" LOOP="-1"
BORDER="3" ALT="Долгое кино">
```

Ну вот и все. Теперь вы можете оживить свои Web-страницы видеороликами. Только имейте чувство меры: не выкладывайте в Сеть "Санта-Барбару". Хотя бы потому, что все авторские фильмы защищены авторским правом (извините за тавтологию, но это так). То, что вы видели выше, просто шутка...

Фоновая музыка

Internet Explorer обладает еще одной фирменной особенностью: он позволяет внедрить на Web-страницу фоновую музыку, которая будет звучать в то время, когда пользователь просматривает страницу. Делается это с помощью тега, формат которого приведен ниже:

```
<BGSOUND SRC="{Адрес аудиофайла}" [BALANCE="{Значение стереобаланса}"]
[LOOP="-1|0|{Количество повторений}"] [VOLUME="{Громкость}"]>
```

Атрибут SRC задает адрес аудиофайла с фоновой музыкой. Имейте только в виду, что аудиофайлы должны быть в форматах *WAV* (WAVe — "волна", стандартный формат аудиофайлов в Windows) или *MIDI* (Music Instruments Digital Interface — цифровой интерфейс музыкальных инструментов).

Атрибут BALANCE задает значение стереобаланса — положение источника звука в пространстве. Доступны значения из диапазона от -10000 до 10000, представляющие крайние значения (левое или правое — зависит от настроек звуковой подсистемы клиентского компьютера). По умолчанию звук воспроизводится в центре (0).

Атрибут LOOP уже должен быть вам знаком — он устанавливает количество повторений аудиоролика. Если в качестве значения задано -1 или 0, "эта музыка будет вечной".

Атрибут VOLUME устанавливает громкость воспроизведения музыки. Доступны значения от -10000 (минимальная громкость) до 0 (максимальная). Значение

по умолчанию — 0. И опять же реальная громкость звука будет зависеть от настроек клиентского компьютера.

```
<BGSOUND SRC="Stoa_Urthona.mid" LOOP="-1">
```

Единственная сложность заключается в том, что все изученные нами в этой главе теги — нестандартные. Они не стандартизованы W3C, не включены в спецификацию HTML и не поддерживаются ни одним обозревателем, кроме Internet Explorer. И если вы хотите, чтобы Web-страницы с музыкой нормально просматривались и в других Web-обозревателях, например Navigator, используйте более подходящий прием. Какой — мы сейчас рассмотрим.

Расширения Web-обозревателя

Очень многие современные программы поддерживают концепцию расширений. Расширения (plug-ins) — это специально написанные модули, обычно динамические библиотеки Windows DLL (Dynamin Link Library — библиотека динамической компоновки) — фрагменты программного кода, которые могут быть использованы разными программами). Они встраиваются в программу и добавляют к ней какие-либо новые функции. Существует обширный рынок коммерческих расширений для популярных программ: текстовых редакторов, СУБД, графических пакетов, программ редактирования звука и трехмерной анимации. И еще больше существует расширений бесплатных, которые можно найти в любом файловом архиве.

Не избежали этой участи и Web-обозреватели. Вначале механизмом поддержки расширений обзавелся Netscape Navigator, а потом по его стопам пошел и Microsoft Internet Explorer. С помощью расширений эти программы получили возможность поддерживать различные форматы файлов, в основном, мультимедийных: фильмы AVI и *QuickTime* (стандартный видеоформат фирмы Apple), музыка в формате MIDI, трекерных модулей (трекеры — это целый класс простых программ для сочинения музыки, сделавших не одного пользователя Интернет настоящим композитором; порождаемые ими файлы называются трекерными модулями), различные форматы графических изображений. Используя поддержку расширений, Web-дизайнер просто размещает в HTML-коде специальный тег, указывает в атрибутах адреса файла данных и дистрибутивного файла расширения. И если нужное расширение на компьютере клиента не установлено, Web-обозреватель сам запросит и установит его, не беспокоя пользователя.

Рассмотрим формат этого волшебного тега.

```
<EMBED SRC="{Адрес файла данных}" [HIDDEN="true|false"]
[PLUGINSPACE="{Адрес Web-страницы с инструкциями по инсталляции
расширения}"] [TYPE="{Тип MIME файла данных}"]>
[<NOEMBED>. . .</NOEMBED>]
</EMBED>
```

Адрес файла данных (мультимедиа, графика, просто какие-то данные — расширения Web-обозревателей бывают разные) задается в атрибуте SRC.

Атрибут HIDDEN устанавливает, будут показаны или нет органы управления модулем расширения, если таковые у него есть: значение false разрешает, а true — запрещает показ. По умолчанию органы управления расширением появляются на экране (значение false).

Атрибут PLUGINSPACE задает адрес Web-страницы с инструкциями по установке расширения. Также на этой странице обязательно должна быть ссылка на дистрибутивный файл — иначе какой от нее толк.

Атрибут TYPE позволяет задать тип *MIME* для данного файла данных. MIME (Multipurpose Internet Mail Extensions — многоцелевые расширения почты Интернета) были разработаны как стандарт идентификации различных типов файлов для отправки их по почте, но впоследствии их функции были расширены и на другие сервисы Интернета. Для типа MIME, который прописывается в Реестре Windows и используется различными программами, в частности, указывается расширение файла и программа, с помощью которой данный тип файлов будет обрабатываться. Однако Navigator ведет свою базу данных типов MIME и соответствующих им программ.

В табл. 1.6 приведены типы MIME для наиболее часто встречающихся типов файлов. (Как видите, очень многие файлы имеют свой тип MIME: текстовые, HTML, документы Microsoft Word и Excel, исполняемые файлы и т. п.)

Таблица 1.6. *Типы MIME*

Тип файлов	Тип MIME
Архив RAR	application/x-tar
Архив ZIP	application/x-zip-compressed
Аудио- или видеозапись ASF	video/x-ms-asf
Аудио- или видеозапись WMV	video/x-ms-wmv
Аудиозапись AIFF	audio/aiff
Аудиозапись AU	audio/basic
Аудиозапись MIDI	audio/mid
Аудиозапись MP3	audio/mpeg
Аудиозапись WAV	audio/wav
Аудиозапись WMA	audio/x-ms-wma
Видеозапись AVI	video/avi
Видеозапись Indeo (IVF)	video/x-ivf

***Таблица 1.6** (окончание)*

Тип файлов	Тип MIME
Видеозапись MPEG	video/mpeg
Визитная карточка, используемая почтовыми программами для хранения данных об адресате	text/x-vcard
Графический файл ART	image/x-jg
Графический файл BMP	image/bmp
Графический файл GIF	image/gif
Графический файл JPEG	image/jpeg
Графический файл Macromedia Flash	application/futuresplash
Графический файл TIFF	image/tiff
Документ Adobe Acrobat	application/pdf
Документ HTML	text/html
Документ Microsoft Excel	application/x-msexcel
Документ Microsoft Word	application/msword
Документ RTF	application/msword
Документ XML	text/xml
Приложение	application/x-msdownload
Приложение HTML (HTA)	application/hta
Таблица стилей HTML	text/css
Текстовый документ	text/plain

Я постарался привести в этой таблице как можно больше типов MIME, которые могут пригодиться при создании Web-страниц. Возможно, не все перечисленные типы файлов вам знакомы; с некоторыми из них мы встретимся далее в этой книге, другие пока оставим в покое.

Тег <EMBED> также поддерживает некоторые атрибуты тега <IMG>, а именно: ALIGN, BORDER, HEIGHT, HSPACE, VSPACE и WIDTH.

Теперь поговорим о парном теге <NOEMBED> и </NOEMBED>. Он предназначен на тот случай, если Web-страницу будут просматривать в Web-обозревателе, совершенно не поддерживающем расширения, например, Navigator 1.0 или Internet Explorer 2.0 (более новые версии уже поддерживают). В этом случае тег <EMBED> игнорируется как неизвестный, а вместо расширения отображается текст, помещенный внутри тега <NOEMBED>. Таким образом, пользователи "древних" Web-обозревателей хоть что-то увидят.

В двух предыдущих разделах мы привели примеры внедрения в Web-страницу видеофильма и фоновой музыки. Беда этих примеров в том, что их поддерживает только Internet Explorer. Давайте перепишем их так, чтобы сделать понятными и для Navigator.

```
<EMBED SRC="SantaBarbara.avi" TYPE="video/avi" BORDER="3"
ALT="Долгое кино" HIDDEN="true">
<EMBED SRC="Stoa_Urthona.mid" TYPE="audio/mid"
ALT="Здесь должна быть музыка" HIDDEN="true">
```

Казалось бы, все прекрасно. Но нет! Дело в том, что и тег <EMBED> — о, ужас! — не стандартизирован W3C. Да, он поддерживается двумя популярнейшими Web-обозревателями и, по всей вероятности, будет еще какое-то время поддерживаться, но кто знает, что случится в отдаленном будущем. Не откажутся ли производители Web-обозревателей от поддержки нестандартных, фирменных тегов? Неизвестно...

Введение в элементы ActiveX

Проигнорировав существующую технологию расширений, W3C сделал выбор в пользу принципиально новой. Ей стала предложенная Microsoft технология компактных объектов *ActiveX*, явившаяся дальнейшим развитием довольно старого стандарта OLE.

Что это такое?

OLE (Object Linking and Embedding — связывание и встраивание объектов) — это стандарт, позволяющий создавать сложные документы, содержащие разнородную информацию. При этом каждый тип информации будет обрабатываться своим отдельным приложением. Пример составного документа OLE — документ Word с встроенной в него таблицей Excel. Вы, наверно, уже имели дело с такими штуками. Для того, чтобы отредактировать внедренную таблицу, вызывается Excel, "встраивающийся" внутрь Word. При этом Word является *клиентом* OLE, документ Word — *контейнером*, Excel — *сервером* (не путать с Web-сервером и вообще с сетевыми серверами), а таблица — *внедренным объектом*.

Word и Excel — приложения очень "тяжелые". И в качестве альтернативы Microsoft предложила разработчикам создавать небольшие OLE-серверы и "упаковывать" их в DLL. Назвали их элементами ActiveX. Фактически, это те же самые расширения Web-обозревателя, но несравнимо более универсальные. В самом деле, в современных операционных системах от Microsoft буквально все базируется на технологии OLE. А значит, элементы ActiveX можно применять везде, в любом таком приложении.

Можно провести аналогию между элементами ActiveX и Java-апплетами. У них есть много общего: они не являются самостоятельными программами,

исполняются внутри Web-обозревателя и распространяются через Сеть. Однако ActiveX представляют собой откомпилированный двоичный код, который исполняется непосредственно процессором, в то время как Java-апплеты нуждаются в виртуальной Java-машине, переводящей байт-код Java в команды процессора, и соответственно, работают медленнее. С одной стороны, ActiveX сделаны более универсальными и могут использоваться в разных программах, с другой, Java-апплеты могут исполняться на разных компьютерных платформах, даже несовместимых.

Рассматривать работу с элементами ActiveX (и Java-апплетами) мы здесь не будем. Дело в том, что для этого нужно сначала изучить программирование на JavaScript и объектную модель документа. Мы до них еще не добрались, а пока имейте в виду, что элементы ActiveX — самый перспективный способ внедрения в Web-страницы нетекстовой информации.

Настройка параметров страницы

Многие текстовые процессоры уровня Microsoft Word предлагают такую возможность, как цветной фон страницы. Вы можете сделать виртуальную страницу с текстом не белой, а цветной, а то и вообще использовать в качестве подложки какой-либо рисунок. Конечно, это не всегда делает Web-страницы лучше, но иногда сменить фон все-таки полезно.

HTML предлагает аналогичные возможности. Вы можете сменить фон страницы, используя атрибуты вашего старого знакомого — тега <BODY>. Более того, вы можете также управлять цветом текста и гиперссылок и изменять расстояния между границами окна Web-обозревателя и его содержимым.

Приведем формат тега <BODY> со всеми его атрибутами.

```
<BODY [BACKGROUND="{Адрес файла фонового рисунка}"]
[BGCOLOR="{Цвет фона страницы}"] [TEXT="{Цвет текста}"]
[LINK="{Цвет гиперссылок}"]
[VLINK="{Цвет уже посещенных гиперссылок}"]
[ALINK="{Цвет активной гиперссылки}"]
[BGPROPERTIES="fixed"]
[LEFTMARGIN="{Левая граница}"] [RIGHTMARGIN="{Правая граница}"]
[TOPMARGIN="{Верхняя граница}"] [BOTTOMMARGIN="{Нижняя граница}"]
[SCROLL="yes|no"]>
. . .
</BODY>
```

Атрибут BACKGROUND задает адрес файла рисунка, который будет использоваться для фонового заполнения страницы. Если геометрические размеры рисунка меньше, чем страницы, то рисунок многократно повторяется, образуя что-то, похожее на мозаику.

Атрибут BGCOLOR задает цвет фона страницы. Вы можете подумать, что он имеет смысл, только если не задан атрибут BACKGROUND, но это не так. Дело в том, что Web-обозреватель сначала отображает текст Web-страницы, а уж потом — все ее нетекстовые элементы. То есть, на экране сначала появится текст, а потом все рисунки, в том числе фоновые. По умолчанию Web-страница имеет белый или серый фон (это зависит от конкретного Web-обозревателя), и если вы выбрали для нее темный фоновый рисунок, просматривающий ее пользователь будет шокирован. Так что стоит задать в атрибуте BGCOLOR цвет, наиболее близкий к общему тону вашего фонового рисунка.

Атрибут TEXT задает цвет текста (по умолчанию — черный).

Атрибуты LINK, VLINK и ALINK задают соответственно цвета непосещенных, посещенных и активной гиперссылок (по умолчанию — синий, бордовый и красный соответственно). Активная гиперссылка — та, на которую наведен курсор мыши.

Атрибут BGPROPERTIES поддерживается только Internet Explorer. Его единственное допустимое значение fixed позволяет "закрепить" фоновый рисунок. По умолчанию при прокручивании Web-страницы в окне обозревателя вместе с текстом прокручивается и фон. Добавив атрибут BGPROPERTIES="fixed", вы можете зафиксировать фон и получить таким образом оригинально выглядящую страницу. Но учтите, что на маломощных компьютерах такие страницы могут прокручиваться очень медленно.

Атрибуты LEFTMARGIN, RIGHTMARGIN, TOPMARGIN и BOTTOMMARGIN опять же поддерживаются только Internet Explorer и задают расстояния между содержимым Web-страницы и соответственно левой, правой, верхней и нижней границей окна Web-обозревателя. Значения задаются в пикселах.

Атрибут SCROLL позволяет убрать полосы прокрутки, появляющиеся, если Web-страница не помещается в окно Web-обозревателя. Для этого дайте ему значение no. Значение yes соответственно разрешает полосы прокрутки (это поведение по умолчанию). Будьте осторожны с данным атрибутом, ведь пользователь может и не увидеть всей Web-страницы, если вы отключите полосы прокрутки. Наверно, именно поэтому Navigator опять же его не поддерживает.

Под конец — пример.

```
<BODY BACKGROUND="Back.gif" BGCOLOR="green" TEXT="white">
```

И еще. Будьте разумны, выбирая цветовую гамму для Web-страницы! В Интернете хватает совершенно, извините, уродских страничек, выполненных в тошнотворнейших цветовых гаммах. Лучше оставьте все цвета в покое, если не уверены, хорошо ли будет выглядеть ваша страница. Поверьте, человечество за всю свою историю не придумало ничего лучше белой бумаги и черных чернил!

Фреймы

Что такое фреймы

Фреймы в свое время произвели настоящую революцию в Web-дизайне. И это притом, что W3C стандартизировал их совсем недавно. Долгое существование "между небом и землей" привело к тому, что фреймы — один из самых "нестандартных" элементов HTML. В самом деле, создавая страницу в виде набора фреймов, невозможно предугадать, как они будут показаны различными Web-обозревателями. Приходится многократно тестировать страницу во всех возможных браузерах, "шаманить" со значениями атрибутов, чтобы получить не то что идеальный — более-менее приемлемый результат.

И все же фреймы используют. И часто.

Что такое фреймы? Грубо говоря, окно Web-обозревателя делится на несколько независимых "форточек", в каждой из которой может отображаться своя Web-страница. Это производится с помощью специальной Web-страницы, где описываются параметры такого разделения. Она называется *набором фреймов*, а отдельные "форточки" окна обозревателя — фреймами. Каждый фрейм можно рассматривать как отдельное окошко Web-обозревателя; оно и ведет себя как отдельное окно.

Какие преимущества предоставляют фреймы? Огромные! Представьте себе набор из трех фреймов. В одном, расположенном вдоль верхнего края страницы, располагается красивое название сайта. В другом, тянущемся вертикально вдоль левого края страницы, находится набор ссылок — своеобразное оглавление сайта. И в третьем, самом большом фрейме выводится собственно содержимое. Это классический дизайн, используемый в превеликом множестве сайтов и уже ставший едва ли не банальностью. Удобство заключается в том, что когда пользователь щелкает по гиперссылке, обновляется только один фрейм — тот, где выводится содержимое. Фреймы с названием и оглавлением сайта при этом остаются на месте.

Да что болтать попусту — давайте лучше сделаем пример. Это будет небольшой сайт, состоящий из трех страниц.

Сначала напишем код страницы набора фреймов и сохраним ее в файле под именем 1.7.htm.

```
<HTML>
<HEAD>
<TITLE>Сайт с фреймами</TITLE>
</HEAD>
<FRAMESET ROWS="50,*">
<FRAME SRC="1.7.header.htm" NAME="frmHeader">
<FRAMESET COLS="100,*">
<FRAME SRC="1.7.contents.htm" NAME="frmContents">
```

```
<FRAME SRC="1.7.1.htm" NAME="frmBody">
</FRAMESET>
</FRAMESET>
</HTML>
```

Обратите внимание, что вместо пары тегов <BODY> и </BODY> используются соответственно <FRAMESET> и </FRAMESET> (выделены жирным). И еще одна важная деталь: в странице набора фреймов указывается только описание набора фреймов — никакого содержимого!

Теперь страница заголовка (файл 1.7.header.htm). Она простая.

```
<HTML>
<HEAD>
<TITLE>Заголовок</TITLE>
</HEAD>
<BODY TEXT="blue">
<H1>Сайт с фреймами</H1>
</BODY>
</HTML>
```

Здесь мы указали тег заголовка страницы <TITLE> и </TITLE>. Этого требуют стандарты, хотя для страниц, которые будут отображаться во фреймах, это не обязательно: в данном случае всегда отображается заголовок набора фреймов.

Страница оглавления заметно интереснее (файл 1.7.contents.htm).

```
<HTML>
<HEAD>
<TITLE>Оглавление</TITLE>
</HEAD>
<BODY>
<P><A HREF="1.7.1.htm" TARGET="frmBody">Страница 1</A></P>
<P><A HREF="1.7.2.htm" TARGET="frmBody">Страница 2</A></P>
<P><A HREF="1.7.3.htm" TARGET="frmBody">Страница 3</A></P>
</BODY>
</HTML>
```

Здесь видно, что у тега <A> появился новый атрибут TARGET. О нем мы поговорим позже, а пока запомните, что с его помощью можно задать фрейм, отображающий страницу, на которую указывает ссылка.

Странички, которые будут изображать из себя содержимое сайта, совсем просты и отличаются друг от друга только выводимым текстом. Сохраните их в файлах 1.7.1.htm, 1.7.2.htm и 1.7.3.htm.

```
<HTML>
<HEAD>
<!-- Подставьте соответствующий номер -->
<TITLE>Содержимое 1 (2, 3)</TITLE>
```

```
</HEAD>
<BODY>
<!-- Здесь тоже подставьте соответствующий номер -->
<H1>Страница 1 (2, 3)</H1>
<P>Это первая (вторая, третья) страница нашего сайта.</P>
</BODY>
</HTML>
```

Теперь откройте в Web-обозревателе страницу 1.7.htm, определяющую набор фреймов. Результат показан на рис. 1.7.

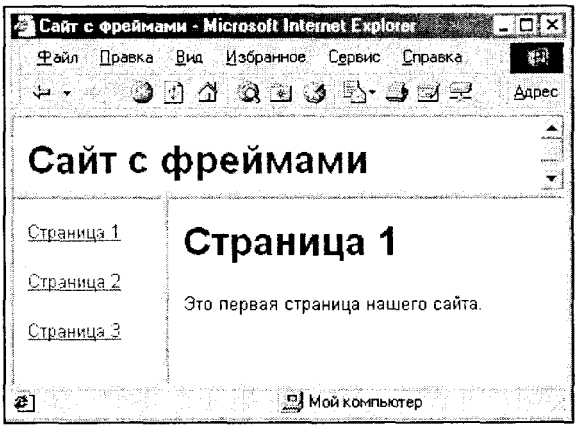


Рис. 1.7. Набор фреймов

Вы видите страницу, разделенную на три части толстыми серыми линиями, которые, кстати, можно двигать мышью, меняя размеры этих частей. Вот так и выглядит набор фреймов. Если щелкнуть по какой-либо гиперссылке в левом фрейме, содержимое соответствующей страницы отобразится в самом большом фрейме, расположенном справа. В этом и есть важнейшее преимущество фреймов: отдельным фреймом можно управлять как отдельным окном Web-обозревателя. И в то же время фрейм — часть вашей Web-страницы.

Набор фреймов

Набор фреймов описывается с помощью пары тегов <FRAMESET> и </FRAMESET>. Эта пара тегов заменяет хорошо знакомые вам <BODY> и </BODY>.

```
<FRAMESET ROWS|COLS="{Размеры фреймов}" [BORDER="{Толщина границы}"]
[BORDERCOLOR="{Цвет границы}"] [FRAMEBORDER="yes|no|0"]
[FRAMESPACING="{Расстояние между фреймами}"]>
. . .
</FRAMESET>
```

Сначала поговорим об атрибутах ROWS и COLS. Именно они определяют разбиение Web-страницы на фреймы.

Атрибут ROWS задает разбиение на строки. То есть, пространство страницы-набора фреймов будет разделено на две строки, разделенные границей (вообще-то, границу можно убрать). Ниже мы приведет формат этого атрибута.

```
ROWS="{Абсолютное значение размера}|
↳{Относительное значение размера}%}|*"
```

Абсолютное значение задает высоту строки в пикселах. Относительное значение — в процентах от высоты окна Web-обозревателя; в этом случае не забудьте поставить знак процента "%". Символ звездочки "*" указывает, что под строку нужно отвести все оставшееся пространство, и всегда используется для указания размера последнего фрейма в наборе. Например:

```
ROWS="100,20%,*"
```

Здесь мы определили три фрейма-строки: один высотой 100 пикселов, другой, занимающий 20% высоты окна, и третий, которому отдали все остальное пространство.

Атрибут COLS также определяет размеры фреймов, но на этот раз имеющих вид столбцов. Он имеет точно такой же формат, как и ROWS, поэтому я не буду его здесь приводить.

Атрибут BORDER позволяет задать в пикселах толщину границы между фреймами.

Атрибут BORDERCOLOR задает цвет границы между фреймами. По умолчанию граница имеет серый цвет.

Атрибут FRAMEBORDER позволяет включить или отключить показ границы между фреймами. Он поддерживается и Navigator, и Internet Explorer, но поразному. Navigator принимает значение yes для включения показа границы и no для отключения. Internet Explorer позволяет задавать также значения 1 и 0. По умолчанию граница показывается.

Атрибут FRAMESPACING позволяет задать расстояние между фреймами. Эта величина задается в пикселах.

Набор фреймов представляет собой контейнер, в котором хранятся описания всех фреймов. Однако, как мы уже видели, набор фреймов может содержать и другие наборы фреймов. Такие наборы фреймов называются *вложенными*. Это единственный способ создать более-менее сложную структуру фреймов, ведь отдельный набор фреймов может содержать только фреймы-строки или фреймы-столбцы.

Сами фреймы описываются с помощью другого тега, который мы ниже рассмотрим.

Тег <FRAME>

Отдельные фреймы набора описываются тегом <FRAME>. Заметьте, что это одиночный тег.

```
<FRAME [SRC="{Адрес отображаемой во фрейме страницы}"]
[NAME="{Имя фрейма}"] [MARGINWIDTH="{Горизонтальный отступ}"]
[MARGINHEIGHT="{Вертикальный отступ}"] [SCROLLING="yes|no|auto"]
[NORESIZE] [FRAMEBORDER="yes|no|0"] [BORDERCOLOR="{Цвет границы}"]>
```

Адрес страницы, которая будет отображаться во фрейме, задается в атрибуте SRC.

Атрибут NAME задает уникальное имя фрейма. Оно используется впоследствии в атрибуте TARGET тега <A>. Если атрибут NAME пропущен, фрейм остается непоименованным.

Атрибуты MARGINWIDTH и MARGINHEIGHT устанавливают расстояние между границей фрейма и его содержимым соответственно по горизонтали и вертикали. Эта величина задается в пикселах. Минимально возможное значение равно 1; по умолчанию Web-обозреватель сам выбирает значение, наиболее подходящее к конкретному случаю.

Атрибут SCROLLING запрещает или разрешает отображение полос прокрутки во фрейме. Значение auto задает их отображение лишь в том случае, если содержимое не помещается во фрейме; это поведение по умолчанию. Значение yes задает отображение полос прокрутки в любом случае, даже если они и не нужны, а значение no отключает их совсем. Используйте последнее значение с осторожностью: кто знает, какими увидит пользователь ваши страницы и увидит ли их вообще.

Атрибут NORESIZE отключает возможность изменения размеров фрейма.

Атрибуты FRAMEBORDER и BORDERCOLOR работают аналогично одноименным атрибутам тега <FRAMESET>.

Новые возможности гиперссылок

Здесь мы рассмотрим атрибут тега <A>, специально предназначенный для поддержки фреймов. Это атрибут TARGET.

```
TARGET="{Имя фрейма}|_self|_parent|_top|_blank"
```

Как видите, значением этого атрибута может быть как имя фрейма, указанное в атрибуте NAME тега <FRAME>, так и какое-либо из зарезервированных значений. С первым все просто: Web-страница, адрес которой указан в атрибуте HREF гиперссылки, отображается в указанном фрейме. А все зарезервированные значения рассмотрены в табл. 1.7.

Таблица 1.7. Зарезервированные значения атрибута TARGET

Значение	Описание
_self	Страница отображается в том фрейме, где находится гиперссылка
_parent	Страница отображается в окне, являющемся *родительским* по отношению к текущему фрейму. В нашем случае — в окне Web-обозревателя, затерев набор фреймов
_top	Страница отображается в окне, находящемся на верхнем уровне иерархии окон и фреймов. В нашем случае — в окне Web-обозревателя, затерев набор фреймов, как и в предыдущем случае
_blank	Страница отображается в новом окне Web-обозревателя

Сводим все воедино

Давайте еще раз рассмотрим наш набор фреймов. Только сначала произведем небольшое переформатирование: выделим все вложенности отступами. Я советую вам в дальнейшем поступать именно так: это помогает "отловить" досаднейшие ошибки вложенности.

```
<HTML>
<HEAD>
<TITLE>Сайт с фреймами</TITLE>
</HEAD>
<FRAMESET ROWS="50,*">
  <FRAME SRC="1.7.header.htm" NAME="frmHeader">
  <FRAMESET COLS="100,*">
    <FRAME SRC="1.7.contents.htm" NAME="frmContents">
    <FRAME SRC="1.7.1.htm" NAME="frmBody">
  </FRAMESET>
</FRAMESET>
</HTML>
```

Мы видим два вложенных набора фрейма. Один — *внешний* или *первого уровня* — делит страницу на две строки. Внутри него определяются фрейм с заголовком сайта и набор фреймов *второго уровня* (или *внутренний*). Внутренний набор фреймов в свою очередь делит вторую строку внешнего фрейма на два столбца и определяет внутри себя фреймы оглавления и содержимого сайта.

Набор фреймов, определенный нами, очень примитивен и коряв. Давайте немного его облагородим. Прежде всего изменим вложенность наборов фреймов так, чтобы заголовок сайт не охватывал всю ширину окна Web-обозревателя, а располагался только над содержимым. Далее, уберем эти

неимоверно широкие границы и отключим прокрутку там, где она не нужна. И под конец, изменим цвета некоторых страниц.

Это — наша новая страница набора фреймов. Назовем ее 1.8.htm.

```
<HTML>
<HEAD>
<TITLE>Новый сайт с фреймами</TITLE>
</HEAD>
<FRAMESET COLS="130,*" BORDER="0" FRAMEBORDER="0">
<FRAME SRC="1.8.contents.htm" NORESIZE>
<FRAMESET ROWS="50,*" BORDER="0" FRAMEBORDER="0">
<FRAME SRC="1.8.header.htm" SCROLLING="no" NORESIZE >
<FRAME SRC="1.7.1.htm" NAME="frmBody" NORESIZE >
</FRAMESET>
</FRAMESET>
</HTML>
```

Заметьте, что мы, кроме всего прочего, убрали имена у фреймов заголовка и оглавления. Они здесь не нужны: ведь мы не ссылаемся на них ни в одной гиперссылке, а их содержимое остается постоянным. Конечно, можно было бы их оставить, но лучше удалить из Web-страниц весь бесполезный текст. Желательно даже удалить из страниц, предназначенных для помещения в Сеть, все комментарии, оставив их в изначальных файлах. Помните, что очень и очень многие пользователи Интернета подключаются к всемирному информационному богатству по медленным телефонным линиям. И чем меньше мусора в ваших Web-страницах, тем меньше им придется качать из Сети.

Это — новая страница заголовка. Здесь мы изменили цвета фона страницы и текста, выравнивание, а также убрали заголовок — он все равно не нужен. Назовем ее 1.8.header.htm.

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="beige" TEXT="teal">
<H1><CENTER>Сайт с фреймами</CENTER></H1>
</BODY>
</HTML>
```

А это — новая страница оглавления. Сохраним ее под именем 1.8.contents.htm.

```
<HTML>
<HEAD>
<TITLE></TITLE>
```

```
</HEAD>
<BODY BGCOLOR="beige" TEXT="darkviolet" LINK="darkviolet
VLINK="darkviolet">
<CENTER>
<P><A HREF="1.7.1.htm" TARGET="frmBody">Страница 1</A></P>
<P><A HREF="1.7.2.htm" TARGET="frmBody">Страница 2</A></P>
<P><A HREF="1.7.3.htm" TARGET="frmBody">Страница 3</A></P>
</CENTER>
</BODY>
</HTML>
```

Здесь мы опять изменили цвета страницы. Заметьте, что мы установили тот же цвет для фона, что и в странице заголовка. Наш сайт не должен выглядеть аляповатой мазней на заборе. Единственно — мы немного выделили цветом заголовок.

Теперь откройте файл 1.8.htm в Web-обозревателе и посмотрите на результат (рис. 1.8).

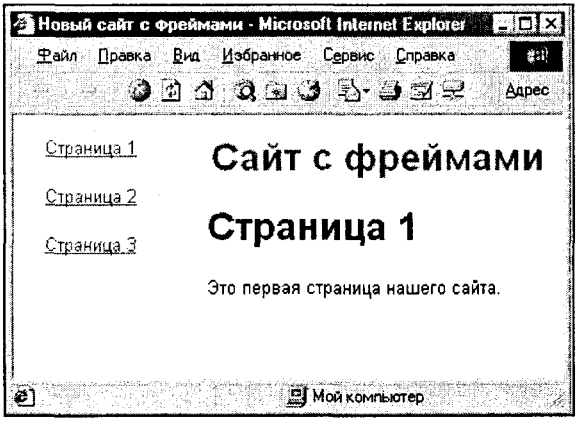


Рис. 1.8. Новый вид набора фреймов

По-моему, получилось заметно лучше, чем раньше. А по-вашему?

Теги <NOFRAMES> и </NOFRAMES>

Фреймы появились в HTML не сразу. И до сих пор кое-где используют программы Web-обозревателей, не поддерживающие их. Такой Web-обозреватель вместо набора фрейма не покажет ничего или вывалит перед вами исходный код HTML, описывающий набор фреймов. И ничего с ним не сделаешь: ну не поддерживает он фреймов!

Для таких "неграмотных", "темных" программ и предусмотрен парный тег <NOFRAMES> и </NOFRAMES>. Внутри него располагается текст, который будет

выведен в окно Web-обозревателя, не поддерживающего фреймы. Знакомые с фреймами Web-обозреватели просто проигнорируют этот тег.

```
<HTML>
. . .
<FRAMESET COLS="130,*" BORDER="0" FRAMEBORDER="0">
. . .
</FRAMESET>
<NOFRAMES>
Ваш Web-обозреватель не поддерживает фреймы. Поэтому вам будет
доступна урезанная версия сайта
<A HREF="chapter_1>Глава 1</A>
. . .
</NOFRAMES>
</HTML>
```

"Плавающие" фреймы

"Плавающие" фреймы поддерживает только Internet Explorer.

Что такое "плавающий" фрейм? Да ничего особенного. Просто фрейм, который можно встроить в обычную Web-страницу с текстом; при этом в месте, где находится тег "плавающего" фрейма, Web-обозреватель отобразит небольшое окошко, где и покажет страницу, интернет-адрес которой указан в атрибуте SRC. Вставляются "плавающие" фреймы с помощью парного тега <IFRAME>...</IFRAME>.

```
<IFRAME [SRC="{Адрес страницы}"] [WIDTH="{Ширина фрейма}"]
[NAME="{Имя фрейма}"] [ALIGN="left|center|right"]
[FRAMEBORDER="1|0|yes|no"] [SCROLLING="auto|no"]
[VSPACE="{Расстояние до текста страницы по вертикали}"]
[HSPACE="{Расстояние до текста страницы по горизонтали}"]
[BORDER="{Толщина границы}"]>
<!-- Текст, отображаемый Web-обозревателями, не поддерживающими
"плавающие" фреймы -->
</IFRAME>
```

Атрибут WIDTH задает ширину фрейма.

Атрибуты SRC и NAME аналогичны одноименным атрибутам тега <FRAME>.

Атрибут BORDER аналогичен одноименному атрибуту тега <FRAMESET>.

Атрибуты VSPACE и HSPACE аналогичны одноименным атрибутам тега <IMG>.

Атрибут ALIGN задает выравнивание "плавающего" фрейма. Доступны значения left, center и right, задающие выравнивание по левому краю, центру и правому краю соответственно.

Атрибут FRAMEBORDER позволяет установить или убрать границу у фрейма. Значения 1 и yes устанавливают границу, а 0 и no убирают, значение по умолчанию 1 (или yes). Чтобы толщина границы фрейма, заданная атрибутом BORDER, имела силу, атрибут FRAMEBORDER должен быть явно задан.

Атрибут SCROLLING позволяет отключить полосы прокрутки, появляющиеся, если содержимое фрейма выходит за его пределы. По умолчанию установлено значение auto, означающее, что полосы прокрутки появляются только когда нужно; чтобы отключить их совсем, дайте атрибуту значение no.

```
<IFRAME NAME="frmBanner" width="600" SRC="banner.htm">
Извините, но ваш Web-обозреватель не поддерживает &quot;плавающие&quot;
фреймы... Щелкните <A HREF="banner.htm" TARGET="_blank">здесь</A>,
чтобы просмотреть содержимое фрейма.
</IFRAME>
```

Таблицы

Изначально HTML не поддерживал таблицы ни в каком виде. Несчастные Web-дизайнеры пытались как-то выйти из положения, используя текст с заданным форматом и заключая его внутрь тегов <PRE> и </PRE>. Получалось, конечно, некрасиво, но что поделаешь. Теперь же они могут создавать какие угодно таблицы. Что и делают.

Сейчас таблицы используются не только для представления данных, но и как элемент оформления Web-страниц. Благодаря таблицам можно очень точно позиционировать текстовые фрагменты и графику, создать элементы оформления, линейки и заливки, отступы и выступы почти как в печатных изданиях.

Но хватит дразнить вас рассказом о мощных средствах современного HTML. Давайте возьмемся за работу! И создадим Web-страницу с таблицей. Пока что достаточно простую.

Самые простые таблицы

Наша первая таблица будет очень проста: всего несколько строк и столбцов. И будет это таблица умножения (заодно вспомним правила арифметики... шучу, конечно).

Назовем эту страничку 1.9.htm.

```
<HTML>
<HEAD>
<TITLE>Таблица умножения</TITLE>
</HEAD>
<BODY>
```

```
<H1>Таблица умножения (от 1 до 5)</H1>
<TABLE BORDER="2">
<TR><TD></TD><TD>1</TD><TD>2</TD><TD>3</TD><TD>4</TD><TD>5</TD></TR>
<TR><TD>1</TD><TD>1</TD><TD>2</TD><TD>3</TD><TD>4</TD><TD>5</TD></TR>
<TR><TD>2</TD><TD>2</TD><TD>4</TD><TD>6</TD><TD>8</TD><TD>10</TD></TR>
<TR><TD>3</TD><TD>3</TD><TD>6</TD><TD>9</TD><TD>12</TD><TD>15</TD></TR>
<TR><TD>4</TD><TD>4</TD><TD>8</TD><TD>12</TD><TD>16</TD><TD>20</TD></TR>
<TR><TD>5</TD><TD>5</TD><TD>10</TD><TD>15</TD><TD>20</TD><TD>25</TD></TR>
</TABLE>
</BODY>
</HTML>
```

Откройте ее в Web-обозревателе. Должно получиться, как на рис. 1.9.

Рис. 1.9. Таблица умножения

Вот такая у нас получилась таблица. Конечно, она не очень красива, но сейчас нам важно уяснить сам принцип. А раскрасить ее мы всегда успеем.

Простейшая таблица представляет собой множество вложенных друг в друга тегов, образующих сложную многоуровневую структуру. Сначала парой тегов <TABLE> и </TABLE> определяется сама таблица, потом внутрь вкладываются строки, определяемые парами тегов <TR> и </TR>. Внутрь каждой строки вкладываются ячейки, определяемые парами тегов <TD> и </TD>. И если вы внимательно посмотрите на приведенный выше листинг, то увидите все это.

Собственно, создать простую таблицу в HTML ничего не стоит. Единственная проблема, которая может при этом возникнуть, — вы устанете набирать все эти бесконечные вложенные теги. В самом деле, сравните объем текста листинга и размеры получившейся таблички. Но HTML-редакторы, работающие по принципу WYSIWYG, вам помогут.

Итак, разберемся, какими тегами форматируется таблица.

Тег таблицы

Тег таблицы имеет следующий формат:

```
<TABLE [BORDER="{Толщина рамки}"]
[CELLPADDING="{Расстояние между границей и содержимым ячейки}"]
[CELLSPACING="{Расстояние между ячейками}"]
[WIDTH="{Ширина таблицы}"] [HEIGHT="{Высота таблицы}"]
[ALIGN="left|center|right"] [BGCOLOR="{Цвет фона таблицы}"]
[BORDERCOLOR="{Цвет рамки}"] [BACKGROUND="{Адрес фонового рисунка}"]
[FRAME="none|above|below|hsides|lhs|rhs|vsides|vsides|box"]
[RULES="none|rows|cols|all"]>
<!-- Определения строк -->
</TABLE>
```

Теперь разберем все атрибуты этого тега.

Атрибут `BORDER` задает толщину рамки таблицы в пикселах. Если он отсутствует, рамка не видна, но Web-обозреватель при отображении таблицы считает, что она имеется. Если дать этому атрибуту значение `0`, рамки у таблицы не будет вообще.

Атрибут `CELLPADDING` позволяет задать размер в пикселах свободного пространства между рамкой и содержимым ячейки. По умолчанию оно равно 1.

Атрибут `CELLSPACING` позволяет задать размер в пикселах свободного пространства между ячейками таблицы. По умолчанию оно равно 2.

Атрибуты `WIDTH` и `HEIGHT` позволяют установить размеры таблицы. Допускается указывать размер и в виде абсолютной величины, в пикселах, и в процентах относительно размеров окна Web-обозревателя (в последнем случае не забудьте добавить к числу знак процента "%"). Если эти атрибуты пропущены, Web-обозреватель сам задает размеры таблицы так, чтобы все содержимое поместилось в ней.

Атрибут `ALIGN` задает "обтекание" таблицы текстом. Значение `left` заставляет таблицу прижаться к левой границе окна Web-обозревателя, а текст будет "обтекать" ее справа, значение `right` задает обратное расположение, а значение `center` размещает таблицу в центре страницы.

Атрибуты `BGCOLOR` и `BACKGROUND` задают соответственно цвет фона и адрес фонового рисунка для таблицы.

Атрибут `BORDERCOLOR` позволяет задать цвет рамки таблицы. Он не стандартизирован W3C и поддерживается только Internet Explorer.

Атрибут `FRAME` определяет, какие части внешней рамки таблицы будут рисоваться. Доступны следующие значения:

- `void` — внешней рамки нет вообще (значение по умолчанию);
- `above` — рисуется только верхняя линия внешней рамки;

- `below` — рисуется только нижняя линия внешней рамки;
- `hsides` — рисуются только горизонтальные линии внешней рамки, т. е. верхняя и нижняя;
- `lhs` — рисуется только левая линия внешней рамки;
- `rhs` — рисуется только правая линия внешней рамки;
- `vsides` — рисуются только вертикальные линии внешней рамки, т. е. левая и правая;
- `box` — рисуются все линии внешней рамки.

Этот атрибут поддерживается только Internet Explorer, хотя и стандартизирован W3C.

Атрибут `RULES` задает, как будут рисоваться внутренние рамки таблицы. Здесь доступны следующие значения:

- `none` — никаких внутренних рамок;
- `rows` — рисуются только горизонтальные линии (между строками);
- `cols` — рисуются только вертикальные линии (между столбцами);
- `all` — рисуются все внутренние рамки.

Этот атрибут поддерживается только Internet Explorer, хотя также стандартизирован W3C.

Тег строки

Строки определяются внутри тега таблицы с помощью пар тегов `<TR>` и `</TR>`.

```
<TR [ALIGN="left|center|right|justify"]
[VALIGN="left|middle|bottom|baseline"] [BGCOLOR="{Цвет фона строки}"]
[BORDERCOLOR="{Цвет рамки}"] [WIDTH="{Ширина ячейки}"]
[HEIGHT="{Высота ячейки}"]>
<!-- Определения ячеек -->
</TR>
```

Атрибут `ALIGN` задает горизонтальное выравнивание текста в ячейках таблицы. Доступные значения `left`, `center`, `right` и `justify` устанавливают выравнивание по левому краю, центру, правому краю ячейки и полное выравнивание по ширине соответственно. По умолчанию текст выравнивается по левому краю.

Атрибут `VALIGN` задает вертикальное выравнивание текста в ячейках таблицы. Доступные значения `top`, `middle`, `bottom` и `baseline` устанавливают выравни-

вание по верхнему краю, центру, нижнему краю ячейки и базовой линии соответственно. По умолчанию текст выравнивается по центру.

Атрибуты BGCOLOR, BORDERCOLOR, HEIGHT и WIDTH ведут себя так же и выполняют те же функции, что и одноименные атрибуты тега <TABLE>. Они задают цвет фона, цвет рамки, высоту и ширину строки таблицы соответственно.

Тег ячейки

Ячейки таблицы определяются внутри тега строки с помощью пар тегов <TD> и </TD>. Как раз в ячейках и находится полезное содержимое таблицы.

```
<TD [ALIGN="left|center|right|justify"]
[VALIGN="left|middle|bottom|baseline"] [WIDTH="{Ширина ячейки}"]
[HEIGHT="{Высота ячейки}"] [NOWRAP]
[COLSPAN="{Количество объединяемых ячеек по вертикали}"]
[ROWSPAN="{Количество объединяемых ячеек по горизонтали }"]
[BGCOLOR="{Цвет фона ячейки}"] [BORDERCOLOR="{Цвет рамки}"]
[BACKGROUND="{Адрес фонового рисунка}"]>
<!-- Содержимое ячейки -->
</TD>
```

Атрибуты ALIGN и VALIGN задают горизонтальное и вертикальное выравнивание текста в ячейке соответственно.

Атрибуты WIDTH и HEIGHT позволяют установить ширину и высоту ячейки. Допускается указывать размер и в виде абсолютной величины, в пикселах, и в процентах относительно размеров таблицы (атрибут WIDTH) или окна Web-обозревателя (атрибут HEIGHT). (Не забудьте добавить к числу знак процента "%", если задаете относительный размер.) Если эти атрибуты пропущены, Web-обозреватель сам задает размеры ячейки так, чтобы все содержимое поместилось в ней.

Атрибут NOWRAP отменяет перенос текста со строки на строку. Он не "свертывается" в несколько строк для того, чтобы полностью вместиться в ячейку, а "выстраивается" в виде одной длинной строки. Будьте осторожны с этим атрибутом, т. к. при его использовании ваша таблица может сильно "раздуться" в ширину и не поместиться в окне Web-обозревателя; пользователю придется прокручивать вашу таблицу по горизонтали, и будьте уверены — он перемоет вам все косточки.

Атрибуты ROWSPAN и COLSPAN мы рассмотрим чуть позже, когда будем говорить о сложных таблицах с объединенными ячейками. А пока наберитесь терпения.

Атрибуты BGCOLOR, BACKGROUND и BORDERCOLOR уже должны быть вам знакомы. На этот раз они управляют параметрами индивидуальной ячейки таблицы.

Таблица в целом

Итак, мы рассмотрели все теги, с помощью которых создаются таблицы. Давайте теперь перейдем к общим вопросам построения таблиц.

Мы знаем, что таблица целиком описывается парой тегов <TABLE> и </TABLE>, внутри которых помещаются описания строк. Отдельная строка описывается опять же парой тегов <TR> и </TR>, внутри которых помещаются описания ячеек. Каждая ячейка описывается парой тегов <TD> и </TD>, внутри которых помещается собственно содержимое ячейки. Уф! А что за содержимое? Что можно "положить" в таблицу?

Все, что угодно. Текст, в том числе и сложно отформатированный, рисунки и даже другие таблицы. Все, кроме фреймов.

Давайте рассмотрим пример таблицы. Сохраним ее под именем 1.10.htm.

```
<HTML>
<HEAD>
<TITLE>Хитрая таблица</TITLE>
</HEAD>
<BODY>
<H1>Хитрая таблица</H1>
<!-- Это наша таблица. Следите за тегами -->
<TABLE BORDER="2" WIDTH="100%">
<TR ALIGN="center">
<TD><P>Просто текст</P></TD>
<!-- Здесь у нас рисунок -->
<TD><IMG SRC="Tips.gif"></TD>
</TR>
<TR ALIGN="center">
<!-- Сложно отформатированный текст -->
<TD>
<P>Строка <B>1</B></P>
<P>Строка <B>2</B></P>
<P>Строка <B>3</B></P>
</TD>
<!-- Здесь у нас вложена ДРУГАЯ ТАБЛИЦА!!! -->
<TD>
  <TABLE BORDER="1">
  <TR ALIGN="center">
  <TD>1</TD>
  <TD>2</TD>
  </TR>
  <TR ALIGN="center">
  <TD>3</TD>
```

```
  <TD>4</TD>
  </TR>
  </TABLE>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

Результаты можно увидеть на рис. 1.10.

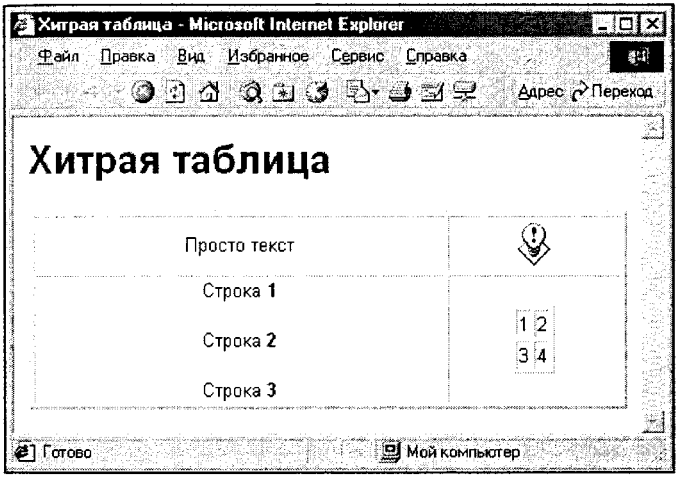


Рис. 1.10. Хитрая таблица

Как видите, таблицы HTML — вещь исключительно мощная. Неудивительно, что они пользуются такой популярностью у Web-дизайнеров. Я уже говорил, что таблицы позволяют отформатировать Web-страницу так, что она будет выглядеть как документ, созданный в настольной издательской программе. Ну, или почти так же...

Теперь поговорим вот о чем. Простые таблицы (как, например, табл. 1.8) создаются очень просто:

Таблица 1.8. *Пример простой таблицы*

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Если вы хотите, можете уже прямо сейчас создать такую таблицу в HTML. За образец можете взять нашу таблицу умножения.

Но иногда нужно получить более сложную таблицу (табл. 1.9).

Таблица 1.9. *Пример таблицы с объединениями ячеек*

1	2	3	
	4	5	6
	7	8	9
	10	11	12

Как видите, все ячейки первого столбца таблицы объединяются в одну (номер 1), как и две последние ячейки первой строки (номер 3). Как сделать так в HTML?

Очень просто. Для этого служат атрибуты COLSPAN и ROWSPAN тега <TD>. Они задают количество объединяемых ячеек по горизонтали и вертикали соответственно.

Рассмотрим HTML-код табл. 1.8:

```
<TABLE>
<TR><TD>1</TD><TD>2</TD><TD>3</TD><TD>4</TD></TR>
<TR><TD>5</TD><TD>6</TD><TD>7</TD><TD>8</TD></TR>
<TR><TD>9</TD><TD>10</TD><TD>11</TD><TD>12</TD></TR>
<TR><TD>13</TD><TD>14</TD><TD>15</TD><TD>16</TD></TR>
</TABLE>
```

Преобразуем ее к виду табл. 1.9. Сначала "сделаем" первый столбец (большая ячейка номер 1).

```
<TABLE>
<!-- Четыре ячейки по вертикали объединяются в одну для получения -->
<!-- большой ячейки номер 1 -->
<!-- Атрибут ROWSPAN выделен -->
<TR><TD ROWSPAN="4">1</TD><TD>2</TD><TD>3</TD><TD>4</TD></TR>
<!-- Заметьте, что далее в каждой строке мы определяем только -->
<!-- три ячейки. Четвертая нами уже задана в предыдущей строке — -->
<!-- это большая ячейка номер 1; она и так есть во всех -->
<!-- последующих строках, так что определять ее не надо -->
<TR><TD>5</TD><TD>6</TD><TD>7</TD></TR>
<TR><TD>8</TD><TD>9</TD><TD>10</TD></TR>
<TR><TD>11</TD><TD>12</TD><TD>13</TD></TR>
</TABLE>
```

Теперь закончим преобразование таблицы, создав ячейку номер 3:

```
<TABLE>
<!-- Две ячейки по горизонтали объединяются в одну -->
<!-- Атрибут COLSPAN выделен -->
```

```
<TR><TD ROWSPAN="4">1</TD><TD>2</TD><TD COLSPAN="2">3</TD></TR>
<!-- Далее все как обычно -->
<TR><TD>4</TD><TD>5</TD><TD>6</TD></TR>
<TR><TD>7</TD><TD>8</TD><TD>9</TD></TR>
<TR><TD>10</TD><TD>11</TD><TD>12</TD></TR>
</TABLE>
```

Вы можете сохранить приведенный выше код в HTML-файле и просмотреть его в Web-обозревателе. Таблица, которую вы увидите, будет выглядеть так же, как и табл. 1.9.

Единственное предупреждение. Работая со сложными таблицами, содержащими объединенные ячейки, очень легко допустить ошибку, "потеряв" ячейку или "приделав" таблице лишнюю. Всегда тщательно пересчитывайте все ячейки таблицы, сверяйте их количество с количеством строк и учитывайте все слияния.

Как еще можно использовать таблицы

Я уже не раз говорил, что таблицы можно использовать для создания Web-страниц, не уступающих по качеству оформления документам, созданным в издательских программах. Давайте рассмотрим пример, подтверждающий справедливость сказанного.

Какой главный недостаток HTML? Его ориентированность на текст, на простые текстовые документы. Вы не можете сверстать документ в несколько колонок, сделать сноски, врезки и другие подобные типографские штучки, которые поддерживают все современные текстовые процессоры. Допускается только набирать текст и форматировать его, т. е. создавать простой текст, заполняющий Web-страницу от начала до конца, либо текст, слегка разбавленный рисунками и таблицами.

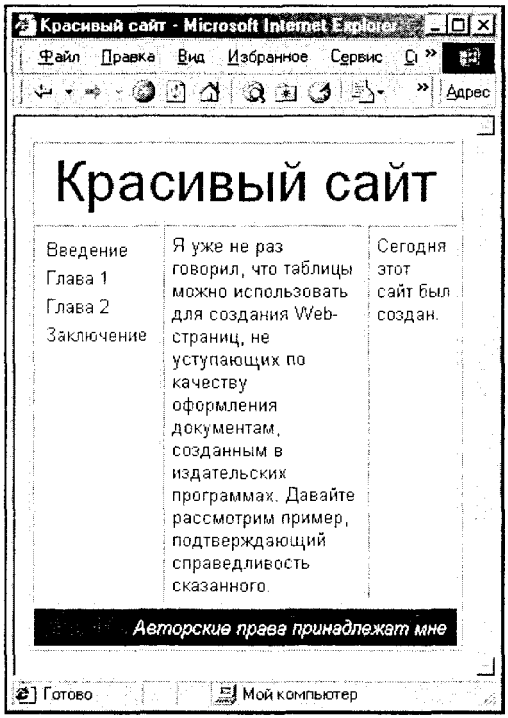
Бывалый верстальщик с ума сойдет от такой бедности.

Но Web-дизайнеры — народ очень изобретательный. Они придумали фреймы, чтобы разбить Web-страницу на независимые, но вместе с тем логически связанные части: заголовок, оглавление и собственно содержание. Их не останавливает то, что фреймы — один из самых "несовместимых" элементов HTML. А еще они придумали, как использовать таблицы для оформления Web-страниц. И привнести в скуку "чистого" HTML, придуманного учеными занудами для своих нужд, элемент фантазии. Вялотекущий по страницам текст сменили шедевры Web-дизайна, которые вы теперь можете видеть на самых разных Web-сайтах.

Давайте и мы займемся высоким HTML-творчеством.

Помните, в разделе, посвященном фреймам, мы рассматривали пример целого сайта. Перепишем его с использованием таблиц. И попутно добавим ему немного красивостей. Сохраним результат в файле 1.11.htm.

```
<HTML>
<HEAD>
<TITLE>Красивый сайт</TITLE>
</HEAD>
<BODY>
<!-- Весь наш документ будет представлять собой большую таблицу -->
<TABLE BORDER="1" CELLPADDING="5">
<!-- Это заголовок нашего сайта. Он займет всю строку -->
<TR>
<TD COLSPAN="3" BGCOLOR="bisque" NOWRAP>
<P ALIGN="center"><FONT SIZE="7" COLOR="midnightblue">Красивый
сайт</FONT></P>
</TD>
</TR>
<!-- А это — содержимое -->
<TR VALIGN="top">
<!-- Столбец ссылок содержит вложенную таблицу. Здесь я указал -->
<!-- ссылку на несуществующую страницу -->
<TD BGCOLOR="bisque">
  <TABLE>
  <TR><TD><P><A HREF="dummy.htm">Введение</A></P></TD></TR>
  <TR><TD><P><A HREF="dummy.htm">Глава 1</A></P></TD></TR>
  <TR><TD><P><A HREF="dummy.htm">Глава 2</A></P></TD></TR>
  <TR><TD><P><A HREF="dummy.htm">Заключение</A></P></TD></TR>
  </TABLE>
</TD>
<!-- Столбец описания сайта -->
<TD>
<!-- Какой-либо текст. Я сам скопировал сюда текст из этой книги -->
</TD>
<!-- Столбец новостей сайта -->
<TD>
<P>Сегодня этот сайт был создан.</P>
</TD>
</TR>
<!-- Сведения об авторских правах -->
<TR>
<TD COLSPAN="3" BGCOLOR="black">
<P ALIGN="right"><I><FONT COLOR="white">Авторские права принадлежат
мне</FONT></I></P>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```



Получившийся результат можно увидеть на рис. 1.11.

Рис. 1.11. Сайт, в оформлении которого использованы таблицы

Теперь таблицы используют в создании Web-страниц так же часто, как и фреймы. Более того, если фреймы стали уже чем-то банальным, не достойным стильного Web-дизайнера, то таблицы в большой моде. Они более "совместимы" с различными Web-обозревателями и порождают меньше неприятных сюрпризов при просмотре, чем фреймы. Они более гибки. Единственный их недостаток: очень долгая загрузка, ведь Web-обозреватель не может отобразить таблицу, пока не загрузит ее целиком. А если таблица велика, то...

Рассмотренный нами дизайн страницы — строка заголовка, колонка ссылок, колонка описания сайта, колонка новостей и строка сведений об авторских правах — очень популярен. Уже сейчас таких сайтов становится все больше и больше; кроме того, подобный дизайн завоевывает популярность и среди любителей. Согласитесь: такие страницы легко читать. А с долгой загрузкой можно и смириться.

Однако здесь кроется еще одна проблема, но уже не для пользователей. Как правило, каждая страница сайта в этом случае должна содержать одинаковые строки заголовка и авторских прав и столбец ссылок, т. е. все страницы будут содержать некоторую одинаковую информацию. И если, скажем, заголовок сайта изменится, то замена его на всех страницах сайта становится весьма трудоемкой задачей. Но уже существует достаточно много программ, выполняющих эту задачу быстрее живого Web-дизайнера.

Заголовок и шапка таблицы

Когда Web-дизайнер форматирует таблицу, используя теги <TR> и <TD> (и парные им закрывающие теги), Web-обозреватель не знает, что будет помещаться в ячейках таблицы: может, это будет шапка таблицы, может, содержимое, а может, они останутся пустыми. Несмотря на то, что Web-дизайнер поместит в ячейки таблицы, все они будут одинаковыми, без каких-либо отличий в форматировании. Поэтому Web-дизайнер сам должен соответствующим образом отформатировать текст в соответствующих ячейках, основываясь на его предназначении.

Неудивительно, что были предложены специальные теги, позволяющие как-то выделить ячейки, составляющие шапку таблицы. И заодно таблицы получили заголовок.

Тег ячейки заголовка практически не отличается от тега простой ячейки. Его формат приведен ниже:

```
<TH [ALIGN="left|center|right|justify"]
[VALIGN="left|middle|bottom|baseline"] [WIDTH="{Ширина ячейки}"]
[HEIGHT="{Высота ячейки}"] [NOWRAP]
[COLSPAN="{Количество объединяемых ячеек по вертикали}"]
[ROWSPAN="{Количество объединяемых ячеек по горизонтали }"]
[BGCOLOR="{Цвет фона ячейки}"] [BORDERCOLOR="{Цвет рамки}"]
[BACKGROUND="{Адрес фонового рисунка}"]>
<!-- Содержимое ячейки -->
</TH>
```

Как видите, он совсем не отличается от формата тега <TD>. И это правильно, ведь ячейка заголовка не должна, по идее, иметь каких-либо отличий от простых ячеек, кроме особого визуального представления: по умолчанию текст в ячейках заголовка выровнен по центру и выделен жирным шрифтом.

Заголовок таблицы устанавливают с помощью пары тегов <CAPTION> и </CAPTION>. Он появляется внутри рамки таблицы, но вне всех ее строк и ячеек.

```
<CAPTION [ALIGN="top|bottom|left|center|right"] [VALIGN="top|bottom"]>
. . .
</CAPTION>
```

Атрибут ALIGN ведет себя по-разному в зависимости от Web-обозревателя. Для Navigator он задает местонахождение заголовка: значение top выводит заголовок вверху таблицы (это поведение по умолчанию), а bottom — внизу. Для Internet Explorer он задает выравнивание текста в заголовке: по левому краю (значение left), по центру (значение center, это же поведение по умолчанию) или по правому краю (значение right).

Атрибут VALIGN поддерживается только Internet Explorer и задает местонахождение заголовка таблицы. Значение top помещает его в верхней части таблицы (это поведение по умолчанию), а значение bottom — в нижней.

Логическое форматирование таблиц

Итак, шапку и заголовок таблицы мы выделили. Следующий шаг: разделить таблицу на различные логические части сообразно их функциональному назначению. Хотелось бы разделить шапку таблицы (в которой были бы выделены отдельные колонки), тело таблицы (содержащее полезные данные) и основание (здесь могут быть какие-то примечания или строка итогов).

Спецификацией HTML определены соответствующие теги. Но пока что их поддерживает только Internet Explorer, поэтому все Web-дизайнеры используют только обычное, нелогическое форматирование таблиц с помощью тегов, описанных мной выше. И если вы будете применять в своих таблицах логическое разбиение, то правильно их отобразит только Internet Explorer (хотя что-то показать смогут все Web-обозреватели).

За логическое форматирование таблиц отвечают пять тегов, из которых три — парные.

Тег <THEAD> выделяет заголовок и шапку таблицы, образуя так называемую *секцию заголовка*.

```
<THEAD [ALIGN="left|center|right|justify"]
[VALIGN="left|middle|bottom|baseline"] [BGCOLOR="{Цвет фона секции}"]>
<!-- Определения строк и ячеек -->
</THEAD>
```

Все атрибуты этого тега ведут себя так же, как и во всех предыдущих случаях. Я не буду подробно на них останавливаться.

Тег <TBODY> образует *секцию тела* таблицы.

```
<TBODY [ALIGN="left|center|right|justify"]
[VALIGN="left|middle|bottom|baseline"] [BGCOLOR="{Цвет фона секции}"]>
<!-- Определения строк и ячеек -->
</TBODY>
```

Тег <TFOOT> отвечает за *секцию основания* таблицы.

```
<TFOOT [ALIGN="left|center|right|justify"]
[VALIGN="left|middle|bottom|baseline"] [BGCOLOR="{Цвет фона секции}"]>
<!-- Определения строк и ячеек -->
</TFOOT>
```

Теперь самое время рассмотреть какой-либо пример. Пусть это будет наша таблица умножения. Чтобы сэкономить место, я удалил заголовок HTML-файла и служебные теги.

```
<TABLE BORDER="2">
<THEAD>
<TR><TH></TH><TH>1</TH><TH>2</TH><TH>3</TH><TH>4</TH><TH>5</TH></TR>
</THEAD>
</TBODY>
<TR><TD>1</TD><TD>1</TD><TD>2</TD><TD>3</TD><TD>4</TD><TD>5</TD></TR>
<TR><TD>2</TD><TD>2</TD><TD>4</TD><TD>6</TD><TD>8</TD><TD>10</TD></TR>
<TR><TD>3</TD><TD>3</TD><TD>6</TD><TD>9</TD><TD>12</TD><TD>15</TD></TR>
<TR><TD>4</TD><TD>4</TD><TD>8</TD><TD>12</TD><TD>16</TD><TD>20</TD></TR>
<TR><TD>5</TD><TD>5</TD><TD>10</TD><TD>15</TD><TD>20</TD><TD>25</TD></TR>
</TBODY>
<TFOOT ALIGN="right">
<TR><TD COLSPAN="6"><I>Вот такая таблица умножения</I></TD></TR>
</TFOOT>
</TABLE>
```

Я добавил кое-какое дополнительное форматирование, но это не принципиально. Главное — *уяснить принцип*.

Как работают эти теги и что они дают? Визуально — ничего. Они просто логически разделяют таблицу на три части. Это может быть полезно во многих случаях, особенно если вы планируете какую-то дополнительную обработку таблицы. И, конечно, эти теги просто необходимы, если вы будете писать какие-либо скриптовые программы для своих Web-страниц.

Вы уже заметили, что теги логического форматирования таблиц можно использовать для задания каких-то общих атрибутов групп строк, например выравнивания и цвета фона. К сожалению, поймет все это правильно только Internet Explorer.

Логическое форматирование колонок

Следующим шагом стало введение специальных тегов логического форматирования для отдельных столбцов таблицы. Визуально эти теги ничего не дают, но помогут задать для целых групп столбцов общие параметры форматирования. Таких тегов два (причем, непарных): `<COLGROUP>` для группы столбцов и `<COL>` для отдельного столбца.

Тег `<COLGROUP>` объединяет несколько столбцов в группу.

```
<COLGROUP [ALIGN="left|center|right|justify"]
[VALIGN="left|middle|bottom|baseline"]
[SPAN="{Количество столбцов в группе}"]>
```

Атрибут `ALIGN` и `VALIGN` мы описывать не будем: они делают то же самое, что и в рассмотренных ранее тегах.

Атрибут `SPAN` задает количество столбцов в группе. Вы уже заметили, что тег `<COLGROUP>` — одинарный, поэтому объединять какое-то количество столбцов внутри себя, как парные теги, он не может. Количество входящих в группу

столбцов нужно задавать другим способом — через специальный атрибут. По умолчанию значение атрибута SPAN равно единице, т. е. группа будет содержать один столбец.

И еще один тег — <COL>, задающий формат сразу нескольких колонок, но не объединяющий их в группу:

```
<COL [ALIGN="left|center|right|justify"]
[VALIGN="left|middle|bottom|baseline"]
[SPAN="{Количество столбцов, на которое распространяется
↳форматирование}"]>
```

Здесь все атрибуты такие же, как у тега <COLGROUP>. За тем исключением, что атрибут SPAN задает не количество столбцов в группе, а количество столбцов, на которые будет распространяться указанное в теге <COL> форматирование. То есть тег <COL> не группирует столбцы в группу, а указывает, сколько столбцов будут иметь заданные в нем параметры форматирования текста.

Давайте рассмотрим пример заголовка некой таблицы.

```
<COLGROUP SPAN="3" ALIGN="left">
<COLGROUP SPAN="2" ALIGN="right">
<COL SPAN="2" ALIGN="justify">
<THEAD>
<TR>
<TD>Столбец 1</TD>
<TD>Столбец 2</TD>
<TD>Столбец 3</TD>
<TD>Столбец 4</TD>
<TD>Столбец 5</TD>
<TD>Столбец 6</TD>
<TD>Столбец 7</TD>
</TR>
</THEAD>
```

Эта таблица имеет пять столбцов, разбитых на две группы, и еще два дополнительных столбца, не входящих ни в одну группу. Первая группа охватит столбцы с первого по третий (атрибут SPAN равен 3), а вторая соответственно — четвертый и пятый (атрибут SPAN равен 2). Первая группа столбцов будет выровнена влево (вообще-то, атрибут ALIGN можно было бы и не писать: по умолчанию текст в ячейках выравнивается влево), а вторая — вправо. А оставшиеся два столбца останутся "сами по себе", и текст в них будет выровнен по обоим краям.

Каскадные таблицы стилей

Каскадные таблицы стилей или *CSS* (Cascading Style Sheets) были второй революцией, потрясшей WWW. (Первой, как вы помните, стало появление фреймов и таблиц, давших Web-документам второе измерение.) Если до

этой поры Web-дизайнер не знал, как будет выглядеть его творение в разных программах Web-обозревателей, то теперь он может контролировать все: от начертания шрифта до положения картинки на странице. Таблицы стилей еще сильнее приблизили Web-документы к их родственникам, документам, порожденным издательскими пакетами. И, самое главное, они отделили представление (внешний вид) Web-документов от их содержания.

Что это значит? Сейчас объясню.

Понятие таблиц стилей

Предположим, вам нужно изменить цвет текста в HTML-документе с черного на синий. Вы помещаете его в пару тегов `<FONT>` и `</FONT>` следующего вида:

```
<P><FONT COLOR="blue">Это синий текст</FONT></P>
```

Заметьте, что здесь мы смешиваем теги, ответственные за логические форматирование (`<P>`) и внешнее представление (`<FONT>`) информации.

А теперь представим, что вы вынесли определение внешнего вида текста в другое место документа. И получили, скажем, нечто подобное:

```
P.bluetext { color: blue }
```

Эта строка означает, что мы определили для текста, находящегося внутри тега `<P>` и помеченного *стилевым классом* `bluetext`, синий цвет шрифта. Такая конструкция HTML называется *определением стиля* или просто *стилем*.

В результате в HTML-тексте у нас останутся только теги логического форматирования текста.

```
<P CLASS="bluetext">Это синий текст</P>
```

Здесь мы пометили нужный текст с помощью нового атрибута `CLASS`, присвоив ему значение `bluetext`. Атрибут `CLASS` задает имя стилевого класса для тега. И — внимание — его поддерживают все теги!

Вы можете переназначить цвет текста для всех тегов `<P>`. (Заметьте, что в этом случае мы не задаем имя стилевого класса.)

```
P { color: blue }
```

Или вы можете задать форматирование для стилевого класса, не привязанного ни к какому тегу:

```
.bluetext { color: blue }
```

И теперь вы можете присваивать стилевой класс тексту, заключенному в любые теги:

```
<H1 CLASS="bluetext">Это синий текст</H1>
<CENTER CLASS="bluetext">Это синий текст</CENTER>
Это <B CLASS="bluetext">жирный синий</B> текст.
```

Вы можете дать специальное форматирование тегу только в том случае, если он заключен внутрь другого тега:

```
H7 B { color: blue }
```

И теперь...

```
<H7><B>Этот</B> текст будет синим</H7>
<P>А <B>этот</B> — не будет!</P>
```

Более того, вы можете встроить определение стиля прямо в тег. Такие определения стилей называются *встроенными в теги.*

```
<P STYLE="color: blue">Это синий текст</P>
```

Это достигается с помощью атрибута `STYLE`, который также поддерживают все теги HTML.

И еще один способ привязать стиль к какому-либо тегу — использовать атрибут `ID`, задающий уникальное имя элемента HTML.

```
#headerofdocument { font-size: 20pt }
```

Здесь мы задали размер шрифта 20 пунктов.

```
<H1 ID="headerofdocument">Это заголовок документа</H1>
```

Вы можете задать несколько *атрибутов* в определении стиля. (Внимание: не путайте совершенно разные вещи — атрибуты стиля и атрибуты тега.) В этом случае они разделяются точкой с запятой:

```
P { color: blue; font-size: 9pt; text-align: center }
```

Здесь мы задали синий цвет текста, размер шрифта 9 пунктов и выравнивание текста по центру. И все это будет применено к любому тексту, расположенному в теге `<P>`.

Определения стилей, вынесенные в заголовок HTML-документа, и составляют *таблицу стилей.* Таблица стилей заключается в теги `<STYLE>` и `</STYLE>`:

```
<STYLE [TYPE="text/css"]>
. . .
</STYLE>
```

Тег `<STYLE>` может содержать необязательный атрибут `TYPE`, содержащий обязательное значение `text/css`.

Таблицу стилей можно вынести в отдельный файл и использовать сразу в нескольких документах. В этом случае в заголовке HTML-документа необходимо разместить тег `<LINK>`, указывающий на эту таблицу стилей:

```
<LINK REL="stylesheet" HREF="{Адрес файла таблицы стилей}">
```

Здесь вы видите два обязательных атрибута: `REL`, содержащий значение `stylesheet`, и `HREF`, где задается адрес таблицы стилей. Вот и все!

Теперь давайте разберем большой и сложный пример Web-страницы, использующей таблицы стилей. Сначала рассмотрим таблицу стилей. Сохраните ее в файле 1.12.css.

```
<!-- Здесь мы задаем цвет фона страницы и шрифт по умолчанию.
Узнали тег BODY? -->
BODY { background-color: bisque; font-family: "Arial" }
P { font-size: 10pt }
H1 { color: teal; font-size: 16pt }
```

Теперь — очередь за Web-страницей. Она связана с только что определенной таблицей стилей, более того — определяет внутри себя еще одну таблицу стилей и даже использует стили, встроенные в теги. Сохраните страницу в файле 1.12.htm.

```
<HTML>
<HEAD>
<TITLE>Разные стили</TITLE>
<LINK REL="stylesheet" HREF="1.12.css">
<STYLE>
<!-- Задаем другой стиль для тега H1 -->
H1 { font-size: 14 pt; font-family: "Times New Roman" }
<!-- Доопределяем свойства тега P -->
P { font-style: italic }
H2 { font-size: 12 pt }
</STYLE>
</HEAD>
<BODY>
<H1>Разные стили</H1>
<P>Это Web-страница, созданная с использованием таблиц стилей.</P>
<H2>Предупреждение!</H2>
<P STYLE="font-size: 8pt">Это Web-страница, созданная с использованием
разных таблиц стилей.</P>
</BODY>
</HTML>
```

Результат показан на рис. 1.12.

Здесь мы подошли к понятию *иерархичности* или *каскадности* таблиц стилей. Давайте рассмотрим подробнее HTML-код нашей страницы и связанной с ней таблицы стилей.

В таблице стилей 1.12.css мы установили для тега <BODY> шрифт "Arial". В таблице стилей, встроенной в Web-страницу, для тега <H1> мы изменили шрифт на "Times New Roman". И что же! Текст, отформатированный тегом <H1>, отобразился шрифтом "Times New Roman". Значит, установки встроенной таблицы стилей перекрыли установки связанной. Как говорят в народе, своя рубашка ближе к телу. Остальной же текст будет отображаться

с использованием шрифта "Arial", т. е. для него будет действовать *унаследованный* от родительского тега <BODY> стиль.

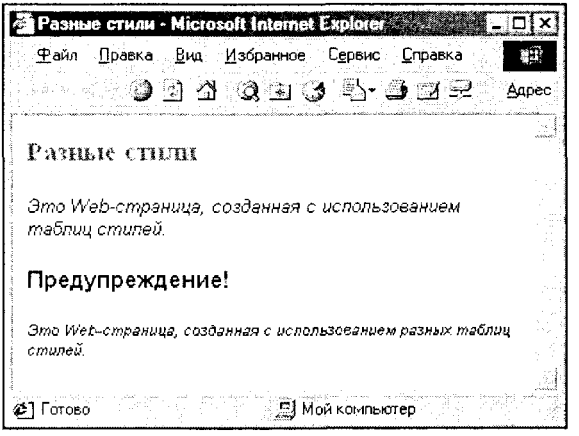


Рис. 1.12. Web-страница с использованием таблиц стилей

Далее, опять же в таблице стилей 1.12.css мы установили для тега <P> размер шрифта 10 пунктов. В таблице стилей, встроенной в Web-страницу, для этого тега мы задали курсивное начертание. Так как эти установки не отменяют друг друга, текст, отформатированный тегами <P>, будет отображаться курсивно и размером 10 пунктов. Но только не последний абзац! Для него мы задали встроенный стиль: размер шрифта 8 пунктов. Именно таким шрифтом он и отобразится, и это значит, что установки встроенного стиля перекрыли установки связанной таблицы стилей.

То есть, выстраивается как бы иерархия таблиц стилей, причем одни стили отменяют другие. Встроенные в тег стили отменяют установки встроенной в документ таблицы стилей, а та в свою очередь отменяет установки связанной таблицы стилей. В общем, те стили, что определены позже, отменяют определенные раньше. Или, как еще говорят, стили, определенные для родительских элементов, имеют более низкий приоритет, чем стили, определенные для дочерних. В частности, мы видели, как стиль, встроенный в тег <P> последнего абзаца, отменил стиль, заданный для тела документа (тега <BODY>).

Стили

Здесь мы приведем список некоторых атрибутов стиля, которые вы можете использовать в своих Web-страницах. Полное описание каскадных таблиц стилей см. в приложении 2.

Цвета

Ниже, в табл. 1.10 мы привели список некоторых атрибутов, описывающих цветовые характеристики различных тегов.

Таблица 1.10. Атрибуты определения цветов

Атрибут	Описание
`color: {Цвет};`	Цвет элемента (текста)
`background-color: transparent|{Цвет};`	Фоновый цвет элемента. Допускается либо цвет, либо значение `transparent`, делающее фон прозрачным
`background-image: none|{Адрес файла рисунка};`	Фоновый рисунок. Допускается либо адрес, либо значение `none`, отключающее фоновую заливку
`background-attachment: scroll|fixed;`	Значение `scroll` включает прокрутку фонового рисунка вместе с содержимым страницы, `fixed` отключает

Цвет может быть задан одним их пяти способов:

- именем цвета (`blue`, `white` или `bisque`);
- значением вида `#{R}{G}{B}`, где `R` — красная составляющая, `G` — зеленая, а `B` — синяя, причем все составляющие должны быть однозначными шестнадцатеричными числами. Пример: `#F00` — это красный цвет;
- значением, аналогичным предыдущему, но уже вида `#{RR}{GG}{BB}`, т. е. каждая составляющая имеет уже по два знака. Пример: `#00FF00` — зеленый цвет;
- значением `rgb({R},{G},{B})`, где каждая составляющая задается десятичным числом. Пример: `rgb(255,0,0)` — красный;
- аналогично предыдущему, но каждое отдельное значение выражает долю в общем цвете. Пример: `rgb(100%,0%,0%)` — опять же красный.

И еще. В отличие от тегов HTML, в атрибутах стилей, требующих интернет-адреса, его следует вводить в формате `url({Собственно адрес})`. Например, `url(/folder1/folder2/file.gif)`. Запомните — это очень важно!

Шрифты

Здесь приведены некоторые атрибуты определения шрифтовых параметров тегов.

Атрибут `font-family` задает начертание шрифта для текста.

```
font-family: "{Имя шрифта}|{Имя семейства шрифтов},...";
```

Имя шрифта задается в виде "Arial" или "Times New Roman". При задании этого атрибута необходимо, чтобы на компьютере клиента такие шрифты были.

Кроме имени конкретного шрифта можно задать имя одного из семейств шрифтов, представляющих целые наборы совместимых шрифтов. Таких семейств пять: "serif" (шрифты с засечками), "sans-serif" (шрифты без засечек), "cursive", "fantasy" и "monospace" (моноширинные шрифты).

Допускается задавать несколько шрифтов через запятую. В этом случае сначала на клиентском компьютере ищется первый шрифт из списка, в случае неудачного поиска — второй, потом третий и т. д. Если ни один шрифт, определенный в свойстве, не найден, используется шрифт по умолчанию.

```
font-family: "Verdana", "Arial", "sans-serif";
```

Если атрибут `font-family` определен во встроенном стиле, названия шрифтов берутся в апострофы, а не кавычки.

```
<P STYLE="font-family: 'Verdana', 'Arial', 'sans-serif'">
```

Атрибут `font-size` определяет размер шрифта.

```
font-size: {Абсолютный размер в пунктах}pt|xx-small|x-small|small|
↳medium|large|x-large|xx-large|larger|smaller;
```

Как вы видите, размер шрифта может быть задан в абсолютных и относительных величинах. Абсолютные величины могут быть заданы в пунктах и в виде имени размера; доступно семь различных значений имен размеров от `xx-small` до `xx-large`. Значением по умолчанию является `medium`. В качестве относительных величин можно задать процент от величины шрифта родительского элемента (например, `120%`) или значения `larger` и `smaller`, представляющие следующий размер шрифта соответственно по возрастанию и убыванию. Так, например, если для родительского тега определен шрифт размера `medium`, то значение `larger` установит для текущего элемента шрифт `large`.

Атрибут `font-weight` устанавливает "жирность" шрифта.

```
font-weight: normal|bold|bolder|lighter|100|200|300|400|500|600|
↳700|800|900;
```

Здесь, как вы видите, доступны семь абсолютных значений от `100` до `900`, представляющих различную "жирность" шрифта, при этом обычный шрифт будет иметь "жирность" `400` (или `normal`), а полужирный — `700` (или `bold`). Значением по умолчанию является `normal` (или `400`). Значения `bolder` и `lighter` относительные и представляют следующие степени "жирности" соответственно в большую и меньшую сторону.

Атрибут `font-style` задает начертание шрифта.

```
font-style: normal|italic|oblique;
```

Доступны три значения, представляющие обычный шрифт (`normal`, это значение по умолчанию), курсив (`italic`) и наклонное начертание (`oblique`).

Последнее начертание отображается как курсив Internet Explorer и вообще не поддерживается Navigator.

Атрибут `font-variant` задает, как будут выглядеть большие и маленькие буквы шрифта.

```
font-variant: normal|small-caps;
```

Значение `small-caps` задает такое поведение шрифта, когда его маленькие буквы выглядят точно так же, как большие, просто меньшего размера. Значение по умолчанию — `normal`.

Текст

Эти атрибуты затрагивают уже не отдельные символы текста, а целые текстовые блоки.

Атрибут `word-spacing` позволяет установить дополнительное пространство, добавляемое к промежутку между словами и поддерживается только Internet Explorer.

```
word-spacing: normal|{Абсолютное или относительное значение};
```

Как вы видите, можно задать либо значение `normal` для нормального, стандартного интервала между словами (это значение по умолчанию), либо абсолютное или относительное значение, которое и задаст интервал.

Абсолютное значение может быть задано в:

- пикселах (в качестве значения задается простое число);
- дюймах (`{числовое значение}in`, например, `1.2in`);
- сантиметрах (`{числовое значение}cm`, `0.5cm`);
- миллиметрах (`{числовое значение}mm`, `5mm`);
- пунктах (`{числовое значение}pt`, `5pt`);
- пиках (`{числовое значение}pc`, `12pc`).

Относительные значения могут быть заданы в следующих единицах измерения:

- высота текущего шрифта (`{числовое значение}em`, например, `0.8em`);
- высота буквы "x" текущего шрифта (`{числовое значение}ex`, например, `1.1ex`).

Процентные значения в этом свойстве не допускаются.

Имейте в виду, что такие способы указания параметров размеров поддерживаются и другими атрибутами таблиц стилей. Если какой-то атрибут поддерживает не все из вышеперечисленных способов, я специально вас предупрежу.

Атрибут `letter-spacing` задает дополнительный промежуток между буквами и также поддерживается только Internet Explorer.

```
letter-spacing: normal|{Абсолютное или относительное значение};
```

Атрибут `text-decoration` задает "украшение", которое будет применено к тексту: подчеркивание, надчеркивание, зачеркивание или мерцание.

```
text-decoration: none|underline|overline|line-through|blink;
```

Здесь доступны пять разных значений:

- `none` — убирает все "украшения" (это поведение по умолчанию);
- `underline` — подчеркивает текст;
- `overline` — "надчеркивает", если так можно выразиться, текст, т. е. проводит линию над строками;
- `line-through` — зачеркивает текст;
- `blink` — заставляет текст мерцать (эта возможность поддерживается только Navigator).

Атрибут `vertical-align` задает вертикальное выравнивание текста.

```
vertical-align: baseline|sub|super|top|text-top|middle|bottom|
↳text-bottom;
```

Возможны восемь предопределенных значений:

- `baseline` (значение по умолчанию) — задает выравнивание базовой линии элемента страницы по базовой линии родителя;
- `sub` — превращает текст в верхний индекс;
- `super` — превращает текст в нижний индекс;
- `top` — выравнивает верх элемента страницы по самому верху родителя;
- `text-top` — выравнивает верх текста элемента страницы по верху текста родителя;
- `middle` — выравнивает центр элемента страницы по центру родителя;
- `bottom` — выравнивает низ элемента страницы по низу родителя;
- `text-bottom` — выравнивает низ текста элемента страницы по низу текста родителя.

Так или иначе, вам, скорее всего, придется поэкспериментировать с этим атрибутом, чтобы достичь нужных результатов. Это обычная практика в Web-дизайне.

Атрибут `text-transform` изменяет регистр символов текста.

```
text-transform: capitalize|uppercase|lowercase|none;
```

Этот атрибут позволяет преобразовать текст к верхнему (значение uppercase) или нижнему (lowercase) регистру, преобразовать к верхнему регистру первую букву каждого слова (capitalize) или не трогать его вообще (none, это, как вы уже поняли, поведение по умолчанию).

Атрибут text-align задает горизонтальное выравнивание текста.

```
text-align: left|right|center|justify;
```

Здесь доступны значения left (левое выравнивание; поведение по умолчанию), right (правое выравнивание), center (центрирование) и justify (выравнивание по ширине).

Атрибут text-indent задает отступ для "красной строки".

```
text-indent: {Абсолютное или относительное значение отступа};
```

Здесь допускаются абсолютные и относительные (относительно ширины текстового элемента), в том числе процентные, величины отступа. По умолчанию отступ "красной строки" равен нулю, т. е. отсутствует.

Атрибут text-height задает вертикальное расстояние между двумя строками (вернее, между базовыми линиями двух строк).

```
text-height: normal|{Абсолютное или относительное значение};
```

Здесь допускаются абсолютные и относительные (относительно высоты шрифта), в том числе процентные, величины расстояния. Все абсолютные значения при этом умножаются на высоту текущего шрифта и уже после этого используются. Таким образом, чтобы сделать стандартный отступ в два интервала, вы можете просто написать text-height: 2. Значение normal устанавливает обычную величину расстояния между строками; это значение свойства по умолчанию.

Атрибут white-space управляет разрывами текста.

```
white-space: normal|pre|nowrap;
```

Здесь доступны три значения:

- normal — устанавливает обычное поведение текста (это установка по умолчанию);
- pre — задает такое поведение текста, словно он заключен в теги <PRE> и </PRE>;
- nowrap — действует аналогично тегам <NOWRAP> и </NOWRAP>.

Отдельные абзацы

А эти атрибуты позволяют вам задать такие характеристики текстового абзаца, которые не снились обычному HTML, например украсить его рамкой и задать отступы.

Четыре атрибута margin-left, margin-top, margin-right и margin-bottom позволяют установить дополнительные отступы соответственно слева, сверху, справа и снизу. Я не буду приводить формат этих атрибутов, т. к. он очень прост. Вы задаете абсолютное или относительное (в том числе процентное) значение отступа; в случае процентного значения за основу берется ширина родительского элемента. Допускаются также отрицательные значения.

Атрибуты padding-left, padding-top, padding-right и padding-bottom задают промежуток соответственно слева, сверху, справа и снизу между элементом и рамкой, если она есть. Также допускаются абсолютные и относительные (в том числе процентные) значения отступа; в случае процентного значения за основу берется ширина родительского элемента. Отрицательные значения не допускаются.

Атрибуты border-left-width, border-top-width, border-right-width и border-bottom-width задают толщину линии рамки.

```
border-left-width|border-top-width|border-right-width|
↳border-bottom-width: thin|medium|thick|{Толщина};
```

Здесь можно задать либо числовое значение толщины рамки в пикселах или иных единицах измерения, либо одно из предопределенных значений: thin (тонкая), medium (средняя) или thick (толстая). По умолчанию линия рамки средняя, т. е. medium.

Атрибуты border-left-color, border-top-color, border-right-color и border-bottom-color задают цвет линии рамки.

```
border-left-color|border-top-color|border-right-color|
↳border-bottom-color: {Цвет};
```

По умолчанию значения этих атрибутов равны значению атрибута color или цвету по умолчанию, если атрибут color не определен.

Атрибуты border-left-style, border-top-style, border-right-style и border-bottom-style задают стиль линии рамки или ее отсутствие.

```
border-left-style|border-top-style|border-right-style|
↳border-bottom-style: none|dotted|dashed|solid|double|groove|
↳ridge|inset|outset;
```

Здесь доступны следующие значения:

- none — вообще не показывает никакую рамку, независимо от установок других атрибутов (это поведение по умолчанию);
- dotted — показывает пунктирную линию;
- dashed — показывает штриховую линию;
- solid — показывает сплошную линию;

- `double` — показывает двойную линию. Сумма толщин двух линий, составляющих ее, и пространства между ними равна значению соответствующего атрибута `border-***-width`;
- `groove` — показывает вдавленную трехмерную линию;
- `ridge` — показывает выпуклую трехмерную линию;
- `inset` — показывает трехмерную "ступеньку вверх";
- `outset` — показывает трехмерную "ступеньку вниз".

Списки

Эти атрибуты устанавливают параметры списков.

Атрибут `list-style-type` задает вид маркера списка.

```
list-style-type: disc|circle|square|decimal|lower-roman|upper-roman|
↳lower-alpha|upper-alpha|none;
```

Доступны следующие значения:

- `disc` — помечает каждую позицию списка черным кружком (это поведение по умолчанию);
- `circle` — помечает каждую позицию светлым кружком;
- `square` — помечает каждую позицию светлым квадратиком;
- `decimal` — нумерует каждую позицию арабскими цифрами;
- `lower-roman` — нумерует каждую позицию маленькими римскими цифрами;
- `upper-roman` — нумерует каждую позицию большими римскими цифрами;
- `lower-alpha` — помечает каждую позицию маленькими латинскими буквами;
- `upper-alpha` — помечает каждую позицию большими латинскими буквами;
- `none` — вообще никак не помечает позиции списка.

Атрибут `list-style-image` позволяет перекрыть установки, заданные предыдущим свойством, и задать в качестве маркера списка какое-либо графическое изображение.

```
list-style-image: {Адрес файла графического изображения}|none;
```

Заметьте, что если вы задали в качестве значения этого атрибута какой-либо интернет-адрес, установки `list-style-type` будут перекрыты. По умолчанию значение этого атрибута равно `none`.

Атрибут `list-style-position` позволяет задать более компактное отображение списка.

```
list-style-position: inside|outside;
```

Значение по умолчанию — outside. Если его установить в inside, то список будет отображаться более компактно, как это бывает при использовании атрибута COMPACT.

Виды элементов Web-страниц

Теперь пора поговорить о разных видах элементов Web-страниц и о том, чем они отличаются друг от друга.

Всего текущий стандарт HTML предлагает три вида различных элементов:

- *блок* или *блочный элемент*, отображающийся отдельно от других, например, абзац <P> или заголовок <H1>;
- *позиция списка*, ведущая себя аналогично блоку, но имеющая маркер списка;
- *встроенный элемент* (не путать со встроенным стилем), отображающийся внутри блока в основном потоке текста, например, рисунок <IMG>, разрыв
 или жирный текст <B>.

Вы можете изменять поведение элемента Web-страницы, просто присвоив нужное значение атрибуту display. Значение block задает элементу поведение блока (это значение по умолчанию), значение list-item — позиции списка, inline — встроенного элемента, а none запрещает ему показываться на экране. При этом Web-обозреватель вообще игнорирует такой элемент, словно его нет в коде Web-страницы.

Псевдостили гиперссылок

Несколько слов о стилях, которые можно применить к гиперссылкам (и, естественно, определить в таблице стилей). Это своего рода предопределенные стили, применяемые к гиперссылкам в особых случаях и называемые *псевдостилями*. Все псевдостили перечислены в табл. 1.11.

Таблица 1.11. *Псевдостили гиперссылок*

Псевдостиль	Описание
A: link	Применяется ко всем гиперссылкам документа. Аналогичен атрибуту LINK тега <BODY>
A: active	Применяется ко всем активным гиперссылкам документа. Аналогичен атрибуту ALINK тега <BODY>
A: visited	Применяется ко всем посещенным гиперссылкам документа. Аналогичен атрибуту VLINK тега <BODY>
A: hover	Применяется к гиперссылке, на которую указывает курсор мыши. Поддерживается только Internet Explorer

Вы можете переопределить их, как обычные стили, и таким образом изменить внешний вид гиперссылок вашего сайта.

```
A: active { color: yellow }
A: visited { text-decoration: line-through }
A: hover { color: yellow; text-decoration: none }
```

Тег <SPAN>

Очень хорошо! Просто замечательно! Теперь мы свободны форматировать наш текст как угодно. Но!..

Допустим, вы хотите взять фрагмент текста в каком-либо теге и применить к нему специальное форматирование с помощью только стилей (неважно, собранных в таблицу или встроенных). Каким образом это делать? Подбирать соответствующие теги форматирования текста: <B>, <I> или аналогичные и потом подгонять их под свои нужды с помощью переопределения? А есть ли способ лучше?

Есть. Это парный тег <SPAN> и </SPAN>. Вы просто заключаете нужный фрагмент текста внутрь этого тега и устанавливаете атрибуты CLASS и (или) STYLE. Как вы уже все догадались, <SPAN> — тег встроенного элемента.

```
<P>Это простой текст, а <SPAN STYLE="color: green;
font-style: italic">это — зеленый курсив</SPAN>.</P>
```

Пример красивой Web-страницы

Ну и, в заключение, перед тем как перейти к следующей, последней теме нашего краткого курса HTML, давайте рассмотрим пример большой и красивой Web-страницы, в оформлении которой были активно использованы стили, как объединенные в таблицу, так и встроенные в HTML-элементы. В качестве основы возьмем страницу, созданную нами в разделе, повествующем о таблицах и сохраненном в файле 1.11.htm. Усовершенствованный вариант мы сохраним в файле 1.13.htm.

```
<HTML>
<HEAD>
<TITLE>Красивый сайт со стилями</TITLE>
<STYLE>
BODY { font-family: "Arial"; font-size: 9pt }
<!-- Задаем стиль ячейки таблицы -->
TD.headercell { background-color: bisque }
<!-- И строки заголовка -->
P.header { color: midnightblue; font-size: xx-large; text-align: center;
white-space: nowrap}
```

```
P {margin-left: 0.5mm; margin-top: 0.5mm; margin-right: 0.5mm;
margin-bottom: 0.5mm }
P.copyright { text-align: right; font-style: italic; color: white }
<!-- А это - псевдостили гиперссылок -->
A: active { color: yellow }
A: visited { text-decoration: line-through }
A: hover { color: yellow; text-decoration: none }
</STYLE>
</HEAD>
<BODY>
<TABLE BORDER="1" CELLPADDING="5">
<TR>
<TD CLASS="headercell" COLSPAN="3">
<P CLASS="header">Красивый сайт</P>
</TD>
</TR>
<TR STYLE="vertical-align: top">
<TD CLASS="headercell">
  <TABLE>
  <TR><TD><P><A HREF="dummy.htm">Введение</A></P></TD></TR>
  <TR><TD><P><A HREF="dummy.htm">Глава 1</A></P></TD></TR>
  <TR><TD><P><A HREF="dummy.htm">Глава 2</A></P></TD></TR>
  <TR><TD><P><A HREF="dummy.htm">Заключение</A></P></TD></TR>
  </TABLE>
</TD>
<TD>
<!-- Какой-либо текст. Я сам скопировал сюда текст из этой книги -->
</TD>
<TD>
<P>Сегодня этот сайт был создан.</P>
</TD>
</TR>
<TR>
<TD COLSPAN="3" STYLE="background-color: black">
<P CLASS="copyright">Авторские права принадлежат мне</P>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

Результат вы можете видеть на рис. 1.13.

Вы можете пощелкать по гиперссылкам и посмотреть, что из этого выйдет. Заметьте, как гиперссылки изменяют свой цвет при наведении на них кур-

сора мыши (правда, видно это только в Internet Explorer). После того как вы щелкнули по ссылке и вернулись на предыдущую страницу, посещенная вами гиперссылка будет зачеркнута. А все потому, что мы переопределили псевдостили гиперссылок.

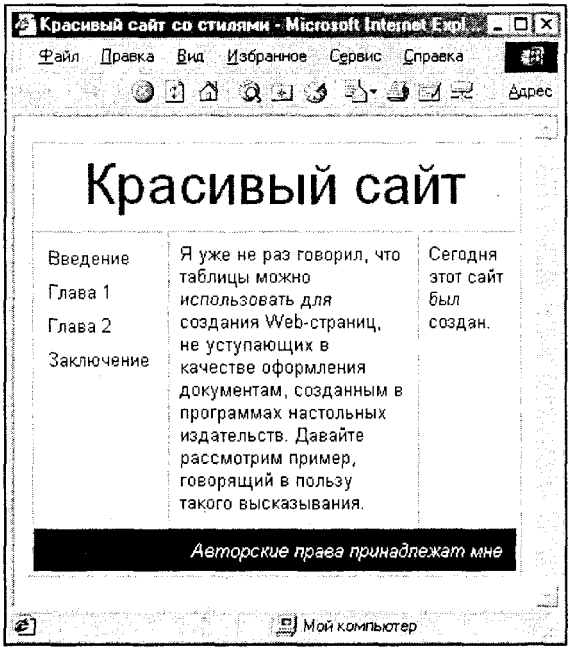


Рис. 1.13. Страница, в которой активно использовались таблицы стилей

Свободное позиционирование

Всем хорош HTML. Много у него различных возможностей форматирования текста, создания таблиц, связывания документов друг с другом. Однако одного до последнего времени не мог сделать ни один Web-дизайнер: взять какой-нибудь элемент страницы, неважно, фрагмент текста, рисунок или таблицу, в общем, блочный элемент, и поместить его в произвольное место страницы. То есть, сделать то, что обычный дизайнер, занимающийся печатными изданиями, проделывает одним движением мыши.

Но таблицы стилей устранили и этот недостаток HTML. Теперь вы можете поместить любой HTML-элемент в любое место Web-страницы просто задав его координаты и геометрические размеры.

Давайте рассмотрим пример небольшой странички, содержащий такой вот ***свободно позиционированный*** элемент. Сохраните ее в файле 1.14.htm.

```
<HTML>
<HEAD>
```

```
<TITLE>Пример Web-страницы</TITLE>
</HEAD>
<BODY>
<H1>Пример Web-страницы</H1>
<P>Это Web-страница, созданная только ради демонстрации того,
что стать Web-дизайнером может всякий. Для этого не нужна сложная в
настройке программа Web-сервера. В данном случае, ваша операционная
система с успехом заменяет ее.</P>
<P STYLE="background-color: white; top: 90; left: 150;
position: absolute">Я мешаю вам читать!</P>
<DIV STYLE="top: 70; left: 70; position: absolute">
<IMG SRC="Tips.gif" >
</DIV>
</BODY>
</HTML>
```

Результат можно увидеть на рис. 1.14. Да-да, такие элементы Web-страниц часто называют также *плавающими*. Вы видите, как рисунок и строка текста как бы "плавают" над обычным текстом Web-страницы и даже закрывают часть его.

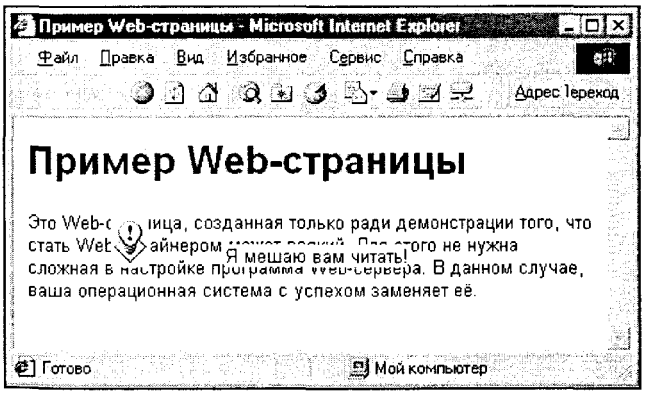


Рис. 1.14. Web-страница с "плавающим" рисунком

Рассмотрим внимательнее код нашей страницы. Мы видим какой-то новый, не знакомый еще нам тег и несколько новых атрибутов стиля, которые мы установили для этого тега. Давайте рассмотрим их.

Тег <*DIV*>

Свободно позиционировать можно только блочные элементы и позиции списка. В вышеприведенном примере мы позиционировали таким образом строку текста, заключенную в теги абзаца <P> и </P>. Но для того, чтобы позиционировать рисунок, являющийся встроенным элементом (т. к. для

него не переопределено свойство display), нам нужно поместить его в блочный тег. Идеальное решение — поместить его в парный тег <DIV> и </DIV>.

Визуально тег <DIV> не делает ничего. Он просто *группирует* один или несколько элементов Web-страницы в один. Конечно, он имеет атрибут ALIGN, выполняющий те же функции и принимающий те же значения, что аналогичный атрибут тега <P>. Но основное его предназначение — группировка нескольких элементов страницы в один, чтобы впоследствии ими было удобнее управлять как единым целым. В частности, позиционировать на Web-странице.

Атрибуты свободного позиционирования

Все волшебство свободного позиционирования осуществляется с помощью особых атрибутов стилей.

Первая пара атрибутов, которые вам нужно знать, — это left и top. Они задают соответственно горизонтальную и вертикальную координаты левого верхнего угла элемента Web-страницы. Можно указать как абсолютное значение в одной из доступных единиц измерения, так и процентное относительно ширины (для свойства left) и высоты (для свойства top) родительского элемента. Допускаются отрицательные значения; в этом случае элемент может полностью или частично оказаться за пределами окна Web-обозревателя. Также доступно значение auto, задающее такое поведение элемента страницы, когда Web-обозреватель сам определяет, где его отобразить (т. е., свободное позиционирование не работает). auto является значением по умолчанию.

Вторая пара атрибутов, которую нужно знать, — это width и height. Они задают соответственно ширину и высоту элемента Web-страницы. Здесь доступны те же самые значения, что и для предыдущей пары атрибутов.

Внимание!

Если высота или ширина родительского элемента установлена в auto (значение по умолчанию), процентное значение высоты или ширины дочернего элемента не имеет смысла.

И еще. Дело в том, что свободно позиционированные элементы ведут себя совершенно по-иному. Они как бы выпадают из общего "потока" HTML-документа. Поэтому к ним не могут быть применены такие свойства стиля, как margin-***, и некоторые другие. Подробности см. в приложении 2.

Еще один очень важный атрибут — position. Он определяет, относительно чего отсчитываются координаты элемента.

```
position: absolute|relative|static;
```

Как видите, здесь доступны три значения:

- `absolute` — задает *абсолютное позиционирование*, т. е., координаты отсчитываются относительно левого верхнего угла родительского элемента. В нашем примере мы использовали именно абсолютное позиционирование, отсчитывая координаты от левого верхнего угла страницы;
- `relative` — задает *относительное позиционирование*, т. е., координаты элемента отсчитываются относительно позиции, в которой Web-обозреватель разместил бы этот элемент, не будь он свободно позиционированным (иначе говоря, не имей бы он атрибутов `left` и `top`). В этом случае допускаются отрицательные координаты объекта;
- `static` — является значением по умолчанию и задает обычное поведение элемента HTML, когда он отображается без свободного позиционирования внутри "потока" текста. И в этом случае значения других атрибутов свободного позиционирования не имеют смысла.

Примечание

Атрибут `position` не наследуется от родителей!

Следующий важный атрибут называется `z-index` и устанавливает порядок, в котором свободно позиционированные элементы будут перекрывать друг друга. Здесь доступны как числовое значение, так и значение `auto`, дающее возможность самому Web-обозревателю задавать порядок перекрытия элементов (обычно определенные позднее элементы перекрывают определенные ранее). Значение `z-index` у родителя равно нулю, элементы с положительным значением `z-index` перекрывают родителя, причем наверху оказывается элемент с максимальным значением этого свойства. Соответственно, элементы с отрицательным значением `z-index` перекрываются родителем, причем самым нижним оказывается элемент с минимальным значением этого свойства. Значение по умолчанию — `auto`.

И еще один атрибут, который вам может понадобиться, — это `overflow`. Он задает поведение элемента, чье содержимое вылезает за его границы.

```
overflow: visible|hidden|scroll|auto;
```

Здесь доступны четыре значения:

- `visible` — позволяет расширить элемент так, чтобы все его содержимое отобразилось в нем полностью. Это поведение по умолчанию;
- `hidden` — скрывает все, что "не лезет ни в какие рамки". При этом пользователь никак не сможет просмотреть то, что скрыто от его глаз;
- `scroll` — аналогично предыдущему значению, но предусматривает полосы прокрутки;
- `auto` — отдает решение проблемы на откуп Web-обозревателю. Как правило, аналогично `scroll`.

И опять же, этот атрибут действует только тогда, когда `position` установлен в `absolute`. В противном случае действует значение по умолчанию, т. е. `visible`.

Последний атрибут может показаться совсем ненужным, но он окажется очень полезным, когда мы начнем программировать скрипты для наших страниц. Это атрибут `visibility`, устанавливающий видимость или невидимость элемента.

```
visibility: inherit|visible|hidden;
```

Здесь доступны три значения:

- `inherit` — задает наследование видимости от родителя. То есть, если родитель видим, то видим и дочерний элемент. Это значение по умолчанию;
- `visible` — задает видимость элемента независимо от видимости родителя;
- `hidden` — скрывает элемент также независимо от видимости родителя.

Заметьте, что невидимый элемент все еще занимает какое-то место на странице и влияет на положение других элементов. Если же вы хотите скрыть элемент совсем, словно его и нет в HTML-коде, установите значение атрибута `display` в `none`.

Ну вот и все. Давайте теперь рассмотрим уже ставший традиционным пример. Это будет страничка, все содержимое которой размещено в свободно позиционируемых элементах. Сохраните ее в файле 1.15.htm.

```
<HTML>
<HEAD>
<TITLE>Свободу HTML-элементам!</TITLE>
<STYLE>
BODY { font-family: "Arial"; font-size: 9pt }
</STYLE>
</HEAD>
<BODY>
<DIV STYLE="top: 5; left: 5; width: 200; position: absolute">
<!-- Какой-нибудь текст. Сам я вклеил сюда фрагмент книги -->
</DIV>
<DIV STYLE="top: 5; left: 210; width: 200; height: 100;
position: absolute; overflow: scroll">
<!-- Какой-нибудь текст -->
</DIV>
<DIV STYLE="top: 110; left: 210; width: 200; height: 100;
position: absolute; overflow: hidden">
<!-- Какой-нибудь текст -->
</DIV>
<DIV>
```

```
<IMG SRC="Hlplogo.gif">
</DIV>
</BODY>
</HTML>
```

На рис. 1.15 вы можете видеть нашу страницу. Заметьте, что у двух элементов, позиционированных в колонку справа, мы задали значения атрибутов `overflow` соответственно `scroll` и `hidden` и получили требуемый результат. У элемента, что находится слева, атрибут `overflow` мы совсем не трогали, поэтому было использовано значение по умолчанию `visible`. А рисунок вообще не позиционирован свободно, а находится в "потоке" текста и лежит позади всех наших свободно позиционированных элементов. (В принципе, тег `<DIV>`, в который заключен рисунок, можно убрать — в данном случае он не нужен и просто засоряет нашу страницу.)

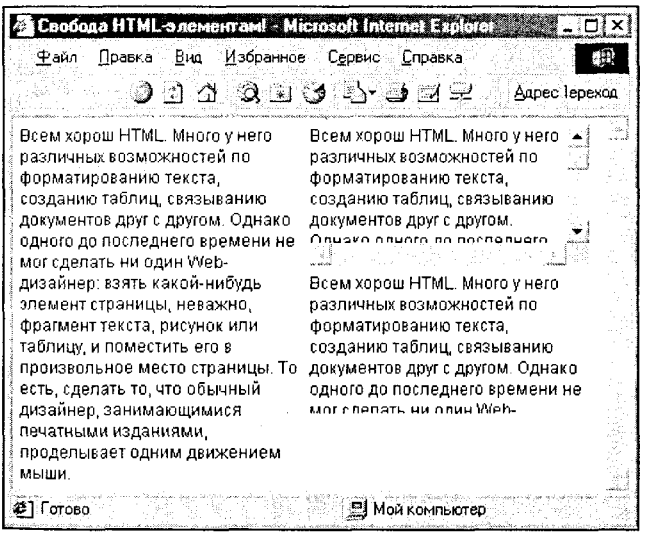


Рис. 1.15. Web-страница со свободно позиционированными элементами

Слои Web-страницы

Слои поддерживаются только Navigator и представляют собой альтернативную технологию свободного позиционирования элементов Web-страницы. Слой может содержать любой HTML-код и располагаться где угодно на странице, закрывая другие элементы или будучи закрытым ими.

Слои бывают двух типов: абсолютно (устанавливаются тегами `<LAYER>` и `</LAYER>`) и относительно (`<ILAYER>` и `</ILAYER>`) позиционированные. Они имеют совершенно одинаковый формат.

```
<[I]LAYER [ID="{Имя слоя}"] [SRC="{Адрес внешнего документа}"]
[LEFT="{Горизонтальная координата верхнего левого угла}"]
```

```
[TOP="{Вертикальная координата верхнего левого угла}"]
[WIDTH="{Ширина слоя}"] [Z-INDEX="{Уровень перекрытия}"]
[ABOVE="{Имя слоя}"] [BELOW="{Имя слоя}"]
[VISIBILITY="show|hidden"] [BGCOLOR="{Цвет фона слоя}"]
[BACKGROUND="{Адрес файла фонового рисунка}"]>
<!-- Содержимое слоя -->
</[I]LAYER>
```

Атрибут ID задает имя слоя. По умолчанию все слои безымянные.

Атрибут SRC позволяет задать внешний файл, который будет отображен в слое. (Здесь поведение слоя аналогично "плавающему" фрейму.) Заметьте, что некоторые старые версии Navigator требуют, чтобы такой внешний документ содержал только собственно тело документа без заголовка и собственно тегов <BODY> и </BODY>.

Атрибуты LEFT и TOP задают соответственно горизонтальную и вертикальную координаты верхнего левого угла слоя. Для абсолютно позиционированных слоев они отсчитываются от верхнего левого угла окна Web-обозревателя, а для относительно позиционированных — от точки, где они должны были появиться, если бы не были свободно позиционированы. По умолчанию они равны нулю.

Атрибут WIDTH задает ширину слоя. Учтите, что в настоящее время нет возможности задать высоту слоя.

Атрибуты Z-INDEX, ABOVE и BELOW позволяют задать порядок, в котором слои будут перекрывать друг друга. Можно задать либо числовое значение в атрибуте Z-INDEX, либо один из атрибутов ABOVE или BELOW, в результате чего слой будет находиться соответственно уровнем выше или ниже указанного в этом атрибуте.

Атрибут VISIBILITY задает видимость слоя. Значение show показывает слой (это поведение по умолчанию), hidden скрывает.

Атрибуты BGCOLOR и BACKGROUND задают соответственно цвет фона и фоновый рисунок. Учтите, что если фоновый рисунок шире слоя, то слой будет расширен, чтобы вместить весь рисунок.

И, как обычно, пример. В качестве основы возьмем нашу страницу со свободно позиционированными элементами и перепишем ее с использованием слоев. К сожалению, слои менее мощны, чем блочные элементы, поэтому упростим наш пример. Сохраним его в файле 1.16.htm.

```
<HTML>
<HEAD>
<TITLE>Свободу HTML-слоям!</TITLE>
<STYLE>
BODY { font-family: "Arial"; font-size: 9pt }
```

```
</STYLE>
</HEAD>
<BODY>
<LAYER TOP="5" LEFT="5" WIDTH="200">
<!-- Какой-нибудь текст. Сам я вклеил сюда фрагмент книги -->
</LAYER>
<LAYER TOP="5" LEFT="210" WIDTH="150">
<!-- Какой-нибудь текст -->
</LAYER>
<DIV>
<IMG SRC="Hlplogo.gif">
</DIV>
</BODY>
</HTML>
```

Результат всего этого вы можете видеть на рис. 1.16. (Не забывайте, что открывать Web-страницу нужно только в Navigator — Internet Explorer не отобразит в этом случае ничего интересного.)

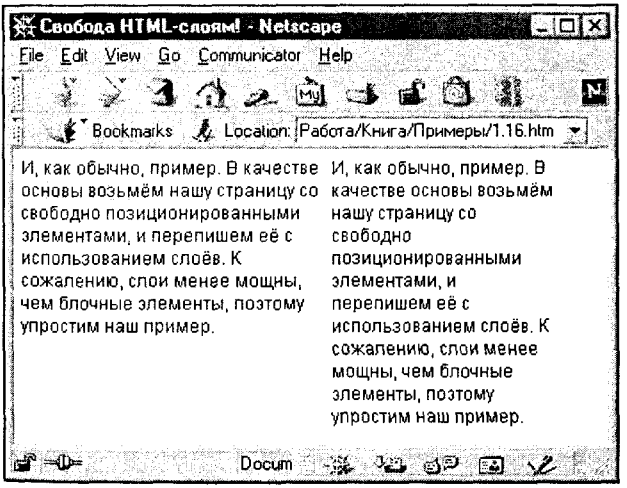


Рис. 1.16. Web-страница со слоями

Я уже говорил, что нет возможности задать высоту слоя. Вместо этого Navigator сам установит высоту так, чтобы отобразить все содержимое слоя. Что и видно на рисунке.

Так зачем мы рассмотрели здесь эти неуклюжие слои? Дело в том, что Navigator предлагает намного меньше возможностей по динамическому управлению содержимым Web-страницы, чем Internet Explorer. И практически единственными элементами страниц, которыми можно управлять из программ, являются слои. Но об этом будет рассказано позже.

Теги <NOLAYER> и </NOLAYER>

Для Web-обозревателей, не поддерживающих слои, Navigator приготовил утешительную пилюлю в виде тегов <NOLAYER> и </NOLAYER>. Внутри их помещается текст, отображаемый на экране в таких Web-обозревателях; теги определения слоев в данном случае игнорируются как неизвестные. В это же время Navigator игнорирует теги <NOLAYER> и </NOLAYER>.

```
<NOLAYER>Здесь должен быть слой, но вам его не видать.</NOLAYER>
```

Web-скрипты

Вот мы и изучили те возможности HTML, что применяются чаще всего. Курс был немного скомканным, но я сразу предупредил, что это только краткое введение. Для более полного описания возможностей HTML см. приложения 1 и 2. А я перехожу к следующей теме: к Web-скриптам и Web-программированию на JavaScript.

HTML очень статичен. В нем невозможно заставить двигаться какую-то часть документа или отобразить в определенном месте страницы какие-то данные, явившиеся результатом какого-то события. Для этого и были созданы скриптовые языки, в частности JavaScript.

Давайте рассмотрим последний в этой главе пример. Сохраним его в файле 1.17.htm.

```
<HTML>
<HEAD>
<TITLE>Сегодня</TITLE>
</HEAD>
<BODY>
<P>
<SCRIPT LANGUAGE="JavaScript">
var d;
d=new Date();
document.write(d.toString());
</SCRIPT>
</P>
</BODY>
</HTML>
```

Результат виден на рис. 1.17.

Здесь в Web-страницу встроен небольшой скрипт, выводящий на страницу текущую дату. И он работает!

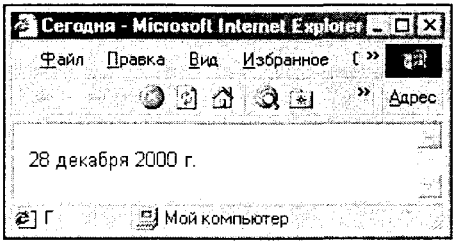
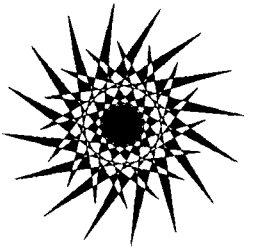


Рис. 1.17. Текущая дата на Web-странице

Хотите знать больше?

В следующей главе мы рассмотрим описание языка JavaScript. Для того чтобы писать сложные скрипты, нам обязательно нужно знать этот язык.

ГЛАВА 2



Язык JavaScript

В этом разделе описывается язык сценариев JavaScript. Краткая история этого языка и принципы его работы изложены во введении, так что повторять их не буду. Вместо этого сразу перейду к собственно языку.

Во введении я говорил, что JavaScript относится к категории так называемых интерпретируемых языков, т. е., виртуальная машина Java читает исходный код программы и тотчас выполняет его. Этим JavaScript отличается от компилируемых языков, в которых исходный код программы переводится в команды процессора с помощью программы-компилятора. Виртуальная машина, выполняющая программы на интерпретируемом языке, именуется также *интерпретатором* — это более старое и более распространенное среди программистов название.

Самое начало

Что такое программа

Так что же такое программа? Давайте рассмотрим небольшой пример программы на JavaScript.

```
function someFunction(x, y, z, t) {
  var a, b, c;
  a = x * y;
  b = z / t;
  c = a - b;
  return c;
}
```

Вы видите набор каких-то формул. Собственно, эти формулы описывают функцию, принимающую в качестве параметров четыре числа и производя-

щую над ними какие-то вычисления. Именно на примере этой функции мы и будем изучать JavaScript.

Прежде всего заметим, что любая программа на любом языке содержит *выражения*. Выражение — это описание некоторых действий, выполняемых программой. Выражения состоят из *операторов* и *операндов*; первые задают действия, которые необходимо выполнить, вторые — данные, над которыми нужно произвести описанные действия. Операндами могут быть переменные и литералы.

Но прежде, чем завести разговор о переменных, операторах и принципах построения выражений, нам нужно разобраться с данными.

Типы данных

Что делает любая программа? Правильно, обрабатывает данные. Данные вводятся в программу, преобразуются согласно каким-то правилам и потом представляются пользователю в удобном для него виде. Собственно, это главная задача любой программы: получить "сырые" данные, преобразовать их и выдать пользователю. Попутно заметим, что описание правил, согласно которым преобразуются данные в программе, называется *алгоритмом*.

Язык JavaScript, как и большинство современных языков программирования, манипулирует данными, относящимися к определенному типу. Тип данных — это своего рода разновидность данных: число, текстовая строка или что-то еще. Он определяет множество возможных значений данных и применимых к ним операций. В табл. 2.1 перечислены все типы данных, которые понимает JavaScript.

Таблица 2.1. *Типы данных языка JavaScript*

Тип данных	Описание
Строковый	Обычные строки текста. Должны быть заключены в двойные или одинарные (апострофы) кавычки. Причем, текст, ограниченный двойными кавычками, может содержать одинарные кавычки, и наоборот
Целочисленный	Целые числа. Могут быть десятичными, восьмеричными или шестнадцатеричными. Восьмеричные целые числа обозначаются лидирующим нулем и могут состоять только из цифр от 0 до 7 (например, `047`, `-012543624`). Шестнадцатеричные целые обозначаются знаками `0X` или `0x`, помещенными в начало (например, `0x35F`), и состоят из цифр от 0 до 9 и букв от A до F
С плавающей точкой	Дробные числа. Содержат целую и дробную часть, разделенные точкой; точка обязательна даже в том случае, если дробная часть отсутствует. Могут быть записаны в экспоненциальной форме: `{мантисса}E{порядок}`. Дробные числа могут быть только десятичными

Таблица 2.1 (окончание)

Тип данных	Описание
Логический	Величины вида "да-нет". Их значениям соответствуют ключевые слова `true` и `false` — "истина" и "ложь". Подробнее о логических величинах мы поговорим позднее
null	Тип данных null, что означает отсутствие данных. Обозначается ключевым словом `null`

Это наиболее часто употребляемые типы данных в JavaScript. Позднее мы рассмотрим еще несколько типов данных и их использование.

Примечание

Логический тип начал поддерживаться в Internet Explorer только с версии 4.0 и в Navigator с версии 3.0.

Литералы

Литералами называются данные какого-либо типа, записанные в соответствии с правилами языка программирования, в данном случае, языка JavaScript. Правила написания литералов рассмотрены в табл. 2.1. Здесь мы только приведем несколько примеров:

```
"Строка"
'Строка в "апострофах"'
```

`123456789` — это десятичное целое;

`01234567` — это восьмеричное целое;

`0x17AF` — это шестнадцатеричное целое;

`7.8` — это число с плавающей точкой;

`45.` — это тоже число с плавающей точкой, но не имеющее дробной части;

`1.2E-10` — записанное по правилам JavaScript число $1{,}2 \cdot 10^{-10}$.

Переменные

Для того чтобы успешно обрабатывать данные, программе нужно где-то их хранить. Для этого выделяются специальные участки памяти, называемые *переменными*.

Каждая переменная должна иметь имя, которое однозначно идентифицирует ее. Короче говоря, не должно быть двух переменных с одним и тем же именем (об исключениях из этого правила поговорим позже). Имя должно состоять из латинских букв, цифр и знаков подчеркивания "_", причем пер-

вым символом должны быть либо латинская буква, либо знак подчеркивания. Например, `FrameName`, `_link`, `GoToKudaMakarTelyatNeGonyal` — правильные имена переменных, а `678vasya`, `ИмяФрейма` — неправильные. Имейте также в виду, что язык JavaScript чувствителен к регистру букв, так что `framename` и `FrameName` — разные переменные.

И еще. Имена переменных не должны совпадать с так называемыми *ключевыми словами*, т. е. словами, используемыми для написания операторов языка. Таких ключевых слов не очень много, и их легко запомнить.

Первым делом нужно переменную объявить. Это требуется для того, чтобы интерпретатор JavaScript знал, что это переменная, и не принял ее за неправильно написанный оператор. Переменные объявляются с помощью ключевого слова `var`:

```
var FrameName, _link, GoToKudaMakarTelyatNeGonyal;
```

Это пример правильного объявления переменных. (Ключевое слово `var` выделено жирным шрифтом.) Заметьте, что вы можете объявить сразу несколько переменных, разделив их имена запятыми.

Что можно делать с переменными? Присваивать им какое-либо число, извлекать из них содержимое, производить над ним какие-то операции. В общем, хранить исходные значения, промежуточные или окончательные результаты вычислений, делать то, для чего они, собственно, и предназначены.

```
FrameName = "HeaderFrame";
_link = 4;
```

Здесь мы присвоили переменным значения с помощью оператора присваивания `=`. Можно непосредственно при объявлении переменных присвоить им начальные значения:

```
var FrameName = "HeaderFrame", _link = 4;
```

Можно также присвоить переменной значение `null`. Это используется для полной очистки переменной (если такая надобность возникнет):

```
somevar = null;
```

Выражения

Выше мы привели несколько примеров выражений языка JavaScript. Это были простейшие выражения, объявлявшие переменные и присваивавшие им какие-либо значения. Они содержали операторы объявления переменных `var` и присваивания `=`, а также имена переменных и литералы в качестве операндов.

Дадим несколько правил, в соответствии с которыми строятся выражения в языке JavaScript. Прежде всего, каждое выражение JavaScript должно за-

канчиваться точкой с запятой ";". Выражение может занимать как одну строку, так и несколько строк, и в этом случае точка с запятой является признаком конца выражения. Выражение может содержать сколько угодно операторов и операндов. Однако следует отдавать себе отчет в том, что слишком уж сложные выражения трудны для понимания не только другого программиста, но зачастую и для самого интерпретатора. Поэтому лучше разбить их на более простые выражения.

А теперь давайте рассмотрим принципы написания различных выражений JavaScript и используемые при этом операторы.

Выражения обработки данных

Арифметические выражения JavaScript состоят из двух частей, разделенных оператором присваивания =. Слева от него указывается переменная, которой присваивается новое значение, а справа — это самое новое значение. Оно может представлять собой литерал, другую переменную или результат вычисления какого-либо выражения.

Как иллюстрацию вышесказанного, рассмотрим несколько примеров (не забываем ставить точку с запятой в конце каждого выражения):

```
a = 127;
b = a;
c = a + b;
```

Эти три выражения абсолютно правильны. Здесь переменной, указанной слева от оператора =, присваивается новое значение, заданное справа от этого оператора. Похоже на математические формулы, только написанные в более строгой нотации.

Арифметические операторы

Раз уж мы заговорили о математике, давайте рассмотрим все арифметические операторы, которые предлагает нам JavaScript. Их довольно много.

Прежде всего, это обычные, прекрасно нам всем знакомые операторы сложения (+), вычитания (-), умножения (*) и деления (/). Они принимают два операнда и выдают один результат. В дополнение к ним существует также оператор взятия остатка (%), делящий одно число на другое нацело и возвращающий остаток от деления.

```
c = a + b;
x1 = x2 / y + ffc;
```

Оператор *математической инверсии* изменяет знак операнда. Так, положительное число становится отрицательным, а отрицательное — положительным.

```
x = -y;
```

Операторы *инкремента* ++ и *декремента* -- увеличивают или уменьшают значение операнда на единицу. При этом, если знак оператора стоит перед операндом, то сначала производится операция — инкремент или декремент — а потом возвращается значение операнда, уже, естественно, измененное. Если же знак оператора стоит после операнда, то сначала возвращается неизмененное значение операнда, а потом производится операция.

```
a = b++;
c = --c;
```

Последний оператор абсолютно верен! В этом случае из переменной c сначала берется старое значение, потом декрементируется и снова присваивается переменной c. JavaScript и другие языки программирования допускают такую запись, не применимую в обычной арифметике.

Оператор обработки строк

Оператор обработки строк всего один. Это оператор слияния строк +. Он похож на своего коллегу из арифметического цеха, но соединяет две строки, помещая вторую в конец первой.

```
title = "Java" + "Script";
```

Двоичные операторы

Двоичные операторы, называемые также побитными, затрагивают двоичное представление чисел и используются, в основном, для низкоуровневого программирования на JavaScript. Всего их семь и они перечислены в табл. 2.2. Я не стану подробно останавливаться на них, просто перечислю и дам краткое описание.

Таблица 2.2. *Двоичные операторы*

Оператор	Обозначение	Описание
Двоичная инверсия	`a = ~b`	
Двоичное И	`a = b & c`	
Двоичное ИЛИ	`a = b | c`	
Двоичное исключающее ИЛИ	`a = b ^ c`	
Сдвиг влево	`a = b << c`	Сдвиг влево на c знаков с заполнением нулями младших разрядов
Сдвиг вправо	`a = b >> c`	Сдвиг вправо на c знаков с заполнением старших разрядов содержимым знакового (старшего) разряда

Таблица 2.2 (окончание)

Оператор	Обозначение	Описание
Сдвиг вправо без учета знака	`a = b >>> c`	Сдвиг вправо на `c` знаков с заполнением старших разрядов нулями

Операторы присваивания

Мы уже знаем один оператор присваивания — =. Это универсальный оператор, который может пригодиться в любом случае. Но JavaScript предоставляет нам еще несколько операторов присваивания, призванных, в основном, упростить построение сложных выражений обработки данных. Они перечислены в табл. 2.3.

Таблица 2.3. Операторы присваивания

Оператор	Эквивалентное выражение
`a += b;`	`a = a + b;`
`a -= b;`	`a = a - b;`
`a *= b;`	`a = a * b;`
`a /= b;`	`a = a / b;`
`a %= b;`	`a = a % b;`
`a <<= b;`	`a = a << b;`
`a >>= b;`	`a = a >> b;`
`a >>>= b;`	`a = a >>> b;`
`a &= b;`	`a = a & b;`
`a ^= b;`	`a = a ^ b;`
`a |= b;`	`a = a | b;`

Можно писать выражения вида `a = a {Оператор} b`, если вам так привычнее. Но лучше для этой цели использовать сложные операторы присваивания; тогда выражение будет иметь вид `a {Оператор}= b`. Например, два следующих выражения делают одно и то же:

```
a = a + 10;
a += 10;
```

Совместимость типов данных

И под конец рассмотрим еще один важный вопрос — совместимость типов данных и автоматическое преобразование одного типа к другому.

Что случится, если сложить два числовых значения? Правильно, еще одно числовое значение. А если сложить число и строку? Тут интерпретатор JavaScript сталкивается с проблемой *несовместимости типов* данных. И пытается сделать эти типы совместимыми, преобразуя один из них к другому. В нашем случае, число будет преобразовано в строку, и потом две полученные строки — слиты в одну:

```
var a, b, c;
a = 11;
b = "12";
c = a + b;
```

Значение переменной `a` будет преобразовано в строковый тип — "11". Таким образом, переменная `c` будет содержать значение "1112" — результат слияния этих двух строк.

Теперь рассмотрим следующий пример:

```
var a, b, c;
a = 11;
b = "12";
c = a - b;
```

В этом случае интерпретатор преобразует значение переменной `b` в числовое значение — 12. И результатом вычисления выражения, помещенным в переменную `c`, станет −1.

Как видите, интерпретатор JavaScript делает все, чтобы выполнить вашу программу. Иногда это хорошо, но часто оказывает программистам медвежью услугу: программа работает не так, как планировалось, или, в конце концов, останавливается по ошибке совсем в другом месте, на абсолютно верном операторе. Поэтому лучше всего не допускайте подобных казусов, оперируйте только переменными совместимых типов и выполняйте преобразование типов сами. Как это делается, мы рассмотрим позднее, когда будем изучать объекты языка JavaScript.

Оператор *typeof*

Оператор `typeof` принимает в качестве операнда переменную или выражение и возвращает строку, описывающую тип данных операнда. Если операнд имеет численный тип (целый или с плавающей точкой), возвращается строка `"number"`, если строковый тип, то `"string"`, а для логического типа — `"boolean"`. Если же переменная вообще не содержит данных (была объявлена оператором `var`, но значение так и не было присвоено) или выражение возвращает неопределенный результат (из-за неудачного преобразования типов), возвращается `"undefined"`.

Оператор `typeof` имеет следующий формат.

```
typeof({Выражение})
```

Рассмотрим несколько примеров использования этого оператора.

```
if typeof(a + 1) == "number"
f = typeof(somevar);
```

В последнем случае мы присваиваем результат оператора `typeof` переменной `f`. Помните, что он имеет строковый тип.

Примечание

Оператор `typeof` появился в Navigator только начиная с версии 3.0.

Приоритет операторов

И еще одно понятие, которое должен иметь в виду программист, создавая сложные выражения. Это приоритет операторов, т. е. последовательность их выполнения.

Рассмотрим такое выражение:

```
a = b + c - 10;
```

В этом случае сначала к значению переменной `b` будет прибавлено значение `c`, а потом из результата вычтено 10. Операторы данного выражения выполняются слева направо.

```
a = b + c * 10;
```

А здесь этот принцип не действует! Сначала будет выполнено умножение значения `c` на 10, а уже потом к результату — прибавлено значение `b`. То есть, обычный порядок выполнения операторов "слева направо" будет нарушен. А все потому, что умножение имеет больший *приоритет*, чем сложение.

А оператор присваивания `=` имеет самый низкий приоритет. Вот поэтому прежде всего вычисляется выражение, а потом его результат присваивается какой-либо переменной.

В табл. 2.4 перечислены все изученные нами операторы в порядке убывания приоритета.

Таблица 2.4. *Приоритет операторов (в порядке убывания)*

Оператор(ы)	Описание
`++ -- - ~ typeof`	Инкремент, декремент, арифметическая и логическая инверсия, определение типа
`* / %`	Умножение, деление, остаток от деления
`+ -`	Сложение или слияние строк, вычитание
`<< >> >>>`	Двоичные сдвиги

Таблица 2.4 (окончание)

Оператор(ы)	Описание
&	Двоичное И
^	Двоичное исключающее ИЛИ
|	Двоичное ИЛИ
= {Оператор}=	Простое и сложные присваивания

Операторы, имеющие одинаковый приоритет, выполняются слева направо. Как в нашем первом примере в этой главе.

Вам следует запомнить эту таблицу. Неправильный порядок выполнения операторов может стать причиной трудно выявляемых ошибок, когда внешне абсолютно правильное выражение дает неверный результат.

Но что делать, если нам нужно нарушить обычный порядок выполнения операторов? Воспользуйтесь скобками:

```
a = (b + c) * 10;
```

Здесь сначала выполняется сложение `b` и `c`, а уж потом результат будет умножен на 10. Вы видите, что первыми выполняются операторы, заключенные в скобки.

Операторы в скобках также подчиняются приоритету. Поэтому часто используют многократно вложенные скобки:

```
a = ((b + c)*10 - d) / 2 + 9;
```

Здесь операторы будут выполнены в такой последовательности:

1. Сложение `b` и `c`.
2. Умножение результата на 10.
3. Вычитание `d` из произведения.
4. Деление результата на 2.
5. Прибавление 9 к частному.

Попробуйте удалить скобки и посмотрите, что получится.

Специальные символы

Строковые выражения JavaScript могут включать в себя так называемые специальные символы — комбинации знаков, обозначающие какие-либо служебные или непечатаемые символы, которые иначе невозможно вставить в выражение. Все доступные в JavaScript специальные символы перечислены в табл. 2.5.

Таблица 2.5. *Специальные символы*

Символ	Описание
\b	Забой (backspace)
\f	Новая страница (перевод страницы)
\n	Новая строка (перевод строки)
\r	Возврат каретки
\t	Горизонтальная табуляция
\'	Апостроф
\"	Кавычка
\\	Обратный слэш

Теперь, скажем, если вы хотите включить в вашу строку символ возврата каретки, напишите следующее выражение:

```
"Some string with carriage return/n"
```

Условные операторы

Условные операторы выделены в особую группу потому, что они занимаются не обработкой данных, а управлением работой самой программы. Грубо говоря, условный оператор может выполнить (или, наоборот, не выполнить) некий участок кода при наступлении (или не наступлении) некоего условия. Таким условием служит значение логической переменной или результат вычисления логического выражения. (Вот зачем нужен логический тип данных!)

Здесь необходимо дать понятие логического выражения. Это обычное выражение языка JavaScript, но результатом его вычисления является логическая величина `true` или `false`. Логическое выражение содержит операторы логического сравнения, которые будут описаны ниже.

Давайте рассмотрим все условные операторы, доступные нам в языке JavaScript. Но сначала несколько слов о блочных выражениях.

Блочные выражения

JavaScript позволяет вам объединить несколько выражений в одно, называемое *блочным* или просто *блоком*. Для этого нужный участок кода заключается в фигурные скобки { и }.

```
a = 11;
{
b = "12";
c = a - b;
}
```

Блоки используются для построения сложных выражений, в частности, выражений с условными операторами.

Оператор ветвления *if-else*

Оператор ветвления позволяет вам выполнить некий участок кода при наступлении некоего события. Таким событием может служить присвоение переменной какого-либо значения или результат вычисления какого-нибудь выражения, на который так или иначе надо реагировать особым способом.

Оператор ветвления имеет следующий формат:

```
if ({Условие}) . . . Блок "то"
[else
. . . Блок "иначе"]
```

Условие — это логическое выражение, согласно которому интерпретатор принимает решение, какой блок выполнить. Если условие имеет значение `true` ("истина"), то выполняется блок "то". Если же условие имеет значение `false` ("ложь"), то выполняется блок "иначе", коль скоро он есть. Если же блок "иначе" отсутствует, выполняется следующее выражение программы.

Примечание

Если результатом условия является `null` или `undefined`, то оператор ветвления реагирует на него, как на `false`.

Рассмотрим несколько примеров.

```
if (x == 1) {
  f = 3;
  h = 4;
  }
else
  {
  f = 33;
  h = 44;
  }
```

Здесь мы сравниваем значение переменной `x` с единицей и в зависимости от результатов сравнения присваиваем переменным `f` и `h` разные значения. Обратите внимание на условие — именно так записывается *оператор логического сравнения*.

Условие может быть довольно сложным.

```
if (x == 1 && y > 10)
  f = 3;
else
  f = 33;
```

Здесь мы использовали сложное условие, возвращающее `true` в случае, если значение переменной `x` равно единице И значение переменной `y` больше десяти. (Заметьте также, что мы подставили одиночные выражения вместо блоков.)

```
if (string1 == "Вася") string2 = "Привет, Вася!";
```

А здесь мы в условии сравниваем строки, а все выражение записано в одной строке. Так тоже можно.

Операторы сравнения

Операторов сравнения довольно много, и все они перечислены в табл. 2.6. Пожалуй, они покрывают все потребности программистов-математиков.

Таблица 2.6. *Операторы сравнения*

Оператор	Описание	Оператор	Описание
<	Меньше	&&	Логическое И
>	Больше	||	Логическое ИЛИ
==	Равно	!	Логическое НЕ
<=	Меньше или равно	===	Строго равно
>=	Больше или равно	!==	Строго не равно
!=	Не равно		

С первыми шестью операторами не должно быть никаких сложностей — они просто сравнивают два значения и возвращают `true`, если условие выполняется, и `false`, если нет.

```
param1 != param2
a > b
wigth >= height
```

Следующие три оператора позволяют строить более сложные логические выражения. Они выполняют логические операции И, ИЛИ и НЕ над двумя логическими выражениями и возвращают результат. В табл. 2.7 и 2.8 показаны результаты, возвращаемые этими операторами при разных значениях операндов.

Остается рассмотреть два последних оператора: "строго равно" и "строго не равно". Дело в том, что обычные операторы "равно" и "не равно", если встречают операнды разных типов, пытаются преобразовать их к одному типу. Операторы строгого равенства и строгого неравенства такого преобразования не делают, а в случае несовместимости операндов возвращают `false`. Иногда это бывает полезно, например при проверке ввода пользователя.

Таблица 2.7. *Операторы "&&" и "||"*

Операнд 1	Операнд 2	&& (И)	|| (ИЛИ)
true	true	true	true
true	false	false	true
false	true	flase	true
false	false	false	false

Таблица 2.8. *Оператор "!"*

Операнд	! (НЕ)
true	false
false	true

Рассмотренные нами операторы сравнения имеют разный приоритет. Самый высокий — у оператора ! (логическое НЕ), такой же, как у операторов инкремента, декремента и инверсии. Операторы <, >, <= и >= идут сразу за операторами битового сдвига, а вслед за ними — операторы ==, !=, === и !==. За двоичным ИЛИ идет логическое И, а за ним — логическое ИЛИ. У них — самый низкий приоритет.

И в заключение несколько примеров:

```
a == 2 && b >= c + 7
param1 != param2 || param3 > param4 * 2 + 1
! (d < 0)
```

Заметьте, что мы заключили условную часть последнего выражения в скобки. Если бы мы этого не сделали, интерпретатор сначала вычислил бы логическое НЕ для значения d, а потом сравнил бы его с нулем. Вспомните, что логическая инверсия имеет больший приоритет, чем сравнение. Результатом было бы, скорее всего, сообщение об ошибке...

Как можно применить логические величины? Разумеется, в условных выражениях.

```
if (a == 2 && b >= c + 7) ...
```

Или присвоить результат логического выражения переменной и использовать потом.

```
var flag;
. . .
flag = (a == 2 && b >= c + 7);
if (flag) ...
```

Примечание

Операторы строгого сравнения начали поддерживаться в Navigator только с версии 4.06.

Оператор ?

Это еще один условный оператор.

```
{Условие} ? {Выражение "то"} : {Выражение "иначе"}
```

Этот оператор возвращает результат выражения "то", если условие истинно, и результат выражения "иначе" — в противном случае.

```
a = (f == 2) ? b : c + 2;
```

Если `f` равно 2, выражение поместит в переменную `a` значение переменной `b`, в противном случае — значение выражения `c + 2`.

Приоритет этого оператора один из самых низких. Ниже него — только операторы присваивания.

Оператор-переключатель *switch*

Оператор-переключатель `switch` может заменить множество условных операторов. Он, собственно, и представляет собой множество операторов вида `if-else`, объединенных в один.

```
switch ({Выражение}) {
  case {Значение 1} :
    {Блок 1}
    [break;]
  [case {Значение 2} :
    {Блок 2}
    [break;]]
  ... Другие секции case
  [default :
    {Блок, исполняемый для остальных значений}]
}
```

Сначала вычисляется выражение, помещенное в скобки после ключевого слова `switch`. После этого его значение сравнивается со значениями, находящимися после ключевых слов `case`, и в случае равенства исполняется блок кода из соответствующей секции. Например, если значение выражения равно "значению 1", выполнится блок кода 1 и т. д. Если же оно не встретилось ни в одной секции `case`, выполняется блок кода, находящийся в секции `default`.

Чтобы проиллюстрировать вышесказанное, рассмотрим следующий пример.

```
switch (a) {
  case 1 :
    out = "Единица";
    break;
  case 2 :
    out = "Двойка";
```

```
    break;
  case 3 :
    out = "Тройка";
    break;
  default :
    out = "Другое число";
}
```

Здесь, если значение a равно единице, переменной out присваивается значение "Единица". Аналогично — для двойки и тройки. Если a имеет другое значение, отличное от единицы, двойки или тройки, исполняется блок кода секции default, и переменная out получает значение "Другое число".

Секция default может отсутствовать; в таком случае для значений, не перечисленный в секциях case, не будет исполняться никакой код. Также можно (теоретически) опустить ключевое слово break, но в этом случае будет выполняться весь код от секции, где встретилось нужное значение, вплоть до конца оператора switch, что не всегда нужно. Изменим наш пример, чтобы проиллюстрировать это.

```
switch (a) {
  case 1 :
    out = "Единица";
  case 2 :
    out = "Двойка";
  case 3 :
    out = "Тройка";
  default :
    out = "Другое число";
}
```

В этом случае для любого значения a переменная out всегда будет содержать строку "Другое число". Это произойдет потому, что будут выполнены все блоки, начиная от блока соответствующей секции до самого конца оператора switch, т. е. до секции default, которая присвоит переменной out значение "Другая строка".

Примечание

Оператор switch начал поддерживаться в Internet Explorer и Navigator только с версий 4.0.

Циклы

Циклы — это блоки кода, выполняемые несколько раз. Повторение цикла прерывается по наступлению некоего условия. (Есть, правда, бесконечные циклы, но необходимость в них возникает довольно редко.) Здесь напраши-

вается аналогия с условными операторами, когда при наступлении некоего условия исполняется или не исполняется какой-либо блок кода.

JavaScript предлагает программистам несколько разновидностей циклов. Ниже мы опишем их все.

Цикл *for*

Цикл `for` выполняется, пока некое условие остается истинным (возвращает `true`). Как правило, он используется для того, чтобы выполнить некий фрагмент кода определенное число раз. В этом случае целое число, подсчитывающее, сколько раз был выполнен этот фрагмент кода, помещается в переменную, называемую *счетчиком цикла*. А блок кода, который выполняется в цикле, является *телом цикла*.

```
for ({Выражение инициализации}; {Условие}; {Выражение инкремента})
  . . . Тело цикла
```

Выражение инициализации, как правило, присваивает переменной-счетчику начальное значение. Далее проверяется условие, и, если его значение истинно, тело цикла выполняется. После этого выражение инкремента изменяет значение счетчика, снова проверяется условие, и так до тех пор, пока условие не станет ложным (`false`). Такие циклы со счетчиком встречаются в программах очень часто.

```
for (i = 1; i < 11; i++) {
  a += 3;
  b = i * 2 + 1;
}
```

Этот цикл будет выполнен 10 раз. Обратите внимание на выражение инициализации и условие. Мы присваиваем счетчику `i` начальное значение 1 и после каждого выполнения тела цикла увеличиваем его на единицу. Цикл перестанет выполняться, когда значение счетчика увеличится до 11, и условие станет ложным.

Заметьте также, что мы использовали счетчик цикла в одном из выражений тела цикла — это вполне допустимо. Счетчик `i` будет содержать последовательно возрастающие значения от 1 до 10, которые можно использовать в вычислениях. (Как и происходит в рассмотренном примере.)

Значение счетчика цикла совсем не обязательно именно инкрементировать. Можно декрементировать его или вообще изменять на значения, отличные от единицы. Чтобы сделать это, просто впишите соответствующее выражение в оператор цикла. В двух следующих примерах это проиллюстрировано:

```
for (i = 10; i > 0; i--) {
  a += 3;
```

```
  b = i * 2 + 1;
}
for (i = 2; i < 21; i + =2) b = i * 2 + 1;
```

В первом примере значение счетчика декрементируется. Начальное его значение равно 10. Цикл выполнится 10 раз и завершится, когда счетчик будет содержать 0; при этом значения последнего будут последовательно уменьшаться от 10 до 1.

Во втором примере начальное значение счетчика равно 2, а конечное — 21; цикл выполнится опять же 10 раз. А все потому, что значение счетчика увеличивается на 2 и последовательно принимает значения 2, 4, 6... 20.

В особом вырожденном случае цикл `for` может даже не содержать тела. В этом случае "полезную нагрузку" цикла несет на себе выражение инкремента.

Оператор , (запятая)

JavaScript предлагает оператор, используемый в основном, в выражениях инкремента циклов `for`. Это оператор , (запятая). Он позволяет выполнять то одно, то другое выражение инкремента последовательно. Его формат таков:

```
{"Нечетное выражение"}, {"Четное выражение"}
```

При первом прохождении цикла выполняется "нечетное" выражение, при втором — "четное", при третьем — снова "нечетное" и т. д.

```
for (i = 0; j < 11; i++, j++) {
  a = i * 2 + 1;
  b = j / 2 - 3;
}
```

Оператор , (запятая) имеет самый низкий приоритет из всех операторов.

Цикл *do-while*

Цикл `do-while` во многом похож на цикл `for`, но в нем не используется счетчик. Цикл `do-while` выполняется до тех пор, пока остается истинным условие. Причем условие проверяется не до, а после выполнения тела цикла, так что цикл `do-while` выполнится хотя бы один раз, даже если условие с самого начала ложно.

```
do
   . . . Тело цикла
while ({Условие});
```

Вы можете использовать цикл `do-while` различными способами.

```
do {
  a = a * i + 2;
```

```
  i = ++i;
} while (a < 100);
```

В рассмотренном выше примере проверяется наступление отвлеченного условия.

```
var a = 0, i = 1;
do {
  a = a * i + 2;
  i = ++i;
} while (i < 20);
```

А здесь мы уже используем счетчик, чье конечное значение ограничено. Конечно, в таких случаях удобнее использовать цикл `for`.

Примечание

Цикл `do-while` начал поддерживаться в Internet Explorer и Navigator только с версий 4.0.

Цикл *while*

Цикл `while` напоминает своего родственника `do-while`, но условие проверяется перед выполнением тела цикла. Так что, если оно изначально ложно, цикл не выполнится ни разу.

```
while ({Условие})
   . . . Тело цикла
```

Я не буду подробно рассматривать этот вид циклов. Приведу лишь пример, аналогичный уже рассмотренному выше в связи с циклом `do-while`:

```
while (a < 100) {
  a = a * i + 2;
  i = ++i;
  }
```

Операторы *break* и *continue*

Иногда бывает нужно прервать выполнение цикла. Для этого JavaScript предоставляет программистам операторы `break` и `continue`.

Оператор `break` позволяет прервать выполнение цикла и перейти к следующему за ним выражению.

```
while (a < 100) {
  a = a * i + 2;
  if (a > 50) break;
  i = ++i;
  }
```

В этом примере мы прерываем выполнение цикла (но не программы), если значение переменной a превысит 50.

Оператор continue позволяет *перезапустить* цикл, т. е. оставить невыполненными все последующие команды, входящие в тело цикла, и вернуться к его началу. Будет протестировано условие (цикл for, кроме того, выполнит выражение инкремента), и, если условие истинно, тело цикла начнет выполняться с начала.

```
while (a < 100) {
  i = ++i;
  if (i > 9 && i < 11) continue;
  a = a * i + 2;
  }
```

Здесь мы пропускаем выражение вычисления a для всех значений i от 10 до 20.

Полный формат операторов break и continue выглядит так.

```
break|continue [{Метка}]
```

Здесь *метка* помечает цикл, к которому применяется данный оператор break или continue. Формат метки таков.

```
{Имя метки}: . . . Выражение
```

Имя метки должно быть уникально в пределах программы. Для имени метки действуют те же правила, что и для имен переменных.

Пример использования меток приведен ниже. Здесь мы рассмотрим так называемые *вложенные* циклы, т. е. циклы, находящиеся один внутри (в теле) другого. Причем, внутренний цикл будет выполняться целиком на каждом шаге внешнего.

```
Outer:
for (i = 0; i < 5; i++) {
  Inner:
  for (j = 0; j < 5; j++) {
    if (j == 2)
      continue Outer;
    else
      a[i,j] = j + 1;
    }
  }
```

Здесь при равенстве значения переменной j двум в условном выражении, расположенном в теле внутреннего цикла, мы перезапускаем внешний цикл, помеченный как Outer. Если бы мы опустили имя метки в операторе continue, то перезапустился бы внутренний цикл Inner.

Примечание

Поддержка меток появилась в Internet Explorer и Navigator только с версий 4.0. В предыдущих версиях этих программ операторы `break` и `continue` использовались без указания метки.

Комментарии

Изучая HTML, вы уже, должно быть, знаете, что частенько возникает необходимость встроить в исходный код Web-страницы пояснения для коллег или самого себя, так называемые комментарии. Причем, эти пояснения не должны никоим образом отображаться на экране или влиять на работу программы Web-обозревателя.

JavaScript тоже не отстает. Он предоставляет в ваше распоряжение два оператора вставки комментариев.

```
// . . . Строка комментария
```

Этот оператор позволяет вставить в конец выражения однострочный комментарий.

```
a = b + c; // Это однострочный комментарий
```

Заметьте, что комментарий находится после точки с запятой, обозначающей конец выражения.

```
/*
. . . Комментарий
*/
```

А этот оператор позволяет вставить в код программы комментарий любого размера.

```
/*
В этом выражении мы складываем содержимое двух переменных
и помещаем результат в третью
*/
a = b + c;
```

Функции

Понятие функции

Функция — это специально написанный и оформленный фрагмент кода JavaScript, который можно вызвать из любого места программы. Функция имеет уникальное имя, возможно, принимает один или несколько параметров и возвращает результат, который допускается использовать в программе.

Как правило, в виде функции оформляется какой-либо часто используемый фрагмент кода; вместо того, чтобы копировать его в разные места программы, лучше написать функцию и вызывать ее, когда нужно, с разными параметрами. Такой фрагмент кода, помещенный в функцию, называется *телом функции.*

Прежде чем функция будет использована, ее необходимо объявить. Это делается с помощью оператора `function`.

```
function {Имя функции}([{Список аргументов, разделенных запятыми}])
  . . . Тело функции
```

Имя функции, как я сказал, должно быть уникально в пределах программы. Для имен функций действуют те же правила, что и для имен переменных и меток.

Список аргументов представляет собой набор переменных, в которые при вызове функций будут помещены нужные аргументы. Вы можете придумать для этих переменных любые имена — все равно они будут использованы только внутри тела функции. Это так называемые формальные аргументы функции.

Имена аргументов функции помещаются в круглые скобки и разделяются запятыми. В особом случае список аргументов может отсутствовать, но скобки при этом должны оставаться. Некоторые функции не требуют скобок; о таких случаях мы поговорим особо (собственно, это не совсем функции).

Тело функции, как вы уже поняли, выполняет какие-то действия над аргументами и получает результат. Чтобы вернуть его из функции в место вызова, используется оператор `return`:

```
return {Переменная или выражение};
```

Здесь переменная должна содержать возвращаемое значение, а выражение должно его вычислять.

Чтобы вызвать функцию, используется следующий формат:

```
{Имя функции}([{Список фактических аргументов, разделенных запятыми}])
```

Здесь указывается имя функции (помните, что оно должно быть уникально), и в круглых скобках перечисляются ФАКТИЧЕСКИЕ аргументы, над которыми нужно выполнить соответствующие действия, а не формальные, использованные в объявлении функции. Функция вернет результат, который можно присвоить переменной или использовать в сложном выражении.

А теперь пора рассмотреть пример.

```
function samplefunc(a, b) {
  var c;
```

```
  c = (a + b) / 2;
  return c;
  }
. . .
var arg1=1, arg2=2, result;
result = samplefunc(arg1, arg2);
```

Здесь мы определили функцию по имени `samplefunc`, возвращающую половину суммы двух чисел. Обратите внимание на то, как мы назвали аргументы в определении функции. После этого мы вызвали эту функцию в коде программы и передали ей два фактических аргумента. Запомните эту разницу: формальные аргументы в объявлении функции не имеют ничего общего с фактическими аргументами, передаваемыми в функцию программой. Формальные аргументы используются только внутри объявления функции.

Результат нашей функции был возвращен оператором `return`. Это понятно.

Функции можно вызывать друг из друга. Только имейте в виду, что прежде чем функция будет вызвана, она должна быть определена.

```
function samplefunc1(a, b) {
  . . .
  }
function samplefunc2(c) {
  . . .
  k = y + samplefunc1(x, 2);
  }
```

А теперь — небольшой фокус:

```
type = typeof(samplefunc);
```

Да, это выражение абсолютно правильно с точки зрения JavaScript. Оператор `typeof` вернет для функции `samplefunc` строку `"function"`.

Вы также можете присвоить функцию переменной.

```
varfunc = somefunction;
```

И потом использовать эту переменную для вызова функции.

```
a = varfunc(b, c);
```

Заметьте, что переменная `varfunc` также будет иметь тип `"function"`.

Глобальные и локальные переменные

Внутри нашей функции `samplefunc` была объявлена переменная `c`, которая впоследствии использовалась в вычислениях для хранения результата. Но вдруг, скажете вы, в программе уже объявлена переменная с таким именем? Не возникнет ли по этой причине конфликт?

Никакого конфликта не возникнет. А все потому, что объявленная в функции переменная не будет видима в программе.

Мы подошли к понятию глобальных и локальных переменных.

Глобальными являются переменные, объявленные в коде программы на внешнем уровне, т. е. вне определений функций. Они будут видимы отовсюду; из любого участка программы, в том числе из определений всех функций можно без проблем обратиться к глобальной переменной, получить или изменить ее значение.

Локальными являются переменные, определенные внутри тела функции. Они будут видимы только в пределах тела функции, но никак не вне его. Остальной код программы не будет ничего знать об их существовании. Можно считать, что локальные переменные создаются при вызове функции и уничтожаются, когда интерпретатор завершает ее выполнение и возвращается в программу.

Вы даже можете определить локальную переменную с тем же именем, что и уже имеющаяся в основной программе глобальная. При этом интерпретатор создаст локальную переменную с заданным именем, но к ее глобальной "тезке" вы уже доступа иметь не будете. Иногда так делают специально, иногда такой казус возникает вследствие ошибки программиста; просто имейте это в виду, когда будете писать свои функции.

И еще. Фактически определить переменную с помощью оператора `var` можно в любом месте функции, но реально она будет создана в самом ее начале, перед тем, как будет выполнен первый оператор тела функции. Это своего рода "защита от дурака", которую предусматривает интерпретатор JavaScript.

Рекурсия

Рассмотрим еще один вопрос, связанный с написанием функций.

Вы уже знаете, что функции могут вызывать другие функции, конечно, если те определены. Более того, функции могут вызывать и сами себя. Такой прием программирования иногда может быть очень полезен и называется *рекурсией* или *рекурсивным вызовом* функции.

В обычных условиях, если функция вызовет сама себя, то она войдет в бесконечный цикл вызовов, из которого нет выхода (так называемая *бесконечная рекурсия*). В конце концов, это завершится аварийной остановкой программы, а то и крахом операционной системы. Поэтому функция, предназначенная для рекурсивного вызова, должна предусматривать возможность выхода из цикла вызовов. Стандартного способа сделать это не существует, решение для каждого конкретного случая нужно искать особо.

Рассмотрим пример.

```
function factorial(a)  {
  if (a == 0) {
    return 1;
    }
  else
    return (a * factorial(a - 1));
  }
```

Эта функция вычисляет факториал числа `a`. Она вызывает сама себя для того, чтобы получить факториал числа `a - 1`. Заметьте также, что в теле функции выполняется проверка условия равенства `a` нулю; если это условие истинно, возвращается единица, и дальнейших рекурсивных вызовов не происходит. Таким образом, данная функция имеет защиту от бесконечной рекурсии, ведь когда-нибудь она все равно получит в качестве аргумента ноль.

Приведем пример другой функции.

```
function endless(a, b)  {
  var c;
  c = endless(a, b);
  }
```

Эта функция будет вызывать себя бесконечно и в конце концов "обрушит" программу. А все потому, что в ней нет условия, которое прерывало бы бесконечный цикл вызовов.

Встроенные функции JavaScript

Мы рассмотрели в общих чертах, как можно писать свои функции, узнали об их преимуществах и "подводных камнях", на которые часто наталкиваются начинающие программисты. О функциях осталось рассказать совсем немного.

Дело в том, что JavaScript определяет свои собственные функции, которые можно использовать в программах. Они называются *встроенными*, т. к. объявлены и реализованы внутри интерпретатора языка. Этих функций немного, но знать их полезно.

Все встроенные функции перечислены в табл. 2.9.

Примечание

Функции `infinity`, `isFinite`, `NaN` начали поддерживаться Internet Explorer только с версии 4.0 и Navigator с версии 4.06. Функция `undefined` начала поддерживаться Internet Explorer только с версии 5.5 и Navigator с версии 4.06. В Navigator версии 2.0 функции `parseFloat` и `parseInt` на платформах Solaris и Irix возвращали `NaN`, а на всех остальных — пустую строку.

Таблица 2.9. *Встроенные функции языка JavaScript*

Функция	Описание	Пример
`escape({Строка})`	Кодирует строку так, чтобы она выглядела как URL. То есть, все недопустимые в URL символы ("*", "@", "_", "+", "/", пробел) будут представлены их шестнадцатеричными кодами	`escape("&")` вернет "%26"
`eval({Строка})`	Вычисляет выражение, находящееся в строковой переменной, как если бы оно было написано в коде программы. В выражении можно использовать переменные, функции, любые операторы JavaScript	`eval("2+2")` вернет 4
`infinity`	Возвращает значение "плюс бесконечность". Служит для математических расчетов. Не принимает аргументов и не требует скобок	
`isFinite({Выражение})`	Проверяет, возвращает ли выражение конечное число. Результат: `true` или `false`	
`isNaN({Выражение})`	Проверяет, возвращает ли выражение правильное число. Результат: `true`, если не возвращает (бесконечность или ошибка, иначе говоря, *NaN* (Not a Number — не число)), или `false`, если возвращает	
`NaN`	Возвращает значение `NaN` (Not a Number — не число). Не принимает аргументов и не требует скобок	
`parseFloat({Строка})`	Преобразует строку в число с плавающей точкой. Если строка не может быть преобразована, возвращает `NaN`	`parseFloat("1.2")` вернет 1.2
`parseInt({Строка}[,{Основание}])`	Преобразует строку в целое число системы счисления, определяемой основанием. В частности, основание 10 определяет десятичную систему, 8 — восьмеричную, 16 — шестнадцатеричную. По умолчанию берется десятичная система. Если строка не может быть преобразована, возвращает `NaN`	`parseInt("FF",16)` вернет 255 `parseInt("HT45")` вернет `NaN`

Таблица 2.9 (окончание)

Функция	Описание	Пример
undefined	Возвращает значение undefined, обозначающее, что переменная не определена, хоть и была объявлена. Может быть использована в выражениях сравнения. Не принимает аргументов и не требует скобок	
unescape({Строка})	Декодирует строку, закодированную функцией escape	escape("%26") вернет "&"

Оператор *void*

Вы уже знаете, что функция должна возвращать значение. Но иногда это значение не нужно. Как тогда поступить?

Первый и самый очевидный выход: объявить какую-либо переменную, присвоить ей возвращаемое значение и забыть. Или использовать эту переменную впоследствии по другому назначению.

```
var dummy;
dummy = someFunction();
```

Но можно также использовать оператор void. Этот оператор просто "проглатывает" значение, возвращаемое функцией.

```
void someFunction();
```

Примечание

Оператор void начал поддерживаться Internet Explorer 4.0 и Navigator только с версий 3.0.

Массивы

Массив — это пронумерованный набор переменных одного типа, называемых *элементами* массива. При этом номер элемента массива именуется *индексом*, а общее число элементов — *размером* массива.

Для определения массива используется специальный литерал массива, представляющий собой список значений элементов, разделенных запятыми, взятый в квадратные скобки.

```
var massive;
massive = [1, 2, 3, 4];
```

Здесь мы определили массив, содержащий четыре элемента, и присвоили его переменной massive. После этого мы можем получить доступ к любому из элементов по его индексу, указав его после имени переменной массива в квадратных скобках.

```
a = massive[2];
```

В этом выражении мы присвоили переменной a значение третьего элемента массива (нумерация элементов массива начинается с нуля).

Мы можем не определять некоторые элементы массива.

```
massive2 = [1, 2,, 4];
```

Обратите внимание на то, что мы пропустили третий элемент, и он остался неопределенным.

Если будет нужно, мы легко сможем добавить к массиву еще один элемент, просто присвоив ему нужное значение. Вот так:

```
massive[4] = 9;
```

При этом будет создан новый элемент массива с индексом 4 и значением 9. (Он будет пятым по счету.)

Можно даже сделать так:

```
massive[7] = 9;
```

В данном случае будут созданы четыре новых элемента, и восьмой элемент получит значение 9. Пятый, шестой и седьмой останутся неопределенными.

Вы можете присвоить любому элементу массива другой массив (так называемый *внутренний* массив).

```
massive2[2] = ["n1", "n2", "n3"];
```

После этого вы можете получить доступ к любому элементу внутреннего массива, указав оба индекса: внешнего и внутреннего массивов.

```
str = massive2[2][1];
```

Переменная str будет содержать строку "n2".

Оператор typeof возвращает для массива значение "object".

Объекты

Итак, мы изучили базовые основы языка программирования JavaScript. Теперь пора переходить к более сложным темам, вплотную связанным с Web-программированием. Это, прежде всего, объекты. Все, что мы уже изучили — просто прелюдия или, если хотите, присказка. Сказка будет впереди...

Итак, что же такое объекты.

Понятие объекта

Объект — это сложный тип данных, включающий в себя множество переменных — *свойств* — и набор функций для манипуляции этими переменными — *методов*. Свойства хранят данные, а методы их обрабатывают. Таким образом, объект можно рассматривать как отдельный, независимый от других фрагмент программы, выполняющий довольно сложные действия.

В качестве примеров объектов рассмотрим HTML-документ. Он имеет набор свойств: интернет-адрес, размер, кодировка символов (если он написан на русском языке). Некоторые свойства вы можете изменить, другие — нет. Также он имеет набор методов: показать в окне Web-обозревателя, сохранить на диске, распечатать. Вы просто вызываете нужные методы, не заботясь о том, как объект HTML-документа все это выполнит: подразумевается, что он знает, как это сделать. Во введении я привел такую аналогию: свойства — это атрибуты объекта, а методы — рычаги управления.

Объект отличается от других типов данных еще и тем, что для использования его нужно создать *экземпляр* соответствующего *класса* объекта. При этом класс — это своего рода тип объекта, аналогичный типу переменной (не путать с классами стилей, рассмотренными нами в предыдущем разделе), а экземпляр — конкретный объект, с которым можно работать. Создается экземпляр объекта с помощью оператора `new`; при этом ссылка на объект присваивается переменной. *Ссылка* (или *указатель*) отмечает место в памяти, где находится созданный экземпляр объекта. Используя ссылку, хранящуюся в переменной, мы можем обращаться к объекту.

```
var obj;
obj = new SomeObject();
```

Здесь `obj` — переменная, которой будет присвоена ссылка на созданный экземпляр объекта, а `SomeObject` — класс создаваемого объекта.

После использования объекта его можно удалить. Для этого воспользуйтесь оператором `delete`.

```
delete obj;
```

Внимание!

Оператор `delete` поддерживается Internet Explorer и Navigator только с версии 4.0. В Internet Explorer и Navigator 3.0 вы можете удалять объекты, присваивая переменным значение `null`. Более старые версии Navigator вообще не позволяют удалять объекты.

Хорошо, мы создали объект и хотим получить доступ к его свойствам и методам. Как это делается? Очень просто! К свойствам вы можете обращаться, используя следующий формат вызова:

```
{Имя переменной, хранящей ссылку на объект}.{Имя свойства}
```

Таким образом, обращение к свойству аналогично обращению к переменной, только перед именем переменной-свойства необходимо поставить имя переменной-объекта, отделенное точкой. Вы ведь помните, что свойство — часть объекта.

```
obj.SomeProperty = 9;
encoding = HTMLDocument.CodePage;
```

Метод вызывается почти так же:

```
{ Имя переменной, хранящей ссылку на объект }.{Имя метода}
↳([{Параметры}])
```

То есть, почти как обычная функция, за исключением обязательного указания имени объекта.

```
void HTMLDocument.SaveToDisk('somefile.htm');
```

Объект можно присвоить элементу массива и передать в функцию. При этом внутри тела функции разрешается изменять его свойства и вызывать методы.

Объект может также иметь свойства типа объекта, т. е. свойства, хранящие ссылки на так называемые *внутренние объекты*. Разрешается обращаться к свойствам и методам внутренних объектов, используя аналогичный прием:

```
encoding = someObject.internalObject1.property1;
v = someObject.internalObject1.method1;
```

Как видите, мы указываем сначала имя объекта, потом имя внутреннего объекта, а потом уже имя свойства или метода этого самого внутреннего объекта.

Кстати, вы не заметили, как мы создали новый объект? На всякий случай приведу эту строку еще раз.

```
obj = new SomeObject();
```

Она удивительно похожа на вызов функции, не так ли? На самом деле, это и есть функция, специальный метод объекта, называемый *конструктором*. От других методов он отличается тем, что вызывается при создании объекта и устанавливает начальные значения его свойств. Имя конструктора всегда совпадает с именем класса объекта.

Конструктор объекта может принимать параметры.

```
obj2 = new SomeObject2(param1, param2, x + y, 3);
```

Существуют также объекты без конструктора. Они создаются с использованием так называемых *инициализаторов*.

```
obj3 = new SomeObject3(prop1: param1, prop2: param2, prop3: x + y,
↳prop4: 3);
```

То есть при создании такого объекта мы заполняет значениями его свойства.

Примечание

Инициализаторы поддерживаются Internet Explorer и Navigator начиная с версии 3.0. В более старых версиях Navigator (Internet Explorer поддерживает JavaScript начиная с версии 3.0) вы можете создавать объекты только с помощью конструкторов.

И еще. Все объекты, доступные вам в JavaScript, подразделяются на *встроенные*, созданные пользователем *(пользовательские)* и предоставляемые другими программами *(внешние)*. К последним, в частности, относятся объекты, образующие DOM, т. к. Web-обозреватель считается внешней программой по отношению к интерпретатору JavaScript. Внешние объекты мы рассмотрим в следующих главах книги.

Оператор `typeof` возвращает для переменных объектов строку `"object"`.

Объекты и массивы

Мы познакомились с объектами, а чуть ранее — с массивами. Теперь самое время соединить эти два типа данных вместе.

Дело в том, что JavaScript поддерживает обращение к объектам как к массивам. Это означает, что доступ к свойствам объекта вы можете получить не только с использованием описанного выше формата, но и указав индекс свойства, словно это элемент массива. Ведь в некоторых случаях объекты JavaScript можно рассматривать как массивы.

```
obj.SomeProperty = 9;
obj[3] = 9;
```

Эти два выражения присваивают одному и тому же свойству объекта значение 9. Только первое использует обычный формат доступа к свойствам, а второе обращается к свойству как к элементу массива. (Естественно, в последнем случае нужно знать числовой индекс этого свойства.)

Для чего может понадобиться такой способ доступа к свойствам? Понятия не имею! По-моему, гораздо полезнее третий способ доступа — как к элементам ассоциативного массива.

В ассоциативном массиве, в отличие от обычного, "классического", элементы имеют не числовой, а строковый индекс. То есть, можно считать, что каждый элемент имеет уникальное имя, совсем как переменная.

```
massive["someElement"] = 4;
```

Так вот. Каждый объект JavaScript представляет собой как раз такой вот ассоциативный массив, в котором индексом каждого свойства является его именем.

```
obj["SomeProperty"] = 9;
```

Сейчас знание этой особенности может показаться вам бесполезным. Но подождите немного до того момента, как мы начнем рассматривать объекты, создаваемые пользователем. Вот там без этого не обойтись.

Несколько новых операторов

Теперь самое время рассмотреть несколько новых операторов, применяемых при работе с объектами.

Оператор `in` проверяет, существует ли у заданного объекта заданное свойство, и возвращает значение `true` или `false`.

```
{Имя свойства} in {Объект}
```

Здесь `Имя свойства` задается в виде строки, а `Объект` — в виде переменной объектного типа.

Использовать этот оператор можно, например, так:

```
if ("SomeProperty" in obj) . . .
```

Примечание

Оператор `in` поддерживается только Internet Explorer.

Оператор `instanceof` проверяет, является ли заданный объект экземпляром заданного класса, и возвращает значение `true` или `false`.

```
{Объект} instanceof {Класс}
```

Здесь `Объект` задается в виде переменной объектного типа, а `Класс` — в виде имени класса.

```
if (obj instanceof SomeObject) . . .
```

Примечание

Оператор `instanceof` начал поддерживаться Internet Explorer 5.5.

Оператор `for-in` позволяет просмотреть все свойства заданного объекта или массива и выполнить для каждого какие-либо действия. Вообще-то, это оператор цикла, но довольно специфичный.

```
for ({Счетчик} in {Объект или массив})
  . . . Тело цикла
```

Счетчик можно использовать для доступа к значениям свойств объекта или элементов массива.

```
for (k in obj) {
  s += k + ' ';
  st += obj[k] + ' ';
  }
```

Этот фрагмент программы просматривает все свойства объекта obj и помещает в переменную s имена свойств, разделенные пробелами, а в st — их значения, также разделенные пробелами.

Примечание

Оператор for-in начал поддерживаться Internet Explorer только с версии 5.5.

Оператор with позволяет значительно сократить длину выражений JavaScript, если в них используются свойства или методы какого-либо объекта (заметьте: одного объекта). Я не буду приводить его формат, а сразу покажу пример:

```
someobject.property1 = 1;
someobject.property2 = 2;
someobject.property3 = 3;
someobject.method1;
```

Это у нас фрагмент программы, где не использовался оператор with. Видите, какие длинные строки выражений. Теперь используем оператор with...

```
with (someobject){
  property1 = 1;
  property2 = 2;
  property3 = 3;
  method1;
  }
```

Фрагмент программы сразу стал компактнее. И быстрее, кстати говоря.

Встроенные классы объектов JavaScript

Еще одной особенностью JavaScript является то, что он может рассматривать обычные, простые типы данных (числовой, строковый, логический) как объекты. Точнее, классы объектов. Большинство встроенных классов как раз и являются объектными представлениями простых типов данных. Давайте рассмотрим их.

Класс массива *Array*

Массивы JavaScript можно создавать как объекты. При этом конструктору передается один или несколько параметров. Если передается один параметр числового типа, создается массив, содержащий соответствующее количество элементов, естественно, не определенных. Если передается несколько параметров или один параметр несовместимого типа, эти параметры используются как элементы массива. Если вы вообще не зададите параметров в конструкторе, создается массив нулевой длины, т. е. совсем без элементов.

```
arr = new Array(10);
arr = new Array(1, 2, 3, 4);
```

Вы можете использовать свойство `length` для получения размера массива. И, конечно же, индексы — для доступа к элементам массива.

Все методы класса массива перечислены в табл. 2.10.

Таблица 2.10. *Методы класса массива* `Array`

Метод	Описание	Пример

`concat({Список добавляемых элементов})`	Возвращает массив, получившийся в результате объединения текущего массива и элементов, перечисленных в списке. Список может содержать другие массивы	`a = [1, 2, 3];`
		`b = [4, 5];`
		`c = a.concat(b, 6);`
		`c` содержит `[1, 2, 3, 4, 5, 6]`

`join({Разделитель})`	Возвращает строку, получившуюся в результате слияния значений всех элементов, разделенных разделителем	
`pop()`	Удаляет последний элемент массива и возвращает его. Если массив пуст, возвращается `undefined`	
`push({Список добавляемых элементов})`	Добавляет в массив элементы, перечисленные в списке, и возвращает новую длину массива. Список может содержать другие массивы	

`reverse()`	Возвращает массив, порядок элементов которого изменен на противоположный	`a = [1, 2, 3];`
		`b = a.reverse();`
		`b` содержит `[3, 2, 1]`

`shift()`	Удаляет первый элемент массива и возвращает его	

`slice({Индекс первого элемента}, [{Индекс последнего элемента}])`	Возвращает массив, образованный из элементов текущего массива, от первого элемента включительно до последнего исключительно. Если индекс последнего элемента пропущен, выбираются все элементы до конца массива	`a = [1, 2, 3, 4];`
		`b = a.slice(1, 3);`
		`b` содержит `[2, 3]`

Таблица 2.10 (окончание)

Метод	Описание	Пример
`sort([{Функция сортировки}])`	Возвращает массив, заполненный отсортированными элементами текущего массива. Функция сортировки должна принимать два параметра и возвращать одно из следующих значений: 1, если первый больше второго, –1, если второй больше первого, и 0, если они одинаковы. Если функция сортировки опущена, выполняется символьная сортировка	
`splice({Индекс первого элемента}, {Количество удаляемых элементов}, [{Список добавляемых элементов, разделенных запятыми}])`	Удаляет из массива заданное количество элементов и вставляет на их место новые из списка, если они определены. Возвращает массив, состоящий из удаленных элементов	

`toSource()`	Возвращает строку, представляющую исходный код массива	`a = [1, 2, 3];`
		`s = a.toSource();`
		`s` содержит
		`"[1, 2, 3]"`

`toString()`	Возвращает строку, представляющую исходный код массива Аналогичен `toSource()`	

`unshift({Список добавляемых элементов})`	Возвращает массив, получившийся в результате объединения текущего массива и элементов, перечисленных в списке, причем элементы вставляются в начало текущего массива. Список может содержать другие массивы
	Аналогичен `join(",")`
		`a = [1, 2, 3];`
		`b = [4, 5];`
		`c = a.unshift(b, 6);`
		`c` содержит `[4, 5, 6, 1, 2, 3]`

Примечание

Класс `Array` поддерживается Internet Explorer начиная с версии 4.0 и Navigator — с версии 3.0. Методы `concat` и `slice` включены в Navigator начиная с версии 4.0. Методы `pop`, `push`, `shift`, `splice` и `unshift` поддерживаются Internet Explorer начиная с версии 5.5 и Navigator — с версии 4.0. Метод `toSource` разрешен только в Navigator начиная с версии 4.06. В Navigator 3.0 на некоторых платформах метод `sort` может не работать. В Navigator 4.0 метод `splice`, если удален один элемент, возвращает именно этот элемент, а не массив из одного элемента.

Класс логической величины *Boolean*

Логические величины также могут быть представлены в виде объектов. За это отвечает класс `Boolean`.

Конструктор класса `Boolean` принимает логическое выражение в виде параметра. Если параметр пропущен, начальное значение равно `true`.

```
bool = new Boolean(f == g - 1);
```

Класс `Boolean` поддерживает три метода. Метод `toSource` возвращает строку, представляющую исходный код класса `Boolean`. Метод `toString` возвращает, в зависимости от значения объекта, `"true"` или `"false"` в строковом виде. Метод `valueOf` — значение объекта: `true` или `false`.

Примечание

Класс `Boolean` включен в Internet Explorer начиная с версии 4.0 и Navigator — с версии 3.0. Метод `toSource` поддерживается только Navigator начиная с версии 4.06.

Класс даты *Date*

Класс даты служит для хранения значений даты и времени.

Конструктор этого класса принимает значение даты в числовом или строковом формате. Если параметр представлен в числовом формате, он трактуется как число миллисекунд, прошедших с полуночи 1 января 1970 года. Если он представлен в строковом формате, конструктор пытается преобразовать его в значение даты по следующим правилам:

- строка, имеющая формат "месяц/число/год" или "месяц.число.год" (например, `12.07.98`), преобразуется в дату;
- строка, имеющая формат "месяц число год" (например, `Июль 10 1998`), преобразуется в дату;
- любой прочий текст воспринимается как комментарий;
- строка, имеющая формат "часы:минуты:секунды" (например, `10:04:57`), преобразуется во время;
- строка, имеющая формат "часы:минуты [PM]" (например, `10:04 PM`), преобразуется во время.

Конструктор объекта даты может иметь и такой формат:

```
Date({Год}, {Месяц}, {Число}[, {Часы}[, {Минуты}[, {Секунды}
↳[,{Миллисекунды}]]]])
```

Если же ни один параметр не указан, конструктор инициализирует объект текущей датой.

Все методы класса даты перечислены в табл. 2.11.

Таблица 2.11. *Методы класса даты* `Date`

Метод	Описание
`getDate()`	Возвращает число
`getDay()`	Возвращает цифру, обозначающую день недели (0 — воскресенье, 1 — понедельник, 2 — вторник и т. д.)
`getFullYear()`	Возвращает год
`getHours()`	Возвращает час
`getMilliseconds()`	Возвращает миллисекунды
`getMinutes()`	Возвращает минуты
`getMonth()`	Возвращает цифру, обозначающую месяц (от 0 до 11)
`getSeconds()`	Возвращает секунды
`getTime()`	Возвращает время в виде количества миллисекунд, прошедших с полуночи 1 января 1970 года
`getTimezoneOffset()`	Возвращает различие в минутах между локальным и универсальным временем
`getUTCDate()`	Возвращает число по универсальному времени
`getUTCDay()`	Возвращает цифру, обозначающую день недели, по универсальному времени
`getUTCFullYear()`	Возвращает год по универсальному времени
`getUTCHours()`	Возвращает час по универсальному времени
`getUTCMilliseconds()`	Возвращает миллисекунды по универсальному времени
`getUTCMinutes()`	Возвращает минуты по универсальному времени
`getUTCMonth()`	Возвращает цифру, обозначающую месяц, по универсальному времени
`getUTCSeconds()`	Возвращает секунды по универсальному времени
`getVarDate()`	Возвращает двоичное значение даты, используемое в элементах ActiveX

Таблица 2.11 *(продолжение)*

Метод	Описание
`getYear()`	Возвращает год. Предусмотрен для совместимости; вместо него рекомендуется использовать метод `getFullYear`
`parse({Дата в строковом виде})`	Расшифровывает строку согласно правилам, описанным в начале этой главы, и возвращает количество миллисекунд, прошедших между полученной датой и полуночью 1 января 1970 года
`setDate({Число})`	Устанавливает число
`setFullYear({Год}[, {Месяц} [, {Число}]])`	Устанавливает год, а также месяц и число, если они заданы
`setHours({Часы}[, {Минуты} [, {Секунды}[, {Миллисекунды}]]])`	Устанавливает часы, а также минуты, секунды и миллисекунды, если они заданы
`setMilliseconds({Миллисекунды})`	Устанавливает миллисекунды
`setMinutes({Минуты}[, {Секунды} [, {Миллисекунды}]])`	Устанавливает минуты, а также секунды и миллисекунды, если они заданы
`setMonth({Месяц}[, {Число}])`	Устанавливает месяц и число, если заданы
`setSeconds({Секунды} [, {Миллисекунды}])`	Устанавливает секунды и миллисекунды, если заданы
`setTime({Время})`	Устанавливает время, принимая в качестве параметра количество миллисекунд, прошедших с полуночи 1 января 1970 года
`setUTCDate({Число})`	Устанавливает число по универсальному времени
`setUTCFullYear({Год}[, {Месяц}[, {Число}]])`	Устанавливает год, а также месяц и число, если они заданы, по универсальному времени
`setUTCHours({Часы}[, {Минуты}[, {Секунды}[, {Миллисекунды}]]])`	Устанавливает часы, а также минуты, секунды и миллисекунды, если они заданы, по универсальному времени
`setUTCMilliseconds({Миллисекунды})`	Устанавливает миллисекунды по универсальному времени
`setUTCMinutes({Минуты} [, {Секунды}[, {Миллисекунды}]])`	Устанавливает минуты, а также секунды и миллисекунды, если они заданы, по универсальному времени

Таблица 2.11 (окончание)

Метод	Описание
`setUTCMonth({Месяц}[, {Число}])`	Устанавливает месяц и число, если заданы, по универсальному времени
`setUTCSeconds({Секунды}[, {Миллисекунды}])`	Устанавливает секунды и миллисекунды, если заданы, по универсальному времени
`setYear({Год})`	Устанавливает год. Предусмотрен для совместимости; вместо него рекомендуется использовать метод `setFullYear`
`toGMTString()`	Преобразует дату в строку в формате гринвичского времени и возвращает ее. Предусмотрен для совместимости; вместо него рекомендуется использовать метод `toUTCString`
`toLocaleString()`	Преобразует дату в строку, используя интернациональные установки системы, и возвращает ее. Возвращаемое значение не рекомендуется использовать для вычислений
`toSource()`	Возвращает строку, представляющую исходный код даты
`toString()`	Преобразует дату в строку и возвращает ее
`toUTCString()`	Преобразует дату в строку в формате универсального времени и возвращает ее
`UTC({Год}, {Месяц}, {Число}[, {Часы}[, {Минуты}[, {Секунды}[,{Миллисекунды}]]]])`	Возвращает количество миллисекунд, прошедших между заданной датой и полуночью 1 января 1970 года по универсальному времени
`valueOf()`	Возвращает значение даты

Примечание

Методы `getFullYear`, `getMilliseconds`, `getUTCDate`, `getUTCDay`, `getUTCFullYear`, `getUTCHours`, `getUTCMilliseconds`, `getUTCMinutes`, `getUTCMonth`, `getUTCSeconds`, `setFullYear`, `setHours`, `setMilliseconds`, `setUTCDate`, `setUTCFullYear`, `setUTCHours`, `setUTCMilliseconds`, `setUTCMinutes`, `setUTCMonth`, `setUTCSeconds`, `toUTCString`, `valueOf` поддерживаются Internet Explorer начиная с версии 4.0 и Navigator — с версии 4.06. Метод `setDate` включен в Internet Explorer начиная с версии 4.0 и Navigator — с версии 2.0. Метод `getVarDate` имеется лишь в Internet Explorer начиная

с 4.0. Метод `toSource` разрешен только в Navigator начиная с 4.06. В старых версиях Navigator, до 4.06, методы `setHours`, `setMinutes`, `setMonth`, `setSeconds` тоже поддерживались, но имели лишь один — первый и обязательный — параметр. Также в старых версиях Navigator до 4.06 метод `UTC` не имел параметра миллисекунд.

Класс функции *Function*

Класс функции позволяет вам манипулировать функцией как объектом.

Конструктор объекта-функции имеет следующий формат:

```
{Имя функции} = new Function({Список аргументов, заключенных в кавычки,
разделенный запятыми} {Тело функции, заключенное в кавычки});
```

Это значит, что вы можете создавать функции как уже знакомым вам путем, с помощью ключевого слова `function`, так и тем же образом, каким создаются объекты. Заметьте, что во втором случае и аргументы, и тело функции заключаются в кавычки.

Давайте рассмотрим пример определения функции сначала традиционным путем, потом — новым, в виде объекта.

```
function samplefunc(a, b) {
  return (a + b) / 2;
  }
samplefunc = new Function("a", "b", "(a + b) / 2");
```

И при этом вызывать созданную функцию вы можете по-старому, независимо от способа создания.

```
f = samplefunc(s, 2) - 9;
```

Класс функции поддерживает много свойств и методов. Все свойства перечислены в табл. 2.12, а методы — в табл. 2.13.

Таблица 2.12. *Свойства класса функции* `Function`

Свойство	Описание
`arguments`	Возвращает массив аргументов, переданных в функцию. Доступно только внутри тела функции. Вы можете использовать свойство `length`, чтобы получить количество аргументов. Подробнее об объекте `Arguments` см. ниже
`arity`	Возвращает количество аргументов, требуемых функцией. Может быть использовано вне тела функции
`caller`	Возвращает ссылку на функцию, вызвавшую текущую функцию. Если функция вызвана из программы, возвращается `null`
`length`	То же, что и `arity`

Таблица 2.13. Методы класса функции `Function`

Метод	Описание
`toSource()`	Возвращает строку, представляющую исходный код функции
`toString()`	То же самое, что и `toSource`
`valueOf()`	Возвращает указатель на функцию

Примечание

Класс `Function` поддерживается Internet Explorer начиная с версии 4.0 и Navigator — с версии 3.0. Свойство `length` имеется только в Navigator начиная с 3.0. Свойство `arity` введено лишь в Navigator начиная с 4.0. Метод `toSource` разрешен только в Navigator начиная с 4.06. Свойство `arity` поддерживается лишь Navigator начиная с 4.0. Свойство `caller` имеется только в Internet Explorer начиная с 4.0.

Класс массива аргументов *Arguments*

Класс массива аргументов перечисляет все аргументы, переданные текущей функции, и еще кое-какие свойства. Массив аргументов доступен только внутри тела функции.

Можно получить доступ к любому аргументу, просто указав его индекс.

```
a = arguments[1];
```

Также массив аргументов поддерживает некоторые свойства, перечисленные в табл. 2.14.

Таблица 2.14. Свойства класса массива аргументов `Arguments`

Свойство	Описание
`callee`	Возвращает указатель на саму текущую функцию. Вызов `function.arguments.callee(n)` аналогичен вызову `function(n)`
`caller`	Возвращает ссылку на функцию, вызвавшую текущую функцию. Если функция вызвана из программы, возвращается `null`. Предусмотрено только для совместимости; вместо него рекомендуется использовать свойство `caller` класса `Function`
`length`	Возвращает количество аргументов, переданных в функцию

Примечание

Свойства `callee` и `length` поддерживаются Navigator только начиная с 4.0 и Internet Explorer — с 5.5. Свойство `caller` имеется только в Navigator начиная с 3.0.

Класс глобального объекта *Global*

Глобальный объект `Global`, существующий в единственном экземпляре и не требующий создания, представляет набор свойств и методов, доступных отовсюду в программе. Эти свойства и методы являются не чем иным, как встроенными функциями JavaScript, перечисленными в табл. 2.9. Те из них, что не требуют ни параметров, ни скобок, являются свойствами. (Вот почему они не требуют скобок — это ведь фактически не функции, а свойства.) При вызове их само имя глобального объекта `Global` не указывается. То есть, так делать нельзя:

```
bool = global.isFinite(x / y);
```

Правильный вариант:

```
bool = isFinite(x / y);
```

Иными словами, следует вызывать методы и свойства глобального объекта как обычные функции. Только запомните, что свойства не требуют скобок.

Странный объект Global

Глобальный объект `Global` — это что-то. Фактически он поддерживается только Internet Explorer (начиная с версии 5.0) и существует чисто номинально. Navigator совершенно ничего не знает о его существовании и чувствует себя превосходно. Зачем программистам Microsoft понадобилось оформлять встроенные функции JavaScript как свойства и методы некоего "невидимого" глобального объекта, неизвестно.

Математический класс *Math*

Математический класс предоставляет программистам-математикам набор тригонометрических и логарифмических функций. Он похож на глобальный объект: также существует в единственном экземпляре и также не требует создания, об этом заботится интерпретатор. Все его свойства перечислены в табл. 2.15, методы — в табл. 2.16.

Таблица 2.15. *Свойства математического класса* `Math`

Свойство	Описание
`E`	Возвращает константу Эйлера (e)
`LN10`	Возвращает значение ln 10
`LN2`	Возвращает значение ln 2
`LOG10E`	Возвращает значение lg e
`LOG2E`	Возвращает значение $\log_2 e$
`PI`	Возвращает значение π

Таблица 2.15 *(окончание)*

Свойство	Описание
SQRT1_2	Возвращает квадратный корень от 0,5
SQRT2	Возвращает квадратный корень от 2

Таблица 2.16. *Методы математического класса* `Math`

Метод	Описание
`abs({Число})`	Возвращает абсолютное значение аргумента
`acos({Число})`	Возвращает арккосинус аргумента в радианах
`asin({Число})`	Возвращает арксинус аргумента в радианах
`atan({Число})`	Возвращает арктангенс аргумента в радианах
`atan2({X}, {Y})`	Возвращает угол в радианах между горизонтальной осью и прямой, проведенной через начало координат и точку с координатами X,Y
`ceil({Число})`	Возвращает ближайшее целое число, большее или равное аргументу
`cos({Число})`	Возвращает косинус аргумента, заданного в радианах
`exp({Число})`	Возвращает значение $e^{Число}$
`floor({Число})`	Возвращает ближайшее целое число, меньше аргумента или равное ему
`log({Число})`	Возвращает натуральный логарифм аргумента
`max([{Список аргументов, разделенных запятыми}])`	Возвращает максимальный из аргументов. Если не задан ни один аргумент, возвращается значение `NEGATIVE_INFINITY` (см. описание класса `Number`). Если один из аргументов равен `NaN`, возвращается `NaN`
`min([{Список аргументов, разделенных запятыми}])`	Возвращает минимальный из аргументов. Если не задан ни один аргумент, возвращается значение `POSITIVE_INFINITY` (см. описание класса `Number`). Если один из аргументов равен `NaN`, возвращается `NaN`
`pow({Основание}, {Порядок})`	Возвращает значение $основание^{порядок}$
`random()`	Возвращает псевдослучайное число от 0 включительно до 1 исключительно

Таблица 2.16 (окончание)

Метод	Описание
round({Число})	Возвращает значение аргумента, округленное до ближайшего целого
sin({Число})	Возвращает синус аргумента, заданного в радианах
sqrt({Число})	Возвращает квадратный корень от аргумента
tan({Число})	Возвращает тангенс аргумента, заданного в радианах

Примечание

Метод max в Navigator имеет формат max({Аргумент 1}, {Аргумент2}), а метод min — min({Аргумент 1}, {Аргумент2}). Метод random в Navigator 2.0 работал только на платформе Unix.

Класс числовой величины *Number*

Класс Number отвечает за объектное представление числовых величин. Его конструктор требует одного аргумента, который станет начальным значением числа.

```
varnumeric = new Number(123);
```

Свойства класса числа перечислены в табл. 2.17, методы — в табл. 2.18.

Таблица 2.17. Свойства класса числа Number

Свойство	Описание
MAX_VALUE	Возвращает максимальное допустимое в JavaScript число. Оно примерно равно $1{,}79 \cdot 10^{308}$
MIN_VALUE	Возвращает минимальное допустимое в JavaScript число. Оно примерно равно $5 \cdot 10^{-324}$
NaN	То же самое, что свойство глобального объекта NaN
NEGATIVE_INFINITY	Возвращает значение "минус бесконечность"
POSITIVE_INFINITY	Возвращает значение "плюс бесконечность"

Таблица 2.18. Методы класса числа Number

Метод	Описание
toLocaleString()	Преобразует число в строку, используя интернациональные установки системы, и возвращает ее. Возвращаемое значение не рекомендуется использовать для вычислений

Таблица 2.18 (окончание)

Метод	Описание
`toSource()`	Возвращает строку, представляющую исходный код объекта
`toString()`	Возвращает строковое представление числа
`valueOf()`	Возвращает само число

Примечание

Класс `Number` поддерживается Internet Explorer только начиная с 4.0 и Navigator — с 3.0. Метод `toLocaleString` существует только в Internet Explorer. Метод `toSource` поддерживается только Navigator с 4.06.

Класс строки *String*

Класс `String` отвечает за объектное представление строк и манипуляцию с ними. Его конструктор может принимать один аргумент, который станет начальным значением строки.

```
varstring = new String("JavaScript");
```

Если параметр конструктора пропущен, создается пустая строки.

Класс строки поддерживает свойство `length`, возвращающее длину строки в символах. Заметьте, однако, что символы в строке нумеруются с нуля, так же, как и элементы в массиве.

Методы класса строки перечислены в табл. 2.19. В основном, они служат для работы с кодом HTML.

***Таблица 2.19.** Методы класса строки `String`*

Метод	Описание
`anchor({Имя "якоря"})`	Преобразует строку в "якорь" HTML с именем, переданным в качестве параметра
`big()`	Помещает текст строки внутрь парного тега `<BIG>`
`blink()`	Помещает текст строки внутрь парного тега `<BLINK>`
`bold()`	Помещает текст строки внутрь парного тега `<BOLD>`
`charAt({Номер символа})`	Возвращает символ, номер которого передан в качестве параметра

Таблица 2.19 *(продолжение)*

Метод	Описание
`charCodeAt({Номер символа})`	Возвращает код символа, номер которого передан в качестве параметра, в формате Unicode
`concat([{Список строковых значений, разделенных запятыми}])`	Объединяет текущую строку со всеми строками, переданными в качестве аргументов, и возвращает результат
`fixed()`	Помещает текст строки внутрь парного тега `<TT>`
`fontcolor({Цвет})`	Помещает текст строки внутрь парного тега `<FONT>` с установленным атрибутом `COLOR={Цвет}`
`fontsize({Размер})`	Помещает текст строки внутрь парного тега `<FONT>` с установленным атрибутом `SIZE={Размер}`
`fromCharCode([{Список кодов символов Unicode, разделенных запятыми}])`	Возвращает строку, созданную из символов, Unicode-коды которых переданы в качестве параметров. Текущая строка не изменяется
`indexOf({Подстрока}, [{Начало поиска}])`	Возвращает номер позиции подстроки в текущей строке. Второй параметр задает номер символа, с которого начинается поиск; если пропущен, поиск начинается с начала строки
`italics()`	Помещает текст строки внутрь парного тега `<I>`
`lastIndexOf({Подстрока}, [{Начало поиска}])`	То же самое, что и `indexOf`, но поиск ведется до конца строки. Таким образом, фактически возвращается последняя позиция подстроки в текущей строке
`link({Интернет-адрес})`	Преобразует строку в гиперссылку, указывающую на адрес, переданный в качестве параметра
`match({Регулярное выражение})`	Выполняет поиск в строке, используя регулярное выражение, переданное в качестве параметра, и возвращает массив с результатами поиска. Если ничего не найдено, возвращает `null`. О регулярных выражениях см. в конце этой главы
`replace({Регулярное выражение}, {Текст для замены})`	Выполняет поиск и замену в строке, используя регулярное выражение, переданное в качестве параметра, и возвращает строку, полученную в результате этих замен. О регулярных выражениях см. в конце этой главы

Таблица 2.19 *(окончание)*

Метод	Описание
`search({Регулярное выражение})`	Выполняет поиск в строке, используя регулярное выражение, переданное в качестве параметра, и возвращает позицию первой подстроки, совпадающей с регулярным выражением. О регулярных выражениях см. в конце этой главы
`slice({Начало фрагмента}, [{Конец фрагмента}])`	Возвращает фрагмент строки в виде объекта. Если второй параметр пропущен, выбираются все символы до конца строки. Последний символ во фрагмент не включается
`small()`	Помещает текст строки внутрь парного тега `<SMALL>`
`split([{Разделитель}[, {Лимит}]])`	Возвращает массив, заполненный строками, полученными в результате разделения текущей строки. Символ, по которому текущая строка будет делиться на подстроки, передается первым параметром; если он опущен, возвращается массив из одного элемента, содержащего целую строку. Второй параметр, если он присутствует, задает лимит количества элементов в результирующем массиве
`strike()`	Помещает текст строки внутрь парного тега `<STRIKE>`
`sub()`	Помещает текст строки внутрь парного тега `<SUB>`
`substr({Начало фрагмента}, [{Длина фрагмента}])`	Возвращает фрагмент строки заданной длины. Если второй параметр пропущен, выбираются все символы до конца строки
`substring({Начало фрагмента}, {Конец фрагмента})`	Возвращает фрагмент строки. Последний символ во фрагмент не включается
`sup()`	Помещает текст строки внутрь парного тега `<SUP>`
`toLowerCase()`	Конвертирует все символы строки в нижний регистр
`toSource()`	Возвращает строку, представляющую исходный код строкового объекта
`toString()`	Возвращает значение строки
`toUpperCase()`	Конвертирует все символы строки в верхний регистр
`valueOf()`	То же самое, что и `toString`

Примечание

Метод `charCodeAt` поддерживается Internet Explorer только начиная с 5.5 и Navigator — с 4.06. Методы `concat`, `fromCharCode`, `match`, `replace`, `search` введены в Internet Explorer и Navigator только начиная с 4.0. Методы `slice`, `substr` имеются в Internet Explorer только начиная с 4.0. Метод `split` поддерживается Internet Explorer только начиная с 4.0 и Navigator — с 3.0. Метод `toSource` существует только в Navigator начиная с 4.06. В Navigator 4.0—4.06 метод `charCodeAt` возвращал код символа ASCII, а метод `fromCharCode` принимал коды символов ASCII.

Пользовательские классы

Создание

Всем хороши встроенные классы JavaScript. Но иногда все же их возможностей не хватает. И тогда пользователь создает свои классы.

Что же нужно сделать, чтобы создать свой класс? Не так уж и много. Если класс не имеет конструктора, необходимо просто создать его оператором `new` с использованием инициализатора:

```
var somePoint;
somePoint = (x: 100, y: 50, color: "black");
```

Вот и все. Таким образом мы создали объект, представляющий некую графическую точку, имеющую свойства декартовых координат и цвета. Теперь мы можем его использовать.

```
somePoint.y = somePoint.x / 2 + 20;
```

Возможно даже добавить в этот объект новые свойства.

```
somePoint.radius = 0.5;
```

Однако мы не создали свой класс объекта. Чтобы действительно создать свой класс, нужно определить конструктор.

Как мы узнали раньше, конструктор определяет свойства класса и заполняет их значениями по умолчанию. (Значения по умолчанию могут быть жестко определены в коде программы или переданы в качестве параметров в конструктор.) Также конструктор определяет методы класса. И еще: фактически имя функции-конструктора становится именем класса.

Давайте определим класс графической точки. Для этого напишем соответствующий конструктор.

```
function Point(x, y, color) {
  this.x = x;
  this.y = y;
  this.color = color;
  }
```

Мы использовали новый оператор this, чтобы сослаться на текущий объект. Запомните этот оператор — он нам впоследствии очень пригодится.

Теперь мы может создать объект класса Point.

```
var somePoint;
somePoint = new Point(100, 50, "black");
```

Давайте добавим метод в наш новоиспеченный класс. Вспомним, что переменные могут содержать указатели на функцию. А раз на это способны переменные, то смогут и свойства класса.

Сначала определим функцию метода. Она будет проверять, попадают ли координаты точки в некоторые пределы, и возвращать соответственно true или false.

```
function isPointInBounds() {
  if (this.x>0 && this.x<400) {
    return (this.y>0 && this.y<200);
    }
  else return false;
  }
```

Переопределяем наш новый класс.

```
function Point(x, y, color) {
  this.x = x;
  this.y = y;
  this.color = color;
  this.isPointInBounds = isPointInBounds;
  }
```

В предпоследней строке мы присвоили свойству isPointInBounds указатель на функцию isPointInBounds. Теперь мы можем использовать это свойство для вызова функции; фактически, эта функция стала частью объекта, т. е. его методом.

```
if (somePoint.isPointInBounds()) then . . .
```

Но что делать, если пользователь не задал все параметры, требуемые конструктором? В этом случае можно предусмотреть значения по умолчанию для свойств, используя специальный вид оператора || (логическое ИЛИ).

```
function Point(x, y, color) {
  this.x = x || 100;
  this.y = y || 50;
  this.color = color || "black";
  this.isPointInBounds = isPointInBounds;
  }
```

Если значение аргумента, указанного слева от оператора ||, определено, возвращается оно. В противном случае возвращается значение аргумента, указанного справа от оператора ||, в данном случае, наше значение по умолчанию. Так что мы может создать наш новый объект таким образом:

```
somePoint = new Point();
```

Теперь вернемся немного назад. Помните, мы добавили в объект, созданный с использованием инициализатора, новое свойство. Мы можем сделать то же самое и с новым объектом somePoint, созданным уже конструктором.

```
somePoint.radius = 0.5;
```

Но вот беда: свойство, добавленное нами, будет доступно только в объекте somePoint, но никак не в других объектах класса Point. Что же делать?

Использование прототипов

Выход есть — воспользоваться прототипом объекта и добавить свойство в него.

Прототип — это своего рода ссылка на класс объекта. У всех классов JavaScript есть свойство prototype, указывающее на прототип, даже встроенных классов. Им-то мы и воспользуемся, чтобы расширить наш класс.

Вместо того, чтобы добавлять свойство объекту:

```
somePoint.radius = 0.5;
```

мы добавим его прототипу, используя новое свойство любого класса — prototype.

```
Point.prototype.radius = 0.5;
```

Теперь все объекты класса Point будут иметь свойство radius. Но только свойство radius объекта somePoint будет установлено в 0.5, а у остальных объектов оно не будет определено.

Мы также можем изменить встроенные классы, добавив им новые свойства и методы. Скажем, добавить в класс Math нужные математические функции и константы.

Но прототипы разрешается использовать и для других целей. Например, мы можем *наследовать* классы. Поясню это на примере.

Предположим, мы хотим добавить в класс Point сразу несколько новых свойств. Каким образом это выполнить? Можно добавить их все по очереди в прототип какого-нибудь объекта этого класса, но это не очень красиво. Мы сделаем по-другому. Напишем новый класс точки SuperPoint и унаследуем старый класс Point со всеми его свойствами и методами. Класс SuperPoint станет *потомком* класса Point и сможет использовать все свойства и методы *класса-родителя* (или *предка*). Таким образом, мы будем иметь

два класса точки — простой и "продвинутый" — и сможем использовать каждый из них там, где он больше всего подойдет.

Сначала определим класс SuperPoint и все новые (подчеркиваю — новые) свойства. И рассмотрим наш код по строчкам.

```
function SuperPoint(x, y, color, radius, edgeColor) {
```

Здесь все понятно. Мы определяем новую функцию конструктора для нашего нового класса. Заметьте, что в списке параметров конструктора мы определяем и новые, и унаследованные свойства.

```
  this.base = Point;
  this.base(x, y, color);
```

Мы устанавливаем новое свойство base класса SuperPoint и присваиваем ему указатель на конструктор класса-родителя Point. После этого вызываем конструктор родителя, передав ему требуемые им параметры. В результате этого вызова все унаследованные свойства получат значения. Имя свойства base роли не играет и выбрано только ради удобства.

```
  this.radius = radius || 0.5;
  this.edgeColor = edgeColor || "black";
```

Теперь инициализируем новые свойства класса SuperPoint. Обратите внимание на то, что мы предусмотрели для них значения по умолчанию, присваиваемые, если программист, который будет использовать наш класс, не передаст в конструктор одно из значений (или вообще ничего не передаст). Использование значений по умолчанию — хороший стиль программирования.

```
  }
SuperPoint.prototype = new Point;
```

Данное выражение задает предка для нашего нового класса. Заметьте, что оно помещается вне тела конструктора.

Теперь мы можем создать объект нашего нового класса и использовать как новые свойства, так и унаследованные.

```
var obj;
obj = new SuperPoint(100, 100, "green", 0.6);
```

Мы передали в конструктор не все параметры, но это не страшно, ведь мы предусмотрели для свойств значения по умолчанию.

```
obj.radius = Math.SQRT(5);
```

Сейчас мы обратились к новому свойству класса SuperPoint и присвоили ему значение квадратного корня из 5.

```
a = obj.x + 20;
```

А теперь мы обратились к унаследованному от Point свойству.

Вы можете переопределить любое свойство и любой метод, унаследованный от родителя. Рассмотрим построчно еще один пример.

```
function SuperPuperPoint(x, y, color, radius, edgeColor) {
  this.base = SuperPoint;
  this.base(x, y, color, radius, edgeColor);
  this.isPointInBounds = isPointInBounds2;
```

Здесь мы переопределили метод `isPointInBounds` потомка, присвоив свойству `isPointInBounds` указатель на другую функцию `isPointInBounds2`.

```
  }
SuperPuperPoint.prototype = new SuperPoint;
```

И не забываем указать родителя нашего нового класса.

Класс *Object* — общий родитель

Все, абсолютно все классы JavaScript — и встроенные, и пользовательские — происходят от класса `Object`. То есть, класс `Object` — родитель всех классов. А это значит, что все классы JavaScript наследуют его свойства и методы и, как правило, переопределяют их.

Конструктор класса `Object` может принимать один числовой, строковый или логический параметр. Если он пропущен, объект не инициализируется никаким значением.

Класс `Object` поддерживает два свойства. Свойство `prototype` уже знакомо вам и определяет прототип класса. Свойство `constructor` возвращает ссылку на конструктор класса и может использоваться, скажем, для определения класса объекта.

```
if (obj.constructor == Point) . . .
```

Все методы класса `Object` перечислены в табл. 2.20.

Таблица 2.20. *Методы класса* `Object`

Метод	Описание
`eval({Строка})`	Аналогичен встроенной функции `eval`. Предусмотрен только для совместимости
`toLocaleString()`	Возвращает строковое представление значения объекта с учетом интернациональных установок. Этот метод был описан ранее для всех классов
`toSource()`	Возвращает строку, представляющую исходный код объекта. Этот метод был описан ранее для всех классов
`toString()`	Возвращает строковое значение объекта. Этот метод был описан ранее для всех классов

Таблица 2.20 *(окончание)*

Метод	Описание
`valueOf()`	Возвращает значение объекта. Этот метод был описан ранее для всех классов

Примечание

Класс `Object` существует в Internet Explorer начиная с версии 4.0. Свойства `constructor` и `prototype` в Navigator введены только начиная с версии 3.0. Методы `eval` и `valueOf` поддерживаются лишь Navigator начиная с 3.0. Метод `toSource` имеется только в Navigator начиная с 4.06. Метод `toLocaleString` существует лишь в Internet Explorer начиная с 3.0.

Регулярные выражения

Понятие регулярного выражения

Регулярные выражения — это своего рода шаблоны для поиска определенных комбинаций символов в строках. Как правило, регулярные выражения отличаются от обычных функций поиска подстроки тем, что позволяют производить очень сложный поиск. Регулярные выражения пришли в JavaScript из языка Perl, очень мощного языка программирования, созданного специально для обработки строк. А поскольку в JavaScript очень часто приходится выполнять обработку пользовательского ввода, то эта новая возможность пришлась как нельзя более кстати.

Примечание

Регулярные выражения поддерживаются Navigator только начиная с 3.0 и Internet Explorer — с 4.0.

Примером регулярного выражения может служить шаблон, который вы используете для поиска файлов на диске. Скажем, "File*.txt" позволит найти все файлы с именами "File.txt", "File1.txt", "Filenn.txt", "Filesome.txt" и т. п., а "File?.txt" отыщет для вас файлы "File1.txt", "Files.txt", "Filer.txt" и т. п. Можете рассматривать эти два шаблона как простейшее регулярное выражение.

Регулярные выражения в JavaScript (как и в Perl) пишутся с использованием специального синтаксиса, использующего зарезервированные литералы. Например, перепишем наши шаблоны поиска файлов в регулярные выражения.

```
File*.txt → /[fF]ile.*\.txt/
```

Это означает следующее: ищи последовательность из одного из символов "f" или "F" (`[fF]`), строки "ile", любого символа (.), встретившегося несколько

раз (*) или не встретившегося вообще, точки (\.) и строки "txt". Заметьте, что мы заключили регулярное выражение не в кавычки, а в слэши "/".

```
File?.txt → /[fF]ile.\.txt/
```

Здесь, в общем, то же самое. Единственное: пропал литерал множественного совпадения *, стоявший после литерала любого символа . (точка). Это значит, что между "file" и ".txt" должен теперь стоять один символ.

Напишем регулярное выражение, совпадающее с любым парным тегом HTML.

```
/<(.*)>.*<\/\1>/
```

Это значит следующее: ищи последовательность из символа "<" и идущей за ним цепочки любых символов (.*), запомни ее (скобки, в которые заключены литералы .*), далее должен быть знак ">". Это у нас будет открывающий тег. После этого — опять последовательность любых символов (.*), содержание тега. Далее: символы "<", "/" (\/), запомненная ранее группа символов (\1) и знак ">" — закрывающий тег.

Литералы регулярных выражений

Все литералы регулярных выражений перечислены в табл. 2.21.

Таблица 2.21. *Литералы регулярных выражений*

Литерал	Описание
\	Входит в состав специальных символов, а также дает понять интерпретатору, что следующий символ — не литерал. Например, \n — символ конца строки (специальный символ). Используется перед восьмеричными кодами символов и для извлечения из памяти сохраненных подвыражений. Если вы хотите использовать в регулярном выражении символ, являющийся литералом, поместите перед ним этот знак, например, \(
^	Обозначает начало строки
$	Обозначает конец строки
*	Предшествующий символ должен встретиться в строке сколько угодно раз или не встретиться вообще
+	Предшествующий символ должен встретиться в строке один или более раз
?	Предшествующий символ должен встретиться в строке один раз или не встретиться вообще

Таблица 2.21 *(продолжение)*

Литерал	Описание
{{Число}}	Предшествующий символ должен встретиться в строке указанное количество раз
{{Число},}	Предшествующий символ должен встретиться в строке указанное количество или больше раз
{{Число 1}, {Число 2}}	Предшествующий символ должен встретиться в строке от {Число 1} до {Число 2} раз
.	Любой символ, кроме \n (новая строка)
({Подвыражение})	Ищет {Подвыражение} и сохраняет в памяти найденную группу символов. Впоследствии ее можно извлечь, использовав синтаксис \{Номер группы}. Группы символов нумеруются согласно порядку их появления в регулярном выражении
{Символ 1}|{Символ 2}	Ищет один из двух символов
[{Набор символов}]	Ищет символ из заданного набора. Набор может иметь вид abcd или a-d, означающий, что необходимо искать символы от "a" до "d"
[^{Набор символов}]	Ищет любой символ, не вошедший в набор
\b	Граница слова, т. е. позиция между словом и пробелом. Например, ke\b совпадает с буквами "ke" в слове "make", но не в слове "kept"
\B	Противоположно границе слова. Например, ke\B совпадет с "ke" в слове "kept", но не в слове "make"
\c{Символ}	Совпадает с управляющим символом вида "Ctrl"+ +"{Символ}"
\d	Любая цифра от 0 до 9
\D	Любой нецифровой символ
\f	Символ перевода страницы
\n	Символ новой строки
\r	Символ возврата каретки
\s	Пробел, табуляция, перевод страницы, новая строка или перевод строки
\S	Все, за исключением вышеперечисленного
\t	Табуляция

Таблица 2.21 (окончание)

Литерал	Описание
\v	Вертикальная табуляция
\w	Буква, цифра или подчеркивание
\W	Все, за исключением вышеперечисленного
\x{Код}	Символ с указанным шестнадцатеричным кодом
\o{Код}	Символ с указанным восьмеричным кодом
\{Номер группы}	Извлекает из памяти сохраненную ранее группу символов с заданным номером

Еще два примера регулярных выражений.

```
/(\w+)@([\w\._]+)/
```

Это выражение совпадает с любым адресом электронной почты. Оно разбивает его на две части — имя почтового ящика и имя сервера — и сохраняет их в памяти в виде групп символов под номерами 1 и 2.

```
/(\w+:\/\/)?([^/]+)(.*)?/
```

Это выражение совпадает с любым Web-адресом. Оно разбивает его на три части — протокол, собственно адрес и имя файла — и сохраняет их в памяти в виде групп символов. При этом оно учитывает, что протокол и имя файла могут отсутствовать.

Класс *RegExp*

Класс `RegExp` отвечает за обработку строк с помощью регулярных выражений. Его конструктор имеет следующий формат:

```
RegExp({Регулярное выражение}[, {Флаги}])
```

Как видите, он принимает один обязательный параметр — регулярное выражение. Учтите, что в этом случае регулярное выражение заключается в кавычки, а не слэши. Второй — необязательный — параметр задает дополнительные параметры поиска и может включать в разных сочетаниях три символа. Символ "g" задает глобальный поиск (т. е. поиск всех вхождений регулярного выражения в строке), символ "i" — игнорирование регистра символов, а символ "m" — многострочный поиск.

Доступен также такой вариант конструктора (близкий к Perl):

```
/{Регулярное выражение}/[{Флаги}]
```

Как же используется этот объект?

При рассмотрении класса строки String мы упомянули о трех методах: match, replace и search. Данные методы описаны в табл. 2.19. Именно они и используются для работы с регулярными выражениями.

```
var result, re, str;
str = "http://www.netscape.com";
re = new RegExp("w{3}", "i");
result = str.match(re);
```

В этом примере мы произвели поиск в строке Web-адреса компании Netscape трех букв "w" без учета регистра. Метод match вернет нам массив result, содержащий результаты поиска. Если не установлен флаг глобального поиска "g", первый элемент массива будет содержать найденную подстроку, а все последующие — сохраненные группы символов, если таковые есть. Если флаг глобального поиска установлен, массив содержит все найденные вхождения. В нашем случае этот массив будет иметь один элемент "www".

Класс RegExp поддерживает несколько свойств. Свойство lastIndex задает позицию в строке, откуда начнется поиск. Свойство source возвращает строку регулярного выражения и доступно только для чтения. Свойство global возвращает true, если установлен флаг "g", и false, если этот флаг не установлен. Свойство ignoreCase возвращает true, если установлен флаг "i", и false, если этот флаг не установлен. Свойство multiline возвращает true, если установлен флаг "m", и false, если этот флаг не установлен.

Также класс RegExp поддерживает три метода. Все они перечислены в табл. 2.22.

Таблица 2.22. *Методы класса* RegExp

Метод	Описание
compile({Регулярное выражение}[, {Флаги}])	Компилирует регулярное выражение во внутренний формат для ускорения работы. Также может использоваться для изменения регулярного выражения
exec({Строка})	Аналогичен методу match класса String, за тем исключением, что принадлежит классу RegExp, а строка, где нужно произвести поиск, передается как параметр
test({Строка})	Аналогичен методу search класса String, за тем исключением, что принадлежит классу RegExp, а строка, где нужно произвести поиск, передается как параметр. Возвращает true или false в зависимости от того, успешным ли был поиск или нет

Примечание

Класс RegExp поддерживается Navigator и Internet Explorer начиная с версий 4.0. Флаг "m" и свойство multiline в классе regExp поддерживается только Internet Explorer. В Navigator в методах exec и test можно опускать параметр. В этом случае строка для поиска берется из свойства input глобального объекта RegExp. О глобальном объекте RegExp см. ниже.

Глобальный объект *RegExp*

Глобальный объект RegExp (не путать с одноименным классом) служит для доступа к результатам поиска с использованием регулярного выражения. Этот объект создается самим интерпретатором (как и объекты Global и Math) и доступен всегда. Формат доступа к его свойствам таков.

```
RegExp.{Свойство}
```

Все свойства, поддерживаемые этим объектом, перечислены в табл. 2.23. Все они доступны только для чтения.

Таблица 2.23. *Свойства глобального объекта* RegExp

Свойство	Описание
${Номер подвыражения}	Возвращает одно их последних найденных подвыражений в зависимости от заданного номера. Номер может быть от 1 до 9. Интерпретатор JavaScript хранит в этих свойствах только девять последних найденных подвыражений; для доступа к остальным используйте массив, возвращаемых методами match и exec. Пример использования: $2
index	Возвращает позицию в строке найденной подстроки
input|$_	Возвращает строку, где производится поиск
lastIndex	Аналогично свойству lastIndex класса RegExp
lastMatch|$&	Возвращает последнюю найденную подстроку
lastParen|$+	Возвращает последнюю найденную группу символов, если в регулярном выражении использовались подвыражения
leftContext|$`	Возвращает строку, составленную из всех символов от начала исходной строки до последней найденной подстроки, не включая ее
rightContext|$'	Возвращает строку, составленную из всех символов от последней найденной подстроки, не включая ее, до конца исходной строки

Кроме того, глобальный объект RegExp поддерживает все свойства и методы, унаследованные от класса Object.

В табл. 2.19 был описан метод replace, служащий для замены найденной подстроки другим текстом, который передавался как второй параметр. В этом втором параметре вы можете использовать свойства глобального объекта RegExp, поместив их прямо в строковый литерал. Доступны свойства $&, $`, $' и ${Номер группы символов}. Для обозначения знака доллара используйте специальный символ $$.

```
result = str.replace(re, "$1 — $2");
```

Примечание

Свойства lastMatch, lastParen, leftContext, rightContext поддерживаются Internet Explorer только начиная с 5.5. В Navigator свойство input доступно для чтения и записи, свойство lastIndex для глобального объекта RegExp не поддерживается.

Пример использования регулярных выражений

В заключение рассмотрим пример использования регулярных выражений для обработки строк.

```
var re, str, protocol, address, filename, result;
str = "http://www.somedomain.ru/index2.html";
re = new RegExp("((\w+):\/\/)?([^/]+)(.*)?","i");
```

Подготавливаем исходные данные, в том числе регулярное выражение. В качестве исходной строки возьмем Web-адрес и "разберем" его на составные части.

Заметьте, что в регулярном выражении мы использовали вложенные друг в друга подвыражения. В этом случае сохраняется сначала внешнее, а потом внутреннее. Мы сделали это, чтобы выделить из группы символов "http://" обозначение протокола "http". А группу "http://", сохраненную под номером 1, можно потом просто не использовать.

```
result = re.exec(str);
```

Эта строка запускает "разборку".

```
if (result != null) {
```

Проверяем, совпало ли наше регулярное выражение с исходной строкой, т. е. правилен ли наш Web-адрес.

```
protocol = RegExp.$2;
address = RegExp.$3;
filename = RegExp.$4;
```

Здесь все должно быть понятно. Под номером 1 была сохранена группа символов "http://", которую мы не используем.

```
}
```

Регулярные выражения, рассмотренные нами здесь, были довольно просты. Впоследствии мы будем использовать более сложные выражения для проверки ввода пользователя.

Ошибки в скриптах JavaScript

Как правило, сам Web-обозреватель предупредит вас, если в скрипте, встроенном в вашу Web-страницу, встретилась ошибка. Формы этого предупреждения различны для разных программ Web-обозревателей. Иногда просто в строке состояния программы выдается сообщение о том, что при выполнении скрипта возникла ошибка. Самое полное, на мой взгляд, сообщение выдает Internet Explorer 5.5; оно включает описание ошибки и место в коде страницы, где она встретилась.

Все вышесказанное касается *синтаксических ошибок*, т. е. ошибок в синтаксисе языка. Ошибки в логике работы скрипта *(логические ошибки)* можно выявить только по результатам его работы. Если скрипт не содержит синтаксических ошибок, то формально он правилен, даже если и выдает неправильные результаты. Поэтому такие ошибки очень трудно найти и исправить. Единственный способ: тщательно проанализировать код скрипта, сопоставить реально полученный результат с ожидаемым и еще раз прочитать справочник по языку.

Есть еще одна разновидность ошибок: *ошибки времени выполнения*. Эти ошибки возникают во время выполнения скрипта в результате наступления каких-либо условий, не предусмотренных программистом. Они могут порождаться кодом, абсолютно правильным как с точки зрения синтаксиса, так и логики. Классический пример ошибки времени выполнения — деление на ноль. От ошибок времени выполнения спасает только тщательное кодирование и многократная проверка правильности всех условий; в нашем случае нужно проверить делитель на равенство нулю перед собственно делением.

Конечно, если вы используете специальные средства программирования, позволяющие писать и отлаживать скрипты на JavaScript (в частности, это позволяет делать Macromedia Dreamweaver 4.0), ваша задача значительно облегчается. Тем же, кто не является их счастливым обладателем, можно посоветовать только внимательнее писать скрипты и избегать совсем уж сложных конструкций — они, как показывает практика, почти всегда содержат ошибки.

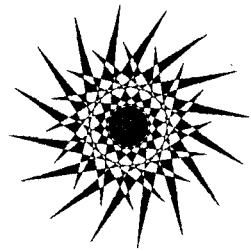
Хотите знать больше?

Мы рассмотрели язык JavaScript и все предлагаемые им для создания Web-скриптов возможности. Но одного языка JavaScript мало. Нужно знать еще и DOM. А вот об этом речь пойдет в следующих главах.

Классы и объекты DOM относятся к внешним, предоставляемым другими программами — не интерпретатором (виртуальной машиной) JavaScript, а Web-обозревателем. Поэтому они и рассматриваются отдельно от собственно языка.

И еще. В этой главе рассмотрены не все возможности JavaScript, а только те, что имеют отношение к Web-программированию. JavaScript может быть использован во многих приложениях: Microsoft призывает писать на своем JScript командные скрипты для замены устаревших bat-файлов командного процессора MS-DOS, а Netscape приспособила JavaScript для написания серверных скриптов. Но в главу вошли только те возможности, которые пригодятся вам для написания Web-страниц.

ГЛАВА 3



HTML и JavaScript: работаем вместе

Итак, мы изучили HTML и JavaScript (те, кто знал их ранее, освежил в памяти свои знания). Что теперь?

Пора свести все наши знания воедино, поговорить о том, ради чего и была написана эта книга: о Web-дизайне с использованием скриптов (сценариев). Настало время браться за главное: за Web-программирование. Пора заставить HTML и JavaScript, этих антиподов мира программирования, работать вместе. Пусть HTML отвечает за представление данных, а JavaScript описывает их поведение.

Я уже много раз говорил, что Internet Explorer и Navigator в этом плане сильно различаются. В дальнейшем, если какая-то возможность будет поддерживаться только одной из этих программ, я специально укажу на это.

HTML+JavaScript

Простейшая динамическая Web-страница

В главе 1, посвященной HTML, мы привели пример динамической Web-страницы, показывающей текущую дату. Эта страница содержала скрипт. Давайте еще раз посмотрим на ее код.

```
<HTML>
<HEAD>
<TITLE>Сегодня</TITLE>
</HEAD>
<BODY>
<P>
<SCRIPT LANGUAGE="JavaScript">
var d;
```

```
d=new Date();
document.write(d.toString());
</SCRIPT>
</P>
</BODY>
</HTML>
```

Когда Web-обозреватель загрузит эту страницу, он сразу же выполнит скрипт. В результате внутрь тега <P> будет вписано текстовое значение сегодняшней даты, созданное с учетом региональных настроек. Это достигается следующим кодом:

```
var d;
d=new Date();
document.write(d.toString());
```

Здесь мы создаем объект класса Date, инициализированный значением текущей даты, вызываем метод toString и передаем возвращенный им результат методу write объекта document. Последний метод как раз и вписывает текст, переданный в качестве аргумента, в данное место Web-страницы.

Скрипты такого рода — самые простые. Фактически, они создают нединамические страницы, ведь после однократного их выполнения страница больше никак не реагирует на действия пользователя и не изменяет своего содержимого. Чтобы научиться писать более "развитые" страницы, мы должны немного узнать о теге <SCRIPT> и способах встраивания скриптов в Web-страницы.

Как писать скрипты

Здесь мы узнаем несколько общих принципов написания Web-скриптов.

Прежде всего, поговорим о теге <SCRIPT>. Этот тег, как вы уже поняли, служит для вставки скриптов в HTML-код страницы. Его формат таков:

```
<SCRIPT [LANGUAGE="{Язык программирования, на котором написан скрипт}"]
[SRC="{Адрес файла со скриптом}"]>
. . . Текст скрипта
</SCRIPT>
```

Текст скрипта помещается внутри тега <SCRIPT>.

Атрибут LANGUAGE позволяет указать, на каком языке программирования написан скрипт. Значение по умолчанию — "JavaScript". Internet Explorer, кроме того, поддерживает скрипты, написанные на языке VBScript, которому соответствует значение атрибута "VBScript". Navigator тоже не остается в долгу и позволяет задать версию интерпретатора JavaScript.

```
<SCRIPT LANGUAGE="JavaScript 1.2">
```

Это может пригодиться в случаях, когда вы используете возможность языка JavaScript, специфичную для какой-либо версии. (В Navigator такое встречается сплошь и рядом.)

В табл. 3.1 приведены версии интерпретатора JavaScript и соответствие их разным версиям Navigator.

Таблица 3.1. *Версии интерпретатора JavaScript*

JavaScript	Navigator	JavaScript	Navigator
1.0	2.0	1.2	4.0
1.1	3.0	1.3	4.06

Второй атрибут `SRC` служит для задания адреса файла, содержащего скрипт. В таком случае обычно парный тег `<SCRIPT>` превращается в одинарный.

```
<SCRIPT SRC="printofdate.js">
```

`js` — стандартное расширение для JavaScript-файлов.

Но что случится, если Web-обозреватель не поддерживает скрипты? (Например, это слишком старая версия программы, или пользователь отключил поддержку скриптов в настройках безопасности.) Web-обозреватель проигнорирует тег `<SCRIPT>` и выведет на экран текст скрипта. Чтобы избежать этого, скрипт внутри тега `<SCRIPT>` рекомендуют заключать в комментарий.

```
<SCRIPT>
<!--
. . . Текст скрипта
-->
</SCRIPT>
```

Однако некоторые версии Navigator в этом случае "не видят" скрипта, хотя, согласно техническим руководствам самой Netscape, должны. Так что это совет применим только для Internet Explorer.

В качестве альтернативы вы можете использовать тег `<NOSCRIPT>`. Этот тег поддерживается и Internet Explorer, и Navigator.

```
<NOSCRIPT>
. . . Текст, отображаемый, если Web-обозреватель не поддерживает скрипты
</NOSCRIPT>
```

Тег `<NOSCRIPT>` может помещаться где угодно в тексте страницы, даже вне тега `<BODY>`. Текст, помещенный внутрь этого тега, будет отображаться вместо всей страницы. То есть, пользователь Web-обозревателя с отключенной поддержкой скриптов увидит в окне только этот текст и больше ничего.

Что же может содержать Web-скрипт? Любой JavaScript-код. Например, функции.

Рассмотрим слегка модифицированный пример нашей простейшей Web-страницы. Сохраним его под именем 3.1.

```
<HTML>
<HEAD>
<TITLE>Сегодня</TITLE>
<SCRIPT>
function writeDate() {
var d;
d=new Date();
document.write(d.toString()); }
</SCRIPT>
</HEAD>
<BODY>
<P>
<SCRIPT LANGUAGE="JavaScript">
writeDate();
</SCRIPT>
</P>
</BODY>
</HTML>
```

Если вы загрузите ее в Web-обозреватель, то увидите показанное на рис. 1.17.

Заметьте, что эта страница содержит два скрипта: один — с кодом функции, другой — фактически вызывающий эту функцию для того, чтобы вставить в документ дату. Мы поместили скрипт с кодом функции в заголовок HTML-документа с тем, чтобы она была определена заведомо раньше всех вызовов. Это обычная практика при написании динамических Web-страниц: определения всех функций выносятся в заголовок страницы.

Вот и все основы. Теперь самое время понять, как HTML и JavaScript взаимодействуют друг с другом.

Объектная модель документа

При рассмотрении JavaScript мы заметили, что он может поддерживать так называемые внешние классы и объекты, определенные другими программами. Тогда вы могли бы удивиться, зачем я вообще о них упоминал. А вот зачем...

Знаете ли вы, что Web-страница, которую вы просматриваете в окне Web-обозревателя, может быть описана как набор объектов? Скажем, она включает большой объект "документ-в-целом" и более мелкие объекты: "абзац-1",

"абзац-2", "абзац-3" и "рисунок". И вы можете обращаться к этим объектам из того же JavaScript, пользуясь прекрасно знакомым вам синтаксисом.

```
Рисунок.ДвигайВперед;
Абзац-1.Ширина = 80%;
Абзац-3.Покажи;
```

Конечно, я сильно утрирую, но принцип таков. И все это работает!

Например, рассмотрим снова наш многострадальный скрипт.

```
var d;
d=new Date();
document.write(d.toString()); }
```

Документ `document` — это наш "документ-в-целом". А `write` — его метод, вставляющий текст, переданный в качестве параметра, в текущее место HTML-документа. Мы все это время работали с Web-страничкой нашего примера как с набором объектов, являющихся внешними для языка JavaScript.

Совокупность объектов, описывающая Web-страницу, со всеми их методами и свойствами называется *объектной моделью документа* (Document Object Model, DOM). А технология создания Web-страницы, при которой обычный HTML-код объединяется с JavaScript-кодом, причем последний управляет страницей с помощью объектной модели, называется *динамическим HTML* (Dynamic HTML). Ранее динамическим HTML называли и совокупность расширений обычного HTML, предложенную Microsoft и Netscape для реализации объектной модели. После принятия W3C стандартной спецификации DOM уровня 1 (DOM level 1) все эти дополнения вошли в окончательную редакцию HTML 4.0.

Полное описание объектной модели документа см. в приложении 3.

Объект *document*

Здесь мы познакомимся с объектом `document` и выясним, что еще с ним можно сделать.

Прежде всего, запомните, что объект `document` существует в единственном экземпляре для всего HTML-документа. Он присутствует всегда, если существует HTML-документ, поэтому специально создавать его не требуется.

Свойства и методы

Объект `document` поддерживает множество свойств. Многие из них перечислены в табл. 3.2.

Таблица 3.2. *Свойства объекта* `document`

Свойство	Описание
`activeElement`	Возвращает ссылку на элемент страницы, находящийся в фокусе. Например, на активную гиперссылку. Поддерживается только Internet Explorer начиная с 4.0. (Об объектах элементов страницы см. ниже.)
`alinkColor`	Цвет активных гиперссылок
`bgColor`	Цвет фона страницы. В Internet Explorer работает правильно, только если цвет страницы установлен атрибутом `BGCOLOR`
`body`	Возвращает ссылку на все содержимое тега `<BODY>`. Поддерживается только Internet Explorer начиная с 4.0
`fgColor`	Цвет текста. В Internet Explorer работает правильно, только если цвет текста установлен атрибутом `TEXT`
`fileCreatedDate`	Возвращает дату создания файла документа в строковом виде. Поддерживается только Internet Explorer начиная с 4.0
`fileModifiedDate`	Возвращает дату последнего изменения файла документа в строковом виде. Поддерживается только Internet Explorer начиная с 4.0
`fileSize`	Возвращает размер файла документа в строковом виде. Поддерживается только Internet Explorer начиная с 4.0
`height`	Высота документа в пикселах. Поддерживается только Navigator начиная с 4.0
`lastModified`	Возвращает дату последнего изменения документа в виде строки
`linkColor`	Цвет гиперссылок. В Internet Explorer работает правильно, только если цвет гиперссылок установлен атрибутом `LINK`
`readyState`	Статус документа. Возвращает одно из четырех значений: `"complete"` означает, что документ полностью загружен; `"interactive"` — загружен не полностью, но доступен для просмотра и управления; `"loading"` — загружается; `"uninitialized"` — недоступен, как правило, загружается. Доступно только для чтения и поддерживается лишь Internet Explorer начиная с 4.0
`referrer`	Возвращает адрес документа, с которого пользователь попал на текущий документ. Netscape выдает правильно значение, но Internet Explorer почему-то возвращает адрес текущего документа
`title`	Название документа, заданное в теге `<TITLE>`. Поддерживается только Navigator
`URL`	Адрес документа

Таблица 3.2 (окончание)

Свойство	Описание
vlinkColor	Цвет посещенных гиперссылок. В Internet Explorer работает правильно, только если цвет посещенных гиперссылок установлен атрибутом VLINK
wigth	Ширина документа в пикселах. Поддерживается только Navigator начиная с 4.0

Методы, поддерживаемые объектом document, перечислены в табл. 3.3.

Таблица 3.3. Методы объекта document

Метод	Описание
close()	Вызывает перерисовку окна после многократных вызовов методов write и writeln
elementFromPoint({X}, {Y})	Возвращает ссылку на элемент, находящийся по координатам X и Y. Поддерживается только Internet Explorer начиная с 4.0
getElementById({Имя элемента})	Возвращает элемент, имя которого передано в качестве параметра. Имя элемента страницы задается атрибутом ID. Поддерживается только Internet Explorer начиная с 5.0
getSelection()	Возвращает строку, содержащую текст, выделенный пользователем на странице. Поддерживается только Navigator начиная с 4.0
write({Текст})	Записывает текст, переданный как параметр, в текущее место документа. Текст может содержать любое HTML-форматирование
writeLn({Текст})	То же самое, что writeLn, но в конце добавляет символы возврата каретки и перевода строки. Но, т. к. эти символы игнорируются Web-обозревателями, никаких видимых отличий между этими методами нет

Я привел здесь только самые простые свойства и методы, оставив наиболее сложные для последующего рассмотрения. А пока обсудим еще одну важную тему, без понимания которой мы не продвинемся дальше.

Коллекции объектов

Что такое объекты, мы уже узнали. Мы также знаем, что такое массивы. Осталось поговорить о так называемых коллекциях объектов, имеющих признаки как массивов, так и объектов.

Коллекция — это своего рода массив объектов, проиндексированный не только по числовым номерам элементов, но и по их именам, и имеющий свойства и методы. Коллекция отличается от уже знакомого нам ассоциативного массива именно наличием свойств и методов, т. е. коллекция — сама по себе объект.

Рассмотрим, например, коллекцию `images`, которую включает в себя объект `document`. Вы можете получить доступ к отдельным ее элементам по порядковому номеру или уникальному имени.

```
document.images.item(1)
document.images.item("image1")
```

Заметьте, что мы указываем индекс элемента коллекции не в квадратных, а в круглых скобках, потому что он фактически является аргументом функции-метода `item`, поддерживаемой всеми коллекциями. Имя этого метода можно опускать, что вносит еще бóльшую путаницу.

```
document.images(1)
document.images("image1")
```

Но вы знайте, что `image` — не массив, а коллекция.

Все элементы Web-страницы нумеруются в порядке их появления в HTML-коде. А уникальные имена задаются специальным атрибутом `ID`, доступным практически во всех тегах. (Когда-то мы использовали его для присвоения стиля элементу.)

```
<IMG SRC="im1.gif" ID="image1">
<SCRIPT>
<!--
var someImage = document.images("image1");
-->
</SCRIPT>
```

Атрибут NAME

Для некоторых тегов (в частности, `<IMG>` и `<FRAME>`) доступен также атрибут `NAME`, выполняющий ту же функцию, что и `ID`. Атрибут `NAME` считается устаревшим и используется фактически только во фреймах.

Коллекция `all` представляет все элементы Web-страницы без исключений, в том числе и изображения. К элементам этой коллекции можно обращаться по номеру (нумеруются они в порядке появления в HTML-коде) или по имени.

```
document.all(8)
document.all("image1")
```

Атрибут `ID` требует уникальности имен всех элементов, но только формально. Реально же на странице могут встретиться два или более элемен-

тов с одинаковыми именами. В таком случае нужно будет указать второй индекс

```
document.all("image1", 2)
```

или выражение `document.all("image1")` вернет коллекцию всех изображений с именем `image1`.

Также все коллекции имеют свойство `length`, вызвращающее количество элементов коллекции. (Запомните, однако, что элементы коллекции нумеруются, начиная с нуля, поэтому последний ее элемент будет иметь номер `length-1`.)

Некоторые коллекции могут иметь методы для добавления и удаления элементов и других целей. Далее, при описании тех или иных коллекций я буду особо отмечать это.

Подчиненные объекты и коллекции объекта *document*

Объект `document` содержит внутри себя множество подчиненных объектов и коллекций. В табл. 3.4 перечислены свойства, возвращающие ссылки на них, и приведены описания соответствующих объектов и коллекций.

Таблица 3.4. *Подчиненные объекты и коллекции объекта* `document`

Свойство-ссылка	Описание
`all`	Все элементы страницы, включая теги <HTML>, <HEAD>, <TITLE> и <BODY>. Поддерживается только Internet Explorer начиная с 4.0
`anchors`	Все "якоря" страницы
`applets`	Все Java-апплеты, изображения и элементы ActiveX
`embeds`	Все расширения, внедренные в страницу
`forms`	Все Web-формы. (О формах см. главу 5.)
`frames`	Все фреймы набора фреймов. Как свойство объекта `document` поддерживается только Internet Explorer
`images`	Все изображения на странице
`layers`	Все слои страницы. Поддерживается только Navigator начиная с 4.0
`links`	Все гиперссылки на странице. В случае Navigator также все "горячие" области <AREA>

Таблица 3.4 (окончание)

Свойство-ссылка	Описание
location	Объект location для данного документа. (Об объекте location см. ниже.) Как свойство объекта document поддерживается только Internet Explorer
scripts	Все скрипты, внедренные в страницу. Поддерживается только Internet Explorer начиная с 4.0
selection	Объект selection, представляющий выделенный пользователем на странице текст. Поддерживается только Internet Explorer начиная с 4.0. В Navigator используется метод getSelection. (Об объекте selection см. далее в книге.)
styleSheets	Все таблицы стилей встроенные или привязанные к странице. Поддерживается только Internet Explorer начиная с 4.0

Теперь подробнее поговорим о коллекциях объекта document, с которыми мы только что познакомились.

Коллекция all имеет дополнительный метод tags, позволяющий фильтровать элементы коллекции по их тегу. В качестве параметра этот метод принимает строковое значение нужного тега.

```
document.all.tags("H1")
```

Данное выражение вернет ссылку на коллекцию, содержащую только заголовки первого уровня.

Такой же метод поддерживает коллекция links.

Здесь также нужно упомянуть два метода, поддерживаемых Internet Explorer начиная с 5.0. Это методы getElementByName и getElementByTagName. Первый из них возвращает коллекцию элементов со значением атрибута NAME, переданным в качестве параметра. Поскольку атрибут NAME применяется в современном HTML очень мало, польза от этого метода невелика. Второй метод возвращает коллекцию элементов, созданных посредством тега, переданного в качестве параметра.

```
document.getElementByName("someimage")
document.getElementByTagName("H1")
```

Обращение к элементам страницы

Объект document имеет несколько полезных методов (в частности, write), которые можно использовать в скриптах. Но главное его назначение — предоставление доступа к отдельным элементам Web-страницы. Именно поэтому мы и рассмотрели его в первую очередь, прежде всех других объектов.

Как же можно добраться до отдельных элементов страницы?

Прежде всего, через коллекции. Например, коллекцию `all`.

```
document.all("image1").outerHTML
```

Или через коллекцию `images`.

```
document.images("image1").outerHTML
```

Internet Explorer позволяет опускать ссылку на объект `document`. Так что можно будет написать и так:

```
all("image1").outerHTML
images("image1").outerHTML
```

Navigator делать так не позволяет; он требует обязательной ссылки на `document`. Вдобавок он не поддерживает коллекцию `all`.

```
document.images("image2").src
```

Как вы уже знаете, имя элемента Web-страницы задается с использованием атрибутов `ID` или `NAME`. Если каждый элемент страницы имеет уникальное имя, то можно использовать так называемое *прямое обращение* к элементам. То есть, обращаться к нему не как к элементу коллекции, а как к отдельному объекту. Это работает и в Internet Explorer, и в Navigator.

```
image1.outerHTML
layer1.top
```

Прямое обращение происходит быстрее, чем обращение через коллекцию. Поэтому его рекомендуется использовать во всех случаях, когда не нужно специальное обращение к коллекциям и когда каждый элемент страницы, к которому производится обращение из скриптов, имеет уникальное имя.

Элементы Web-страницы

В этом разделе мы рассмотрим, что можно делать с отдельными элементами Web-страницы после того, как к ним получен доступ через объект `document`.

Дело в том, что любой элемент страницы можно представить в виде объекта с набором свойств и методов. Полный список их приведен в приложении 3; просмотрев их, вы поразитесь, как велики возможности работы с обычным абзацем или рисунком. Многие свойства связаны с соответствующими атрибутами HTML-тега, образующего элемент, другие же вообще не имеют ничего общего с HTML. В дальнейшем не забывайте заглядывать в приложения, чтобы понимать, как работают приведенные в книге примеры.

И запомните: все это будет работать только в Internet Explorer. Объектная модель Navigator несравнимо беднее и позволяет программировать только слои, гиперссылки и некоторые другие объекты. Программирование скриптов для Navigator мы рассмотрим в конце этой главы.

Работа с элементами страницы

Так что же можно делать с элементами Web-страницы?

Все, что угодно.

Предположим, мы имеем на нашей Web-странице рисунок.

```
<IMG ID="someimage" SRC="someimage.gif" HEIGHT="30" WIDTH="50">
```

Из кода скрипта мы может изменить как сам рисунок, так и его геометрические размеры.

```
someimage.src = "anotherimage.gif";
someimage.height = 40;
```

Как видите, мы используем свойства объекта рисунка, имена которых совпадают с названиями соответствующих атрибутов тега `<IMG>`. Это сделано для упрощения программирования скриптов.

Имея гиперссылку, можно изменить адрес, на который она указывает.

```
<A ID="somelink" HREF="beyond.html">Куда-то</A>
somelink.href = "somewhere.html";
```

Можно присвоить текстовому абзацу другой стиль.

```
<P ID="par" CLASS="someclass">Некий текст</P>
par.className = "anotherclass";
par.style = "font-weight: normal";
```

Можно убрать рамку у таблицы.

```
<TABLE ID="sometable" BORDER="2">
sometable.border = 0;
```

Как видите, каждый элемент страницы представляет собой объект, имеющий множество свойств и методов. Я даже не буду приводить их здесь — просто загляните в главу 1, посвященную HTML, и посмотрите, что делает тот или иной атрибут того или иного тега. Или сразу обратитесь к приложению 3 — там описано абсолютно все, что предлагает тот или иной объект.

Кроме свойств, элементы страниц поддерживают некоторое количество методов. Как правило, это стандартные методы, поддерживаемые всеми элементами страницы независимо от тега (они будут описаны ниже). Свои собственные методы имеют только сложные элементы, такие как таблицы.

Например, метод `insertRow` позволит вставить в таблицу строку с заданным номером.

```
sometable.insertRow(4);
```

Это выражение вставит в таблицу `sometable` строку под номером 5. (Индексация начинается с нуля.)

Метод deleteRow удаляет из таблицы строку с заданным номером.

```
sometable.deleteRow(3);
```

Таблица как сложный объект поддерживает внутреннюю коллекцию rows, которую можно использовать для доступа к отдельным строкам.

```
sometable.rows(5).bgColor = "black";
```

В свою очередь строка таблицы тоже является сложным объектом и также поддерживает внутреннюю коллекцию cells.

```
sometable.rows(5).cells(2).bgColor = "white";
```

Как видите, объектная модель документа очень сложна, но и, вместе с тем, очень логично организована. Нет практически ничего, недоступного для выполнения в скрипте с тем или иным элементом Web-страницы. Остается только посетовать на неповоротливость фирмы Netscape, не желающей дать своей программе Web-обозревателя такую же мощную объектную модель, какую имеет Microsoft Internet Explorer.

Общие свойства и методы

Большинство общих свойств и методов, поддерживаемых всеми элементами страницы, перечислено в табл. 3.5 и 3.6 соответственно.

Таблица 3.5. *Общие свойства элементов Web-страницы*

Свойство	Описание
all	Возвращает ссылку на коллекцию дочерних элементов. То же самое, что аналогичное свойство объекта document
className	Класс свойств, присвоенный элементу Web-страницы
clientHeight	Возвращает высоту элемента без учета рамок, границ, отступов и полос прокрутки. Доступно только для чтения
clientLeft	Возвращает смещение левого края элемента относительно левого края родителя в пикселах без учета рамок, границ, отступов и полос прокрутки
clientTop	Возвращает смещение верхнего края элемента относительно верхнего края родителя в пикселах без учета рамок, границ, отступов и полос прокрутки
clientWidth	Возвращает ширину элемента без учета рамок, границ, отступов и полос прокрутки
height	Высота элемента. Определено только, если задано с помощью атрибута HEIGHT тега элемента
id	Имя элемента, заданное атрибутом ID

***Таблица 3.5** (окончание)*

Свойство	Описание
innerHTML	Все содержимое элемента: текст и теги дочерних элементов. Если присвоить этому свойству новое значение, все содержимое элемента будет заменено
innetText	Текстовое содержимое элемента, включающее и текст дочерних элементов, но исключая любые теги HTML. Если присвоить этому свойству новое значение, все содержимое элемента будет заменено
offsetHeight	Возвращает высоту элемента в пикселах относительно высоты родителя
offsetLeft	Возвращает смещение левого края элемента относительно левого края родителя в пикселах
offsetParent	Возвращает ссылку на родителя, относительно которого вычисляются значения свойств offset***
offsetTop	Возвращает смещение верхнего края элемента относительно верхнего края родителя в пикселах
offsetWidth	Возвращает ширину элемента в пикселах относительно ширины родителя
outerHTML	Все содержимое элемента: текст и теги дочерних элементов, включая теги, образующие этот элемент. Если присвоить данному свойству новое значение, все содержимое элемента и сам элемент будут заменены
outerText	Текстовое содержимое элемента, включающее и текст дочерних элементов, но исключая любые теги HTML. Если присвоить этому свойству новое значение, все содержимое элемента и сам элемент будут заменены
parentElement	Возвращает ссылку на родителя
readyState	Возвращает состояние элемента. Аналогично свойству readyState объекта document
scrollHeight	Возвращает полную высоту содержимого элемента
scrollLeft	Положение горизонтальной полосы прокрутки
scrollTop	Положение вертикальной полосы прокрутки
scrollWidth	Возвращает полную ширину содержимого элемента
sourceIndex	Возвращает порядковый номер элемента, который можно потом использовать для ссылки на элемент из коллекции all
tagName	Возвращает тег элемента без символов "<" и ">"
width	Ширина элемента. Определено только, если задано с помощью атрибута HEIGHT тега элемента

Таблица 3.6. *Общие методы элементов Web-страницы*

Метод	Описание
`clearAttributes()`	Удаляет все атрибуты тега элемента
`contains({Имя элемента})`	Возвращает `true`, если элемент с данным именем содержится внутри текущего элемента
`getAdjacentText ({Местонахождение})`	Возвращает текстовую строку, находящуюся в текущем элементе или рядом с ним. Параметр местонахождения может принимать следующие значения: `BeforeBegin` — текст, находящийся перед открывающим тегом элемента, `AfterBegin` — после открывающего тега, но перед всем содержимым текущего элемента, `BeforeEnd` — перед закрывающим тегом, но после всего содержимого, `AfterEnd` — после закрывающего тега
`getAttribute({Имя атрибута}, true|false)`	Возвращает значение атрибута тега текущего элемента. Если второй параметр установлен в `true`, поиск атрибута будет производиться с учетом регистра символов его имени. В зависимости от атрибута может вернуть значение разного типа
`getElementsByTagName({Тег})`	Возвращает указатель на коллекцию дочерних элементов, созданных с использованием тега, переданного в качестве параметра
`insertAdjacentHTML ({Местонахождение}, {Текст})`	Позволяет вставить текст внутрь текущего элемента. Текст может содержать HTML-форматирование. Параметр местонахождения может принимать следующие значения: `BeforeBegin` — текст вставляется перед открывающим тегом текущего элемента, `AfterBegin` — после открывающего тега, но перед всем содержимым текущего элемента, `BeforeEnd` — перед закрывающим тегом, но после всего содержимого, `AfterEnd` — после закрывающего тега
`insertAdjacentText ({Местонахождение}, {Текст})`	То же самое, что предыдущий метод, но при вставке текста игнорируются все HTML-теги
`removeAttribute ({Имя атрибута}, true|false)`	Удаляет атрибут у тега текущего элемента. Если второй параметр установлен в `true`, поиск атрибута будет производиться с учетом регистра символов его имени. Возвращает `true`, если удаление было выполнено успешно, и `false` — в противном случае

Таблица 3.6 (окончание)

Метод	Описание
`replaceAdjacentText ({Местонахождение}, {Текст})`	Позволяет заменить текст, находящийся в текущем элементе или рядом с ним, на другой текст, переданный в качестве второго параметра
`scrollIntoView (true|false)`	Вызывает прокрутку страницы в окне Web-обозревателя так, чтобы текущий элемент оказался в поле зрения. Если в качестве параметра передано `true`, то текущий элемент окажется у верхнего края окна, если `false` — у нижнего
`setAttribute({Имя атрибута}, {Значение атрибута}, true|false)`	Присваивает значение атрибуту тега текущего элемента. Если третий параметр установлен в `true`, поиск атрибута будет производиться с учетом регистра символов его имени

Использование общих свойств и методов

Для начала рассмотрим свойства `innerHTML`, `outerHTML`, `innerText` и `outerText` и методы `insertAdjacentHTML` и `insertAdjacentText`.

Предположим, что мы имеем элемент HTML, код которого представлен ниже.

```
<P ID="par1">Это элемент <B>1</B></P>
```

Четыре вышеперечисленных свойства вернут для него следующие значения:

- `innerHTML` — `"Это элемент <B>1</B>"`;
- `outerHTML` — `"<P ID="par1">Это элемент <B>1</B></P>"`;
- `innerText` и `outerText` — `"Это элемент 1"`.

Как видите, свойства `inner***` затрагивают только содержимое текущего элемента, а `outer***` — еще и теги самого элемента. Свойства `innerText` и `outerText` вернули нам текст элемента без тегов HTML, а `innerHTML` и `outerHTML` — вместе с ними.

Теперь давайте присвоим этим четырем свойствам одно и то же значение — `"А это элемент <I>2</I>"`. И посмотрим, что у нас получится.

```
document.all("par1").innerHTML = "А это элемент <I>2</I>";
<P ID="par1">А это элемент <I>2</I></P>
```

Предыдущее содержимое элемента было без всяких сюрпризов заменено на новое.

```
document.all("par1").outerHTML = "А это элемент <I>2</I>";
А это элемент <I>2</I>
```

А здесь вы видите новый текст, не заключенный ни в какой элемент страницы. Это значит, что элемент с именем par1 исчез из страницы, и все последующие попытки обратиться к нему из других скриптов обречены на провал (вернется значение null). Так что будьте осторожны, когда имеете дело со свойством outerHTML (и outerText тоже).

```
document.all("par1").innerText = "А это элемент <I>2</I>";
<P ID="par1">А это элемент &lt;I&gt;2&lt;/I&gt</P>
```

Да, именно так будет выглядеть наш элемент в этом случае. То есть, даже если и присвоить свойству innerText (и outerText тоже) какой-то HTML-код, он не будет обработан и отобразится Web-обозревателем как текст.

```
document.all("par1").outerText = "А это элемент <I>2</I>";
А это элемент &lt;I&gt;2&lt;/I&gt
```

Час от часу не легче!.. Теперь и сам элемент par1 пропал.

Заметьте, что при изменении значений свойств innerHTML и outerHTML Web-обозреватель может скорректировать мелкие ошибки во вновь добавляемом HTML-коде. Так, он добавляет закрывающие теги, если они были пропущены. (Правда, при этом может получиться не то, что вы ожидали.)

```
document.all("par1").innerHTML = "А это элемент <I>2";
<P ID="par1">А это элемент <I>2</I></P>
```

Теперь поговорим о "сладкой парочке" методов insertAdjacentHTML и insertAdjacentText. Давайте вставим в наш элемент новый текст "<BIG>I</BIG>" с разными значениями параметра местоположения.

```
document.all("par1").insertAdjacentHTML("BeforeBegin","<BIG>I</BIG>");
<BIG>I</BIG><P ID="par1">Это элемент <B>1</B></P>

document.all("par1").insertAdjacentHTML("AfterBegin","<BIG>I</BIG>");
<P ID="par1"><BIG>I</BIG>Это элемент <B>1</B></P>

document.all("par1").insertAdjacentHTML("BeforeEnd","<BIG>I</BIG>");
<P ID="par1">Это элемент <B>1</B><BIG>I</BIG></P>

document.all("par1").insertAdjacentHTML("AfterEnd","<BIG>I</BIG>");
<P ID="par1">Это элемент <B>1</B></P><BIG>I</BIG>
```

По-моему, логика работы этого метода должна быть понятна.

Метод insertAdjacentText ведет себя аналогично, за тем исключением, что игнорирует любое HTML-форматирование подобно свойствам innerText и outerText. Я не буду его рассматривать.

Имейте также в виду, что метод insertAdjacentHTML тоже может подвигнуть Web-обозреватель на исправление ошибок во вставляемом HTML-коде. И здесь риск получить совсем не то намного выше.

Вам могут пригодиться методы `getAdjacentText` и `replaceAdjacentText`. Первый из них возвращает текст, находящийся внутри элемента или рядом с ним, а второй позволяет заменить его на другой.

Теперь о методах `getAttribute`, `removeAttribute` и `setAttribute`. Первый из них позволит получить значение какого-либо атрибута тега текущего элемента.

```
bodyColor = document.body.getAttribute("BGCOLOR", false);
```

Второй параметр установлен в `false`; это значит, что поиск атрибута будет производиться без учета регистра символов его имени.

Метод `setAttribute` в свою очередь позволит вам дать атрибуту новое значение.

```
document.body.setAttribute("BGCOLOR", "teal", false);
```

А метод `removeAttribute` позволит вообще удалить данный атрибут из тега элемента, тем самым сбросив его в значение по умолчанию.

```
document.body.removeAttribute("BGCOLOR", false);
```

И немного о свойствах `height` и `width`. Дело в том, что их значения определены, только если они установлены с помощью соответствующих атрибутов тега.

```
<IMG SRC="image1.gif" HEIGHT="100" WIDTH="100">
nheight = image1.height;
```

Если же высота и ширина элемента заданы через стиль, значения свойств `height` и `width` не определены.

```
<IMG SRC="image1.gif" STYLE="height: 100; width: 100">
nheight = image1.height; // Не будет работать!
// В этом случае используйте объект style (см. его описание ниже)
nheight = image1.style.height; // Вот так
```

Метод `clearAttributes` позволит вам удалить сразу все атрибуты тега элемента и, естественно, все их значения. Это может пригодиться, если вы хотите полностью изменить внешний вид какого-либо элемента.

Два примера Web-страниц с изменяемым содержимым

Чтобы проиллюстрировать вышеописанное, рассмотрим два примера динамичных Web-страниц.

Новая модификация страницы с датой

Давайте еще раз изменим нашу первую динамичную страницу. Поскольку модификация будет весьма серьезная, рассмотрим ее код построчно.

```
<HTML>
<HEAD>
<TITLE>Сегодня</TITLE>
<SCRIPT>
function writeDate() {
var d;
d=new Date();
dateishere.innerText = d.toLocaleDateString(); }
</SCRIPT>
```

Это уже знакомая вам функция вставки даты. В новом ее варианте мы использовали свойство `innerText`, чтобы вставить значение даты внутрь элемента `dateishere`. Заметьте, что мы обращаемся к элементу `dateishere` напрямую. Также мы использовали метод `toLocaleDateString` класса даты вместо `toString`, т. к. первый возвращает дату, сообразуясь с интернациональными установками системы.

```
</HEAD>
<BODY onload="writeDate()">
```

Здесь вы видите нечто, похожее на новый атрибут `onload`. Однако это совсем не атрибут. Мы поговорим о нем позже, когда начнем обсуждать события и их обработку, а пока имейте в виду, что строка `onload="writeDate()"` вызывает выполнение функции `writeDate()` после загрузки страницы.

```
<P ID="dateishere">Здесь будет дата</P>
```

Мы вставили в элемент `dateishere` текст по умолчанию, чтобы отобразить его, если вдруг скрипт не сработает. Если же он сработает, этот текст по умолчанию будет заменен датой, вставленной скриптом.

```
</BODY>
</HTML>
```

Сохраним нашу новую страницу в файле 3.2.htm и откроем ее в Web-обозревателе. Результат виден на рис. 1.17.

"Всплывающие" подсказки

А теперь создадим наконец-то нечто более полезное.

Вы, может быть, видели на некоторых сайтах, как при подведении курсора мыши к какому-либо элементу страницы (картинке или гиперссылке) рядом с ней высвечивается описание или пояснение. Такие штуки называются

"всплывающими" подсказками. И сейчас мы сделает страничку с чем-то подобным.

Пусть подсказки высвечиваются после щелчков на гиперссылках. Таким образом, мы упростим наш и без того довольно сложный пример. Создадим четыре гиперссылки — этого вполне хватит. Ведь мы же пока еще не собираемся делать сайт...

Код нашей страницы будет столь сложен, что мы впервые в этой книге разобьем его на несколько отдельных частей и рассмотрим их по отдельности. И начнем с HTML-кода страницы.

Сама страница

Нашу страницу мы оформим в виде таблицы. Так будет проще и понятнее. Не будем применять к элементам страницы никаких дополнительных стилей с целью сделать ее красивее. Я уверен, что вы сможете это сделать и без моей помощи; моя же задача — научить вас менять содержимое страницы в ответ на действия пользователя.

Рассмотрим код страницы построчно.

```
<HTML>
<HEAD>
<TITLE>Справочник</TITLE>
<!-- Здесь у нас со временем будут скрипты -->
</HEAD>
<BODY>
<TABLE>
<TR>
<TD VALIGN="top" BGCOLOR="#FFFFE0" WIDTH="100">
```

Все должно быть вам знакомо. Мы создаем таблицу, в которой и будет помещаться все содержимое страницы. Обычнейшая практика в Web-дизайне...

```
<P><A HREF="javascript:showTopic(1)">Пункт 1</A></P>
```

А вот здесь остановимся. Вы видите, что в тег гиперссылки в качестве значения параметра `HREF` вписан уж никак не интернет-адрес, а что-то, напоминающее выражение JavaScript. Это и есть выражение JavaScript, которое выполнится при щелчке на гиперссылке. Единственное требование: оно должно предваряться идентификатором скриптового языка `"javascript"`, иначе Web-обозреватель начнет искать документ с таким интернет-адресом.

По поводу Java-кода вопросов быть не должно. Это обычный вызов функции (правда, пока еще не определенной нами) с одним параметром.

```
<P><A HREF="javascript:showTopic(2)">Пункт 2</A></P>
<P><A HREF="javascript:showTopic(3)">Пункт 3</A></P>
<P><A HREF="javascript:showTopic(4)">Пункт 4</A></P>
```

Остальные гиперссылки создаются по образу и подобию первой.

```
</TD>
<TD WIDTH="20">
</TD>
```

Пустую ячейку таблицы мы создали только для того, чтобы разделить столбец гиперссылок и столбец описаний. Заметьте, что цвет его фона не задан, т. е. совпадает с цветом фона страницы.

```
<TD BGCOLOR="#FFFFE0" WIDTH="*">
<P ID="desc">Здесь будут появляться описания гиперссылок.</P>
</TD>
```

А в этом столбце будут появляться описания гиперссылок, на которых щелкает пользователь. Внутри столбца находится абзац с именем `desc`, в котором и будут появляться описания. А пока что там помещен некий начальный текст.

```
</TR>
</TABLE>
</BODY>
</HTML>
```

Вот и все. За исключением некоторых моментов, которые мы рассмотрели по ходу создания нашей страницы, здесь нет ничего особенно сложного. Страница как страница.

Скрипт

А теперь добавим к нашей странице скрипт, "отвечающий" за динамику. Вдохнем жизнь, так сказать...

Скрипт вставляется в заголовок страницы, где мы специально зарезервировали для него местечко. (И поставили поясняющий комментарий, чтобы не забыть об этом.)

```
<SCRIPT>
<!--
```

Так или иначе, а Navigator этот код все равно не поддерживает: свойство `innerHTML` ему не знакомо. Так что заключим наш скрипт в комментарий.

```
function showTopic(n) {
  var cdesc;
  switch (n) {
```

```
    case 1 :
      cdesc = "Пункт первый: <BIG>большой</BIG> и <B>страшный</B>.";
      break;
    case 2 :
      cdesc = "Пункт второй: <SMALL>маленький</SMALL> и <I>смешной</I>.";
      break;
    case 3 :
      cdesc = 'Пункт третий: <FONT COLOR="red">внимание</FONT>!';
      break;
    case 4 :
      cdesc = 'Пункт четвертый: <FONT COLOR="green">зеленая</FONT>
      ↳тоска...';
      break;
    default :
      cdesc = "<U>По гиперссылкам</U> надо щелкать!";
  }
  desc.innerHTML = cdesc;
}
-->
</SCRIPT>
```

Ну вот и все. Комментировать здесь особо нечего: функция просто проверяет переданный ей параметр и в зависимости от его значения присваивает внутренней переменной `cdesc` одну из предопределенных строк, которые и являются описаниями наших гиперссылок. После этого значение `cdesc` присваивается свойству `innerHTML` элемента `desc`.

Теперь объединим код страницы и код скрипта, сохраним в файле под именем 3.3.htm и откроем в Web-обозревателе. То, что мы увидим, показано на рис. 3.1.

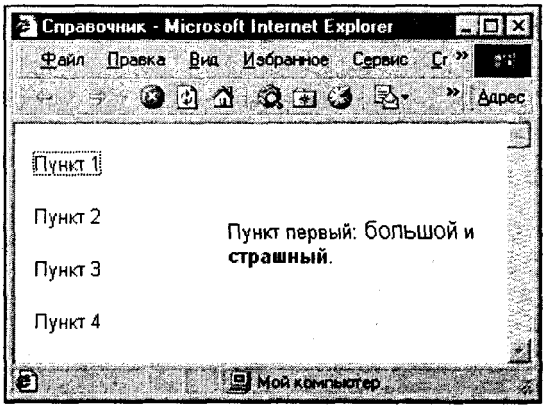


Рис. 3.1. Страница с описаниями гиперссылок

Объект *location*

При описании объекта `document` было сказано, что у него имеется свойство `location`, возвращающее ссылку на объект `location` для данного документа. Пришла пора объяснить, что это такое.

Объект `location` содержит информацию о местонахождении текущего документа, т. е. его интернет-адресе. Его также можно использовать для перехода на другой документ и перезагрузки текущего документа.

Свойства и методы

Объект `location` поддерживает ряд свойств и методов. Все поддерживаемые им свойства перечислены в табл. 3.7, а методы — в табл. 3.8.

Таблица 3.7. *Свойства объекта* `location`

Свойство	Описание
`hash`	Имя "якоря" в интернет-адресе документа, если оно есть
`host`	Имя компьютера в сети Интернет вместе с номером порта, если он указан
`hostname`	Имя компьютера в сети Интернет
`href`	Полный интернет-адрес документа
`pathname`	Путь и имя файла, если они есть
`port`	Номер порта. Если не указан, возвращается номер `80` — стандартный порт, через который работает протокол HTTP
`protocol`	Идентификатор протокола. Если не указан, возвращается "`http:`"
`search`	Строка параметров, если она есть

Таблица 3.8. *Методы объекта* `location`

Метод	Описание
`assign({Адрес})`	Загружает документ, адрес которого передан в качестве параметра. Поддерживается только Internet Explorer начиная с 4.0
`reload([true|false])`	Перезагружает документ с Web-сервера. Необязательный параметр нужен только для Navigator; значение `true` заставляет Web-обозреватель перезагрузить документ с жесткого диска, где он был сохранен ранее, а `false` (значение по умолчанию) — прямо с Web-сервера

Таблица 3.8 (окончание)

Метод	Описание
`replace({Адрес})`	Загружает документ, адрес которого передан в качестве параметра, и заменяет в списке истории Web-обозревателя адрес предыдущего документа адресом нового

Вот такие свойства и методы. Осталось еще немного рассказать об интернет-адресах.

Пользуясь объектом `location`, можно загрузить другой документ на место текущего. Для этого просто присвойте значение нового интернет-адреса свойству `href`.

```
document.location.href = "http://www.gothic.ru";
```

Пользователи Internet Explorer также могут воспользоваться методом `assign`.

```
document.location.assign("http://www.gothic.ru");
```

Если вы хотите полностью заменить текущий документ, чтобы даже адрес его не появлялся в списке истории, воспользуйтесь методом `replace`.

```
document.location.replace("http://www.gothic.ru");
```

Новое об интернет-адресах

В главе 1 мы изучили четыре составных части интернет-адреса:

1. Необязательное обозначение протокола (http://).
2. Доменный или IP-адрес компьютера, на котором работает программа Web-сервера (www.coolsite.ru).
3. Необязательное имя файла, включающее или не включающее путь (/folder1/folder2/file.html).
4. Необязательное имя "якоря" (#chapter_2). Всегда предваряется символом "решетки" "#".

На самом деле, интернет-адрес может иметь еще две необязательных части: номер порта и строку параметров. О них-то мы сейчас и поговорим.

Порт TCP/IP — это своего рода виртуальный канал передачи данных. Всего существует более 65 000 портов, которые можно использовать для пересылки данных. Каждый интернет-протокол (HTTP, FTP, почта, новости и пр.) используют какой-либо порт, например для HTTP выделен порт 80, для FTP — порт 21 и т. п. Это порты по умолчанию, используемые, если специально не указан другой порт (что, впрочем, бывает довольно редко). Чтобы указать

другой порт для передачи данных какого-либо стандартного протокола, его номер указывают сразу после адреса компьютера, отделив двоеточием.

```
http://www.something.ru:5674/
```

Web-сервер может не только посылать пользователям в ответ на запросы файлы Web-страниц, но и сам генерировать эти страницы, используя специальные программы, называемые *серверными* (или программами стороны сервера). Эти программы помещены на сервере. Последний в ответ на запрос пользователя запускает такую программу, передает ей некоторые параметры, принимает результат работы — HTML-код — и посылает его пользователю. Вы понимаете? Страницу генерирует "на лету" сам сервер!

Зачем это может понадобиться? Например, для создания Web-сайтов, отображающих информацию из базы данных, генерирующих специальные страницы, основываясь на характеристиках клиента, да и во всех случаях, когда жестко заданных Web-страниц недостаточно. Скажем, интернет-магазины бóльшую часть своих страниц генерируют на сервере с помощью серверных программ.

Я не буду рассматривать в этой книге написание серверных программ, т. к. это довольно сложная тема, которой впору посвящать цикл книг. Я хочу только сказать, что параметры, которые могут быть переданы серверной программе, включаются в интернет-адрес. Они размещаются в виде строки в самом конце адреса, отделенные от него вопросительным знаком.

```
http://www.supershop.ru/catalogs/search.exe?good=tapochkifortarakans
```

Здесь мы видим, что в интернет-адресе указан путь к обычному исполняемому файлу — это и есть серверная программа. Сразу после пути находится строка параметров вида "имя параметра=значение параметра". Именно такой вид строки параметров применяется в системах поиска. Заметьте также, что далеко не все символы могут быть включены в строку параметров (помните, что она — часть интернет-адреса), скажем, пробелы, русские буквы, кавычки недопустимы. Такие символы кодируются их кодами, например, пробел будет выглядеть как "%20".

Итак, давайте перечислим все части интернет-адреса, которые мы теперь знаем.

1. Необязательное обозначение протокола (http://).
2. Доменный или IP-адрес компьютера, на котором работает программа Web-сервера (www.coolsite.ru).
3. Необязательный номер порта, через который передаются данные, если он отличается от порта по умолчанию для данного протокола. Отделяется от адреса компьютера двоеточием (:5764).
4. Необязательное имя файла, включающее или не включающее путь (/folder1/folder2/file.html).

5. Необязательное имя "якоря" (#chapter_2). Всегда предваряется символом "решетки" "#".
6. Необязательная строка параметров, передаваемая серверной программе. Отделяется от остального адреса вопросительным знаком (?good=tapochkifortarakans).

Объект *style*

Объект `style` представляет стиль элемента Web-страницы.

Свойства и методы

Как и все остальные объекты, `style` поддерживает ряд свойств и методов. Их можно разделить на две группы: задающие стиль элемента и относящиеся к самому объекту `style`. Свойства первой группы в целом аналогичны соответствующим атрибутам стиля и имеют почти такие же имена за тем исключением, что символы "-" убираются, т. к. не соответствуют соглашению об именах JavaScript, а первые буквы всех слов, образующих имя атрибута, кроме первого, делаются прописными. В табл. 3.9 показаны примеры преобразования имен атрибутов стиля в имена свойств объекта `style`, устанавливающих стиль элемента.

***Таблица 3.9.** Преобразование атрибутов стиля в свойства объекта `style`*

Атрибут стиля	Свойство объекта
`background-attachment`	`backgroundAttachment`
`border-bottom-color`	`borderBottomColor`
`font-family`	`fontFamily`
`z-index`	`zIndex`

По аналогии вы можете преобразовать все атрибуты стилей в свойства объекта `style`. Это просто.

Все не относящиеся к стилю свойства объекта `style` перечислены в табл. 3.10. Кроме того, у этого объекта имеются методы `getAttribure`, `removeAttribute` и `setAttribute`. Но учтите, что эти дополнительные свойства и методы поддерживаются только Internet Explorer начиная с 4.0.

***Таблица 3.10.** Свойства объекта `style`*

Свойство	Описание
`cssText`	Текстовое представление стиля (параметр атрибута `STYLE`)

Таблица 3.10 *(окончание)*

Свойство	Описание
pixelHeight	Высота элемента в пикселах
pixelLeft	Смещение левого края элемента в пикселах
pixelTop	Смещение верхнего края элемента в пикселах
pixelWidth	Ширина элемента в пикселах
posHeight	Высота элемента в тех единицах измерения, в которых она была установлена в определении стиля
posLeft	Смещение левого края элемента в тех единицах измерения, в которых оно было установлено в определении стиля
posTop	Смещение верхнего края элемента в тех единицах измерения, в которых оно было установлено в определении стиля
posWidth	Ширина элемента в тех единицах измерения, в которых она была установлена в определении стиля

Работа с объектом *style*

Объект `style` позволяет изменить стиль любого элемента Web-страницы, просто присвоив нужному свойству необходимое значение.

```
paragraph1.style.fontSize = 7;
```

Можно изменить геометрические размеры элемента:

```
image1.style.height = "100mm";
image1.style.width = "120mm";
```

и его местоположение:

```
image1.style.top = "200px";
image1.style.left = "50px";
```

Заметьте, что мы присваиваем свойству строковые значения геометрических параметров с указанием единицы измерения. Это не очень удобно для вычислений, поэтому объект `style` предоставляет свойства `pixel***`, принимающие и возвращающие числовые значения в пикселах:

```
image1.style.pixelHeight = 400;
image1.style.pixelLeft += 5;
```

Также можно использовать свойства `pos***`, возвращающие и принимающие числовые значения в тех единицах измерения, в которых эти значения были заданы в определении стиля.

```
<IMG SRC="image1.gif" ID="image1" STYLE="height: 100mm; width: 100">
nheight = image1.style.posHeight; // Значение в миллиметрах
nwidth = image1.style.posWidth; // Значение в пикселах
```

Вы можете использовать методы `getAttribute`, `setAttribute` и `removeAttribute` для получения и установки значения и удаления какого-либо свойства стиля.

```
paragraph1.style.setAttribute("borderBottomWidth", 5, false);
paragraph1.style.removeAttribute("borderTopWidth", false);
```

Объект *style* в Internet Explorer

Как же получить доступ к объекту `style`? Internet Explorer предоставляет для этого три свойства: `style`, `currentStyle` и `runtimeStyle`.

Первое свойство позволяет получить доступ к стилю, встроенному в тег элемента с помощью атрибута `STYLE`:

```
<P ID="par1" STYLE="color: green">Некий текст</P>
someColor = par1.style.color;
```

Вышеприведенное выражение поместит в переменную `someColor` значение атрибута `color` встроенного стиля элемента, т. е. `"green"`.

```
someFontSize = par1.style.fontSize;
```

Это выражение вернет `null`, даже если где-то в таблице стилей атрибут `font-size` для этого элемента определен. А все потому, что интересы свойства `style` не выходят за рамки встроенного стиля.

Чтобы получить стиль элемента с учетом и встроенных стилей, и таблиц стилей, и атрибутов тега, используйте свойство `currentStyle`:

```
<FONT ID="par1" STYLE="color: green" SIZE="7">Некий текст</FONT>
someColor = par1.currentStyle.color;
someBColor = par1.currentStyle.backgroundColor;
someFontSize = par1.currentStyle.fontSize;
someOther = par1.currentStyle.textDecoration;
```

Предположим, что где-то в таблице стилей определен для этого элемента атрибут `background-color`, равный `"teal"`, а `text-decoration` не определен вообще. Эти выражения вернут следующие значения: первое — `"green"`, второе — `"teal"`, третье — `"largest"` (или `"7"`; это значение, определенное в атрибуте `SIZE` тега), а четвертое — внимание! — `"none"`, т. е. значение по умолчанию для этого атрибута стиля. Одним словом, `currentStyle` собирает все в одну кучу.

Если вы измените какое-либо свойство объекта `style`, изменится то же свойство и у объекта `currentStyle`, и наоборот. Единственная деталь: если

вы изменили какое-либо свойство currentStyle и потом сразу же обратились к нему, вернется старое значение. То есть, между присвоением значения и его применением к элементу должно пройти некоторое время.

Свойство runtimeStyle довольно странное. Оно возвращает ссылку на объект runtimeStyle, который является примерно тем же самым, что и currentStyle, но присвоение его свойствам новых значений не затрагивает аналогичных свойств style. То есть, вы можете переопределить свойства стиля runtimeStyle, и соответствующие свойства style (но не currentStyle) не изменятся.

Пример страницы

Давайте рассмотрим пример Web-страницы, содержащей элемент, которым можно управлять уже знакомым нам способом — щелкая на гиперссылках. Можно изменять некоторые свойства его текста: "жирность" и курсивное начертание шрифта, подчеркивание текста. Давайте сделаем еще вот что: после щелчка на соответствующей гиперссылке зачеркнем ее, говоря таким образом пользователю, что данное свойство уже установлено. При повторном же щелчке мы уберем это свойство и снимем зачеркивание с гиперссылки.

Сама страница

Здесь мы построчно рассмотрим HTML-код самой страницы.

```
<HTML>
<HEAD>
<TITLE>Изменяющиеся стили</TITLE>
<STYLE>
```

Так как мы собираемся пользоваться объектом style, нам понадобится таблица стилей, чтобы задать начальные стилевые установки. В противном случае нам придется добавлять соответствующие свойства методом setAttribute, что неудобно, т. к. нужно будет помнить, какое свойство стиля добавлено, а какое — еще нет.

```
a { text-decoration: none }
```

Начальные установки текста гиперссылок. (Вспомните, ведь мы будем их зачеркивать, а зачеркивание текста задается именно этим свойством стиля.)

```
#sample { font-weight: normal;
  font-style: normal;
  text-decoration: none }
```

А это — наш пример текста, над которым мы будем экспериментировать. Изначально он имеет обычную "жирность", обычное начертание и никаких

украшений в виде подчеркивания. Заметьте, как мы используем для привязки стиля к элементу страницы его уникальный идентификатор.

```
</STYLE>
<!-- Здесь у нас со временем будут скрипты -->
</HEAD>
<BODY>
<TABLE>
<TR>
<TD VALIGN="top" BGCOLOR="#FFFFE0" WIDTH="100">
```

Оформляем нашу страничку в виде таблицы.

```
<P><A ID="hbold" HREF="javascript:switchBold()">Полужирный</A></P>
<P><A ID="hitalic" HREF="javascript:switchItalic()">Курсив</A></P>
<P><A ID="hunder" HREF="javascript:switchUnder()">Подчеркнутый</A></P>
```

Это гиперссылки, включающие или отключающие какой-либо из параметров текста нашего элемента `sample`. Так как нам придется в дальнейшем изменять и их стиль (зачеркивать), то мы даем им идентификаторы.

```
</TD>
<TD WIDTH="20">
</TD>
```

Это пустой столбец таблицы, играющий роль разделителя.

```
<TD BGCOLOR="#FFFFE0" WIDTH="*">
<P ID="sample">Здесь вы можете видеть результат щелчков
по гиперссылкам.</P>
```

Это примерный текст, над которым мы будем экспериментировать.

```
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

Вот и вся страница.

Скрипты

А здесь мы рассмотрим скрипты, которые нужно вставить в страницу на зарезервированное ранее место.

```
<SCRIPT>
<!--
```

Эта страница не будет поддерживаться Navigator, так что со спокойной душой можно поместить скрипт в комментарий.

```
var isBold = false;
var isItalic = false;
var isUnder = false;
```

Здесь мы объявляем три глобальные переменные. Они будут хранить данные о том, является ли наш примерный текст жирным, курсивом или подчеркнутым. Изначально они все установлены в `false`, это значит, что текст не жирный, не курсив и не подчеркнут.

Ниже приведены три функции, включающие и отключающие "жирность", курсивное начертание и подчеркивание соответственно. Мы рассмотрим подробно одну из них. Для остальных приведем лишь текст. Все они организованы и работают аналогичным образом.

```
function switchBold() {
```

Эта функция включает или отключает "жирность" шрифта. Ее-то мы и рассмотрим.

```
isBold = ! isBold;
```

Инвертирует текущее значение глобальной переменной `isBold`.

```
if (isBold) {
  sample.style.fontWeight = 'bold';
  hbold.style.textDecoration = 'line-through'; }
```

Если значение `isBold` равно `true`, делаем жирным шрифт примерного элемента и зачеркнутым — шрифт соответствующей гиперссылки.

Заметьте, что мы используем объект `style`. Это сделано для того, чтобы наша страница была совместима и с более старыми версиями Internet Explorer, т. к. объекты `currentStyle` и `runtimeStyle` появились только в версии 5.0.

```
else {
  sample.style.fontWeight = 'normal';
  hbold.style.textDecoration = 'none'; } }
```

В противном случае убираем "жирность" шрифта примерного элемента и зачеркивание — у шрифта гиперссылки.

```
function switchItalic() {
  isItalic = ! isItalic;
  if (isItalic) {
    sample.style.fontStyle = 'italic';
    hitalic.style.textDecoration = 'line-through'; }
  else {
    sample.style.fontStyle = 'normal';
    hitalic.style.textDecoration = 'none'; } }

function switchUnder() {
  isUnder = ! isUnder;
```

```
  if (isUnder) {
    sample.style.textDecoration = 'underline';
    hunder.style.textDecoration = 'line-through'; }
  else {
    sample.style.textDecoration = 'none';
    hunder.style.textDecoration = 'none'; } }
-->
</SCRIPT>
```

Объедините HTML-код страницы с кодом скриптов, сохраните в файле 3.4.htm и откройте в Web-обозревателе. Пощелкайте по гиперссылкам и посмотрите, что происходит с примерным текстом. Должно получиться что-то, похожее на рис. 3.2.

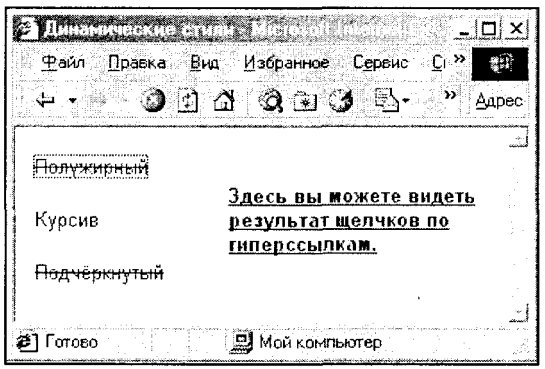


Рис. 3.2. Страница с изменяющимися стилями

Объект *styleSheet* и коллекция *styleSheets*

Объект `document` в Internet Explorer имеет встроенную коллекцию `styleSheets`, которую можно использовать для доступа к отдельным таблицам стилей документа. А отдельная таблица стилей представляется объектом `styleSheet`:

```
address = document.styleSheets(1).href;
```

Для нас будут полезны три свойства объекта таблицы стилей. Первое из них — это `href`, представляющее интернет-адрес файла внешней таблицы стилей. Вы можете изменить этот адрес, чтобы загрузить другую таблицу стилей. Второе — `disabled`, разрешающее или запрещающее Web-обозревателю применять эту таблицу для форматирования документа. Заметьте, что в данном случае `false` разрешает применение таблицы, а `true` запрещает. И последнее свойство — `type` — имеет для нас только теоретический интерес: оно задает тип таблицы стилей, который для Internet Explorer всегда равен "`text/css`".

Объект *style* в Navigator. JavaScript-стили

Теперь пора обратить наше благосклонное внимание на Navigator. Фирма Netscape серьезно поработала над своим Web-обозревателем, чтобы сделать его максимально нестандартным с точки зрения поддержки спецификаций W3C. Да и мощь его объектной модели, мягко говоря, слабовата. Неудивительно, что мы до сих пор игнорировали эту живую легенду (или полуживое ископаемое — как кому нравится) Интернета.

Но сейчас я положу этому замалчиванию конец. Я расскажу об особенностях поддержки объекта `style` в Navigator и новой разновидности таблиц стилей, не поддерживаемой Internet Explorer.

Как вы уже знаете, Navigator поддерживает прямое обращение по идентификаторам только для объектов `layer` и не поддерживает коллекцию `all`. Да и поддержка атрибута `ID` в нем реализована так себе, в основном, для присвоения элементам стилей. Но объект `document` Navigator предусматривает четыре очень мощных свойства для работы со стилями. И сейчас мы их рассмотрим.

Свойство `classes` позволит вам получить доступ к отдельному поименованному стилю.

```
document.classes.{Имя стиля}.{Имя тега}|all.{Имя свойства}
```

Здесь `{Имя тега}` может быть именем любого тега. Если нужно определить стиль для всех тегов, используйте слово `all`.

Например, предположим, что вы определили в таблице стилей некий стиль.

```
.somestyle { color: green }
```

Тогда в коде скрипта вы сможете получить к нему доступ.

```
document.classes.somestyle.all.fontFamily = "Arial";
```

Аналогично вы можете определить какой-либо стиль для одного определенного тега.

```
P.somestyle { color: green }
document.classes.somestyle.P.fontFamily = "Arial";
```

В свою очередь свойство `ids` позволит вам получить доступ к стилю, присвоенному отдельному идентификатору `ID`.

```
document.ids.{Идентификатор}.{Имя свойства}
```

И, как обычно, пример:

```
#someid { color: green }
document.ids.someid.fontFamily = "Arial";
```

Но что делать, если вы переопределили стиль какого-либо тега? В этом случае воспользуйтесь свойством tags.

```
document.tags.{Тег}.{Имя свойства}
```

Пример использования свойства:

```
H1 { color: green }
document.tags.H1.fontFamily = "Courier";
```

А если нам нужно, скажем, изменить стиль элемента <B>, находящегося внутри заголовка второго уровня? Для этого существует свойство contextual:

```
document.contextual({Контекст 1}[, {Контекст 2}[, ...]]).{Имя свойства}
```

Тогда для описанного нами случая:

```
document.contextual(document.tags.H2, document.tags.B).fontStyle =
↳"oblique";
```

что аналогично заданию в таблице стилей

```
H2 B { fontStyle: oblique }
```

Но если вы думаете, что можете динамически изменять внешний вид элементов страницы в Navigator путем изменения их стилей, вы жестоко ошибаетесь. Реально объект style нужен Navigator только для поддержки нового вида таблиц стилей, так называемых JavaScript-стилей. И эти стили мы сейчас рассмотрим.

JavaScript-стили отличаются от обычных тем, что больше всего похожи на скрипты. Собственно, это и есть скрипты. Давайте рассмотрим два примера: обычной таблицы стилей и таблицы JavaScript-стилей.

```
<STYLE TYPE="text/css">
P { color: blue }
.bigtext { font-size: 72pt;
  font-weight: boldest }
H1 B { text-decoration: underline }
</STYLE>
```

Это обычная таблица стилей. (Собственно, атрибут TYPE здесь указывать было необязательно, т. к. сейчас ему присвоено значение по умолчанию.)

Теперь преобразуем ее в таблицу JavaScript-стилей.

```
<STYLE TYPE="text/javascript">
document.tags.P.color = "blue";
with (document.classes.bigtext.all) {
  fontSize = "72pt";
  fontWeight = "boldest" }
```

```
document.contextual(document.tags.H1, document.tags.B).textDecoration =
↳"underline";
</STYLE>
```

Теперь вы видите, как таблица JavaScript-стилей похожа на скрипт. Заметьте также, что в этом случае атрибут `TYPE` обязателен, т. к. он указывает тип таблицы стилей.

Во всех предыдущих примерах для Navigator мы всегда приводили ссылку на объект `document`. Но в таблицах JavaScript-стилей ее можно опустить.

```
<STYLE TYPE="text/javascript">
tags.P.color = "blue";
with (classes.bigtext.all) {
  fontSize = "72pt";
  fontWeight = "boldest" }
contextual(tags.H1, tags.B).textDecoration = "underline";
</STYLE>
```

Объект *window*

Объект `window` представляет текущее окно Web-обозревателя или отдельный фрейм, если окно разделено на фреймы.

Свойства и методы

Объект `window` имеет множество свойств и методов. Все поддерживаемые им свойства перечислены в табл. 3.11, а методы — в табл. 3.12.

Таблица 3.11. *Свойства объекта* `window`

Свойство	Описание
`closed`	Возвращает `true`, если текущее окно закрыто. Может быть использовано при работе с несколькими окнами
`defaultStatus`	Сообщение по умолчанию, отображаемое в строке состояния окна
`document`	Возвращает ссылку на документ, загруженный в текущее окно
`frames`	Возвращает ссылку на коллекцию фреймов. (О коллекции `frames` см. ниже.)
`history`	Возвращает ссылку на объект истории Web-обозревателя. (Об объекте `history` см. ниже.)
`innerHeight`	Возвращает высоту клиентской области окна (без рамок, меню, панелей инструментов) в пикселах. Поддерживается только Navigator начиная с 4.0

Таблица 3.11 *(продолжение)*

Свойство	Описание
innerWidth	Возвращает ширину клиентской области окна (без рамок, меню, панелей инструментов) в пикселах. Поддерживается только Navigator начиная с 4.0
length	Возвращает количество фреймов
location	Возвращает ссылку на объект location документа, загруженного в текущее окно
locationbar	Возвращает ссылку на объект панели адреса окна Navigator (она же Location Toolbar). Единственное свойство этого объекта visible позволит показать ее или убрать; значение true этого свойства показывает панель адреса, false убирает. Поддерживается только Navigator начиная с 4.0
menubar	Возвращает ссылку на объект меню окна Navigator. Единственное свойство этого объекта visible позволит показать ее или убрать; значение true этого свойства показывает панель адреса, false убирает. Поддерживается только Navigator начиная с 4.0
name	Возвращает имя окна или фрейма
navigator	Возвращает ссылку на объект Web-обозревателя. (Об объекте navigator см. ниже.)
opener	Возвращает ссылку на окно, которое открыло текущее окно, например, методом open
outerHeight	Возвращает полную высоту окна (с рамками, меню, панелями инструментов) в пикселах. Поддерживается только Navigator начиная с 4.0
outerWidth	Возвращает полную ширину окна (с рамками, меню, панелями инструментов) в пикселах. Поддерживается только Navigator начиная с 4.0
pageXOffset	Возвращает расстояние по горизонтали между текущей позицией окна и левой границей документа. При прокручивании содержимого окна вправо значение этого свойства увеличивается, влево — уменьшается. Поддерживается только Navigator начиная с 4.0
pageYOffset	Возвращает расстояние по вертикали между текущей позицией окна и верхней границей документа. При прокручивании содержимого окна вниз значение этого свойства увеличивается, вверх — уменьшается. Поддерживается только Navigator начиная с 4.0
parent	Возвращает ссылку на родительское окно, если текущий объект window представляет собой фрейм. В противном случае возвращается ссылка на само это окно

Таблица 3.11 *(окончание)*

Свойство	Описание
personalbar	Возвращает ссылку на объект панели Personal Bar окна Navigator. Единственное свойство этого объекта visible позволит показать ее или убрать; значение true этого свойства показывает панель адреса, false убирает. Поддерживается только Navigator начиная с 4.0
screen	Возвращает ссылку на объект screen. (Об объекте screen см. ниже.)
screenLeft	Возвращает горизонтальную координату левого верхнего угла окна. Поддерживается только Internet Explorer начиная с 5.0
screenTop	Возвращает вертикальную координату левого верхнего угла окна. Поддерживается только Internet Explorer начиная с 5.0
screenX	Горизонтальная координата левого верхнего угла окна. Поддерживается только Navigator начиная с 4.0
screenY	Вертикальная координата левого верхнего угла окна. Поддерживается только Navigator начиная с 4.0
scrollbars	Возвращает ссылку на объект полос прокрутки окна Navigator. Единственное свойство этого объекта visible позволит показать полосу или убрать; значение true этого свойства показывает полосу прокрутки, false убирает. Поддерживается только Navigator начиная с 4.0
self	Возвращает ссылку на объект window текущего окна
status	Текст, отображаемый в строке состояния окна Web-обозревателя
statusbar	Возвращает ссылку на объект строки состояния окна Navigator. Единственное свойство этого объекта visible позволит показать ее или убрать; значение true этого свойства показывает строку состояния, false убирает. Поддерживается только Navigator начиная с 4.0
toolbar	Возвращает ссылку на объект панели инструментов окна Navigator. Единственное свойство этого объекта visible позволит показать ее или убрать; значение true данного свойства показывает панель инструментов, false убирает. Поддерживается только Navigator начиная с 4.0
top	Возвращает ссылку на родительское окно самого верхнего уровня, если текущий объект window представляет собой фрейм. В противном случае возвращается ссылка на само это окно
window	То же, что и self

Таблица 3.12. *Методы объекта* `window`

Метод	Описание
`alert({Текст})`	Выводит на экран окно предупреждения с текстом, переданным в качестве параметра
`back()`	Возвращается к предыдущему документу, как если бы на панели инструментов нажали кнопку **Назад**. Поддерживается только Navigator начиная с 4.0
`blur()`	Удаляет фокус с окна
`clearInterval({Таймер})`	Останавливает таймер, установленный методом `setInterval`
`clearTimeout({Таймер})`	Останавливает таймер, установленный методом `setTimeout`
`close()`	Закрывает текущее окно. Если окно было открыто методом `open()`, то оно закрывается сразу же, если же оно было открыто пользователем, сначала появляется окно предупреждения, предлагающее пользователю сделать выбор
`confirm({Текст})`	Выводит на экран окно предупреждения с текстом, переданным в качестве параметра, предлагающее пользователю сделать выбор. Если пользователь нажмет кнопку **OK**, возвращается `true`, если **Отмена** — `false`
`execScript({Выражение}, {Язык})`	Вычисляет переданное в качестве первого параметра выражение. Второй аргумент должен иметь значение `"JavaScript"`. Поддерживается только Internet Explorer начиная с 4.0
`find([{Строка поиска}[, true|false, true|false]])`	Выводит на экран диалоговое окно задания параметров поиска. Первым аргументом передается строка поиска. Второй задает, будет ли поиск производиться с учетом регистра символов (`true`) или нет (`false`, значение по умолчанию). Третий — будет ли поиск производиться с конца документа к началу (`true`) или обычным порядком (`false`, значение по умолчанию). Возвращает `true`, если поиск был успешным. Поддерживается только Navigator начиная с 4.0
`focus()`	Переносит фокус на текущее окно
`forward()`	Переходит к следующему документу в списке истории, как если бы на панели инструментов нажали кнопку **Вперед**. Поддерживается только Navigator начиная с 4.0

***Таблица 3.12** (продолжение)*

Метод	Описание
`home()`	Переходит на "домашнюю" страницу, заданную в настройках Web-обозревателя, как если бы на панели инструментов нажали кнопку **Домой**. Поддерживается только Navigator начиная с 4.0
`moveBy({X}, {Y})`	Перемещает окно на X пикселов вправо и на Y пикселов вниз. Для перемещения влево и вверх задайте отрицательные значения X и Y
`moveTo({X}, {Y})`	Перемещает окно в точку экрана, заданную координатами X и Y
`navigate({Адрес})`	Загружает в окно Web-страницу, адрес которой передан в качестве параметра. Поддерживается только Internet Explorer
`open({Адрес}, {Имя окна}, [{Список свойств окна, разделенных запятыми}])`	Открывает новое окно Web-обозревателя, загружает в него документ, адрес которого передан в первом параметре, и присваивает окну имя, переданное во втором параметре. В третьем параметре может быть передан список свойств окна; описание их приведено ниже по тексту, в табл. 3.13
`print()`	Печатает содержимое окна на принтере
`prompt({Приглашение}, [{Значение по умолчанию}])`	Выводит на экран диалоговое окно с полем ввода, приглашающее пользователя ввести какое-либо строковое значение. Текст приглашения передается в качестве первого параметра; во втором параметре может быть передано значение по умолчанию
`resizeBy({X}, {Y})`	Увеличивает окно на X пикселов по горизонтали и на Y пикселов по вертикали. Для уменьшения окна задайте отрицательные значения X и Y
`resizeTo({X}, {Y})`	Увеличивает или уменьшает окно до размера, заданного значениями X и Y
`scroll({X}, {Y})`	Прокручивает содержимое окна до точки с координатами X и Y. Не рекомендуется к использованию и сохранен только для совместимости
`scrollBy({X}, {Y})`	Прокручивает содержимое окна на X пикселов вправо и на Y пикселов вниз. Для прокрутки влево и вверх задайте отрицательные значения X и Y
`scrollTo({X}, {Y})`	Прокручивает содержимое окна в точку, заданную значениями X и Y

Таблица 3.12 (окончание)

Метод	Описание
`setHotKeys(true|false)`	Разрешает (`true`) или запрещает (`false`) "горячие" клавиши в окне, не имеющем строки меню. Поддерживается только Navigator начиная с 4.0
`setInterval({Функция или выражение}, {Интервал}, [{Список аргументов функции, разделенных запятыми}])`	Вычисляет значение выражения или вызывает функцию каждый раз по истечении заданного интервала (в миллисекундах). Может передавать в функцию заданные в списке аргументы. Возвращает указатель на объект таймера, который можно использовать в методе `clearInterval` для остановки и уничтожения таймера
`setResizeable(true|false)`	Разрешает (`true`) или запрещает (`false`) пользователю изменять размеры окна. Поддерживается только Navigator начиная с 4.0
`setTimeout({Функция или выражение}, {Интервал}, [{Список аргументов функции, разделенных запятыми}])`	Вычисляет значение выражения или вызывает функцию по истечении заданного интервала (в миллисекундах), если до этого не был вызван метод `clearTimeout`. Может передавать в функцию заданные в списке аргументы. Возвращает указатель на объект таймера, который можно использовать в методе `clearTimeout` для остановки и уничтожения таймера
`stop()`	Останавливает загрузку текущей страницы. Поддерживается только Navigator начиная с 4.0

***Таблица 3.13.** Свойства окна, передаваемые методу `open`*

Свойство окна	Описание
`alwaysLowered=yes|no`	Если `yes`, то создаваемое окно будет всегда находиться под другими окнами, даже если оно имеет фокус. Поддерживается только Navigator начиная с 4.06
`alwaysRaised=yes|no`	Если `yes`, то создаваемое окно будет всегда находиться над другими окнами, даже если оно не имеет фокуса. Поддерживается только Navigator начиная с 4.06.
`channelmode=yes|no`	Если `yes`, то создаваемое окно будет отображаться с панелью каналов (так называемый "режим театра"). Поддерживается только Internet Explorer начиная с 4.0
`dependent=yes|no`	Если `yes`, то создаваемое окно будет дочерним по отношению к создающему, т. е. при закрытии создающего окна будет закрываться и создаваемое. Поддерживается только Navigator начиная с 4.06

Таблица 3.13 *(продолжение)*

Свойство окна	Описание
`directories=yes|no`	Включает или отключает отображение кнопок директорий Navigator, у создаваемого окна
`fullscreen=yes|no`	Если `yes`, то создаваемое окно займет весь экран (так называемый "режим киоска"). Поддерживается только Internet Explorer начиная с 4.0
`height={Высота}`	Задает высоту создаваемого окна в пикселах
`hotkeys=yes|no`	Если `no`, то большинство "горячих" клавиш будут запрещены в создаваемом окне. Поддерживается только Navigator начиная с 4.06
`innerHeight={Высота}`	Задает высоту клиентской области создаваемого окна в пикселах. Поддерживается только Navigator начиная с 4.06; рекомендуется к использованию вместо `height`
`innerWidth={Ширина}`	Задает ширину клиентской области создаваемого окна в пикселах. Поддерживается только Navigator начиная с 4.06; рекомендуется к использованию вместо `width`.
`left={X}`	Задает горизонтальную координату левого верхнего угла создаваемого окна. Поддерживается только Internet Explorer начиная с 4.0
`location=yes|no`	Включает или отключает отображение панели адреса, включающего строку ввода адреса, у создаваемого окна
`menubar=yes|no`	Включает или отключает отображение строки меню у создаваемого окна
`outerHeight={Высота}`	Задает полную (с рамками, строкой меню, полосами инструментов) высоту создаваемого окна в пикселах. Поддерживается только Navigator начиная с 4.06; рекомендуется к использованию вместо `height`
`outerWidth={Ширина}`	Задает полную (с рамками, строкой меню, полосами инструментов) ширину создаваемого окна в пикселах. Поддерживается только Navigator начиная с 4.06; рекомендуется к использованию вместо `width`
`replace=yes|no`	Если `yes`, то адрес документа, размещаемого в создаваемом окне, заместит в списке истории адрес документа, находящегося в создающем окне. Поддерживается только Internet Explorer начиная с 4.0
`resizeable=yes|no`	Включает или отключает возможность изменения размера создаваемого окна
`screenX={X}`	То же, что и `left` для Internet Explorer. Поддерживается только Navigator начиная с 4.06

Таблица 3.13 (окончание)

Свойство окна	Описание
screenY={Y}	То же, что и top для Internet Explorer. Поддерживается только Navigator начиная с 4.06
scrollbars=yes|no	Включает или отключает отображение полос прокрутки у создаваемого окна
status=yes|no	Включает или отключает отображение строки состояния у создаваемого окна
titlebar=yes|no	Включает или отключает отображение заголовка у создаваемого окна. Поддерживается только Internet Explorer начиная с 4.0
toolbar=yes|no	Включает или отключает отображение панели инструментов у создаваемого окна
top={Y}	Задает вертикальную координату левого верхнего угла создаваемого окна. Поддерживается только Internet Explorer начиная с 4.0
width={Ширина}	Задает ширину создаваемого окна в пикселах
z-lock=yes|no	Если yes, то создаваемое окно никогда не будет перемещаться выше других окон, даже если получает фокус. Поддерживается только Navigator начиная с 4.06

* Вместо значений yes и no вы можете использовать 1 и 0. Некоторые версии Web-обозревателей для отдельных параметров принимают только значения 1 и 0.

Работа с окнами

Теперь остановимся подробнее на некоторых моментах использования свойств и методов окон.

Прежде всего, требуется дополнительное рассмотрение метода open. Он позволяет Web-дизайнеру открыть на экране дополнительное окно Web-обозревателя и поместить в него какую-либо страницу — очень полезное средство для разработки сложных многостраничных сайтов.

```
var contentsWindow;
contentsWindow = window.open("http://www.somesite.ru/contents.htm",
↳"contents");
```

В этом примере мы создали новое окно, загрузили в него некую Web-страницу и присвоили ей имя contents. Метод open вернул ссылку на объект вновь созданного окна, которую мы можем впоследствии использовать для работы с этим окном:

```
contentsWindow.Left = 300;
```

Имя окна можно использовать в атрибуте TARGET тега <A>:

```
<A HREF=" http://www.somesite.ru/contents2.htm"
↳TARGET="contents">Содержание</A>
```

Можно использовать дополнительные свойства окна, чтобы тонко управлять его характеристиками. Например, мы очень легко можем убрать у вновь создаваемого окна полосу инструментов и строку состояния:

```
contentsWindow = window.open("http://www.somesite.ru/contents.htm",
↳"contents", "toolbar=no", "status=no");
```

Мы можем задать начальные координаты и размеры создаваемого окна:

```
contentsWindow = window.open("http://www.somesite.ru/contents.htm",
↳"contents", "top=100", "left=200", "width=400", height="200");
```

А эта строка будет работать только в Navigator версии 4.06 или более поздней:

```
contentsWindow = window.open("http://www.somesite.ru/contents.htm",
↳"contents", "screenY=100", "screenX=200", "outerWidth=400",
↳outerHeight="200");
```

Когда созданное окно перестанет быть нужным, его можно закрыть при помощи метода close. Свойство closed позволит в дальнейшем проверить, закрыто ли это окно (например, если его закроет пользователь):

```
contentsWindow.close;
if (contentsWindow.closed) . . .
```

Свойство status позволяет поместить в строку состояния какой-либо текст, отличный от текста по умолчанию.

```
window.status = "Это мой крутой сайт";
```

А свойство defaultStatus позволит также просто вернуть туда текст по умолчанию.

```
window.status = window.defaultStatus;
```

Вы также можете вести примитивный диалог с пользователем, используя три специально предусмотренных метода. Метод alert выводит на экран окно предупреждения с заданным текстом и кнопкой **OK**:

```
window.alert("Привет, мир!");
```

Получившийся результат показан на рис. 3.3.

Метод confirm отображает окно сообщения с текстом и кнопками **OK** и **Отмена**. Если пользователь нажал **OK**, возвращается true, иначе — false.

```
window.confirm("Выберите что-нибудь.");
```

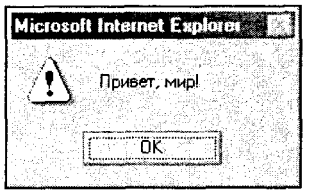


Рис. 3.3. Окно предупреждения, отображаемое методом `alert`

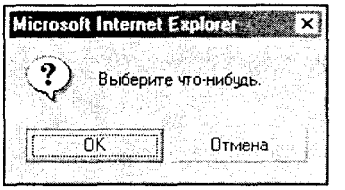


Рис. 3.4. Окно предупреждения, отображаемое методом `confirm`

Появившееся на экране окно предупреждения показано на рис. 3.4.

Метод `prompt` позволяет запросить у пользователя какие-либо данные. Он выводит диалоговое окно с полем ввода и кнопками **OK** и **Отмена**. В качестве параметров принимается текст приглашения и необязательный текст по умолчанию, отображаемый в поле ввода.

```
window.prompt("Введите ваше имя.", "Вася Пупкин");
```

На рис. 3.5 вы можете видеть, что появится на экране.

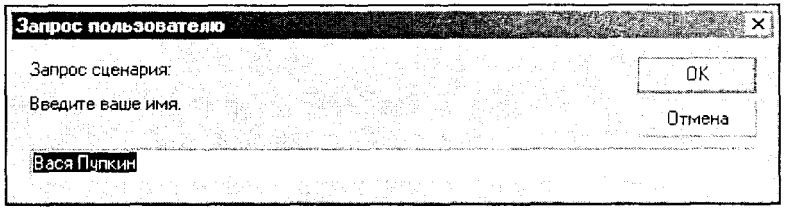


Рис. 3.5. Диалоговое окно ввода данных, отображаемое методом `prompt`

С остальными свойствами и методами все должно быть понятно.

А вот на четырех методах — `setInterval`, `setTimeout`, `clearInterval` и `clearTimeout` — мы остановимся подробнее. И заодно рассмотрим основные принципы анимации.

Анимация элементов страницы

Вы, наверное, часто видели, как на некоторых Web-страницах рисунки или заголовки ползают туда-сюда или спешат за курсором мыши. Как это делается? Понятно, что с помощью скриптов, встроенных в HTML-код. Но каких скриптов? И каким образом?

Прежде чем перейти к рассмотрению основных принципов создания "живых" рисунков, немного поговорим о собственно анимации.

Чтобы рисунок или фрагмент текста двигался по странице, нужно, прежде всего, изменять его координаты: вертикальную, горизонтальную или сразу обе. Как это делается, в принципе, понятно. Существуют свойства `pixelTop`

и pixelLeft объекта style, который имеется у каждого элемента страницы. Обратиться к ним проще простого. А изменить ту или иную координату чуть сложнее.

```
liveImage.style.pixelTop += 1;
```

Здесь мы увеличиваем вертикальную координату элемента liveImage на единицу, в результате чего он будет перемещаться вниз. Это простейший способ анимации — движение по вертикальной или горизонтальной прямой.

Имейте только в виду, что мы манипулируем с объектом style, поэтому элемент, который мы будем анимировать, должен иметь встроенный стиль, где определены все атрибуты местоположения и геометрических размеров: left, top, width и height. Иначе нам придется пользоваться объектом currentStyle, что доступно только на Internet Explorer версии 5.0 и новее. Разумеется, элемент, который мы хотим анимировать, должен быть позиционирован на странице свободно. То есть, атрибут стиля position этого элемента должен иметь значение absolute или relative, но никак не static:

```
<DIV ID="livediv" STYLE="top: 80; left: 0; width: 30;
position: absolute">
```

Чтобы заставить элемент двигаться по диагонали, нужно изменять обе его координаты одновременно по определенному закону:

```
liveImage.style.pixelTop += 1;
liveImage.style.pixelLeft -= 2;
```

Сейчас наш liveImage будет двигаться не только вниз, но и влево.

Так как со временем элемент страницы, который мы двигаем, может дойти до конца страницы, нам придется предусмотреть какие-то действия, чтобы прекратить движение или направить его по другому пути. Ниже приведен фрагмент скрипта, выполняющий такую проверку. Давайте рассмотрим его построчно.

```
var dx, dy;
dx = 1;
dy = -2;
```

Это начальные значения приращения координат.

```
liveImage.style.pixelTop = 0;
liveImage.style.pixelLeft = 0;
```

Изначально наш элемент находится в левом верхнем углу страницы.

```
do {
  liveImage.style.pixelTop += dy;
  liveImage.style.pixelLeft += dx;
```

Эти строки вам уже знакомы. Конечно же, они заставляют liveImage двигаться по странице.

```
if (liveImage.style.pixelTop + liveImage.style.pixelHeight >=
↳document.body.clientHeight) dy = -dy;
```

А здесь мы проверяем, не подошел ли наш "резвый" элемент к нижнему краю страницы. Для этого мы складываем вертикальную координату левого верхнего угла элемента с его высотой и сравниваем с некой предельной величиной, заданной нами. (Такой предельной величиной станет высота документа, которую мы получили, обратившись к объекту body, т. е. телу документа, и запросив свойство clientHeight.) Если элемент находится у края документа или пересек его, мы инвертируем приращение вертикальной координаты, и после этого элемент начнет двигаться в противоположном направлении.

```
if (liveImage.style.pixelTop <= 0) dy = -dy;
```

Определяем, не подошел ли элемент к верхнему краю документа. И выполняем те же действия. В принципе, эти две проверки можно было объединить в одну с помощью оператора логического ИЛИ ||, но я решил разнести их для наглядности.

```
if (liveImage.style.pixelLeft + liveImage.style.pixelWidth >=
↳document.body.clientWidth) dx = -dx;
if (liveImage.style.pixelLeft <= 0) dx = -dx;
```

Здесь то же самое, только вертикальная координата сменилась горизонтальной.

```
while (true);
```

В этом случае не обойтись без бесконечного цикла.

Если вы хотите заставить элемент страницы двигаться по более сложной траектории, например круговой, задача усложняется. В этом случае вам придется манипулировать не двумя линейными координатами, а одной линейной — радиусом — и одной угловой. Из них, пользуясь тригонометрическими функциями, вы получите две линейные координаты и присвоите их соответствующим свойствам объекта style.

Давайте рассмотрим пример движения какого-либо элемента по кругу.

```
var angle, rad, radius, xbegin, ybegin;
radius = 100;
angle = 0;
```

Пусть начальный угол равен нулю, а радиус — сотне пикселов. Разумеется, вы можете подставить другое значение.

```
xbegin = 200;
ybegin = 200;
```

Это координаты центра окружности, по которой будет двигаться наш элемент.

```
do {
angle += 1;
```

Увеличиваем значение угла.

```
rad = angle * Math.PI / 180;
```

Преобразуем градусы в радианы. Мы ведь помним, что тригонометрические функции объекта `Math` требуют в качестве параметров именно радианы.

```
liveImage.style.pixelLeft = xbegin + radius * Math.sin(rad);
liveImage.style.pixelTop = ybegin + radius * Math.cos(rad);
```

Вычисляем координаты нашего элемента как функции от угла и радиуса и не забываем прибавить координаты центра окружности.

```
if (angle >= 360) angle = 0;
```

Сбрасываем значение угла, если оно равно или больше 360, т. е. был сделан полный круг.

```
} while (true);
```

Вот такой скрипт.

Можно заставить наш элемент двигаться и по более заковыристой траектории. Это просто, надо только написать для нее функцию.

Итак, наш элемент свободно летает по странице. Что дальше?

Мы только что рассмотрели простейший тип анимации, когда элемент страницы меняет только свое положение, но не форму. А чтобы заставить его изменять и форму, нужно написать гораздо более сложный код и вдобавок много поработать в графическом редакторе. Вспомните мультфильмы, которые вы смотрели в детстве (и продолжаете смотреть сейчас). Персонажи их не просто ползают по экрану, но ЖИВУТ! А все потому, что изменяются не только их координаты, но и внешний вид. Давайте и мы так сделаем.

При классической анимации видимость движения персонажа достигается быстрой сменой нескольких изображений, представляющих собой разные его *фазы*, иначе говоря, состояния. Скажем, если необходимо получить изображение вращающегося земного шара (как в правом верхнем углу панели инструментов Internet Explorer), рисуются несколько земных шариков, повернутых к зрителю разными сторонами — это и есть фазы движения. После этого полученные изображения быстро сменяются одно другим так, чтобы шар казался вращающимся. Вот и все.

Анимированный таким способом элемент страницы можно также двигать, изменяя его геометрические размеры. В этом случае получится сложная анимация.

Чтобы сделать на своей страничке нечто подобное, нам понадобятся несколько изображений-кадров, запечатлевших разные фазы движения нашего элемента. Предположим, что они у нас есть, всего четыре. В таком случае вставим в тег тела документа такую строчку.

```
<BODY . . . onLoad="preloadImages();">
```

Здесь мы вызываем при загрузке страницы функцию `preloadImages`, код которой приведен ниже.

```
function preloadImages() {
  var arr;
  arr = new Array(4);
  arr[1] = new Image;
  arr[1].src = "liveimage1.gif";
  . . .
  arr[4] = new Image;
  arr[4].src = "liveimage4.gif"; }
```

В этой функции мы создаем массив из четырех элементов, по числу изображений-фаз. Далее, каждому этому элементу мы присваиваем вновь созданный объект `Image` (объект изображения, создаваемый тегом `<IMG>`, правда, не видимый на странице) и помещаем в свойство `src` такого объекта адрес соответствующего изображения. (Свойство `src` объекта `Image` хранит адрес файла изображения.) Это заставит Web-обозреватель подгрузить файл изображения и сохранить его в памяти. Сам массив, где хранятся ссылки на объекты изображений, нам впоследствии больше не понадобится.

А теперь сам скрипт анимации.

```
var angle, rad, radius, xbegin, ybegin, counter;
radius = 100;
angle = 0;
xbegin = 200;
ybegin = 200;
counter = 0;
```

Это дополнительный счетчик, в зависимости от значения которого мы будем помещать в анимируемый элемент то или иное изображение-фазу.

```
do {
angle += 1;
counter += 1;
```

Инкрементируем значение дополнительного счетчика.

```
liveImage.src = "liveimage" + counter + ".gif";
```

Подставляем в свойство `src` объекта `liveImage` имя соответствующего файла изображения-фазы. Так как все эти файлы уже сохранены в промежуточной памяти Web-обозревателя, их загрузка не займет много времени.

И еще. Скорость смены кадров можно менять, изменяя величину инкремента дополнительного счетчика. Так, чтобы замедлить анимацию в два раза, можно увеличивать его на 0.5 вместо единицы.

```
rad = angle * Math.PI / 180;
liveImage.style.pixelLeft = xbegin + radius * Math.sin(rad);
liveImage.style.pixelTop = ybegin + radius * Math.cos(rad);
if (angle >= 360) angle = 0;
if (counter >= 4) counter = 0;
```

Проверка дополнительного счетчика.

```
} while (true);
```

Все! Теперь у нас на странице будет что-то вроде мультфильма.

Вместо непосредственной смены отдельных кадров анимированного персонажа из кода скрипта можно использовать анимированный GIF-файл. Так даже будет проще: не надо будет заготавливать уйму изображений-фаз и писать сложный код для их смены.

Итак, мы поняли, что при анимации координаты движения элемента должна существовать функция вычисления траектории, принимающая в качестве параметров какие-либо геометрические данные. Также следует предусмотреть случаи выхода координат или параметров за какие-либо границы: естественные, например границы окна Web-обозревателя или родительского элемента, и искусственные, установленные программистом. Как мы убедились, это не очень сложно, если не переусердствовать с выбором сложной траектории.

Казалось бы, все... Но это не так.

Когда мы изменяем координаты какого-либо элемента страницы, Web-обозреватель старается переместить его так быстро, насколько хватает мощности компьютера. И, естественно, на более быстром компьютере наш элемент будет перемещаться значительно быстрее, чем на медленном. На самых современных многомегагерцевых монстрах этот несчастный элемент будет летать со скоростью пули, вы даже вряд ли сможете уловить его взглядом. Получается, что мы не можем управлять скоростью движения нашего элемента?

Нет! На самом деле, можем. Для этого в функцию траектории вводится еще одна координата — время. А для управления временем служат методы `setInterval`, `setTimeout`, `clearInterval` и `clearTimeout` объекта `window`. В данный момент нам будут более полезны методы `setInterval` и `clearInterval`. Они позволяют задать интервал и функцию (или выражение), которая будет выполняться каждый раз по прошествии этого интервала.

Чтобы жестко привязать нашу анимацию ко времени, оформим код, ответственный за анимацию, в виде функции, удалим оттуда бесконечный цикл,

создадим объект таймера методом setInterval и выберем подходящий интервал.

Ниже приведен полный код скрипта, выполняющего все вышесказанное.

```
var angle, radius, xbegin, ybegin, counter, timer;
radius = 100;
angle = 0;
xbegin = 200;
ybegin = 200;
counter = 0;
function setupAnimation() {
  var arr;
  arr = new Array(4);
  arr[1] = new Image;
  arr[1].src = "liveimage1.gif";
  . . .
  arr[4] = new Image;
  arr[4].src = "liveimage4.gif";
  timer = window.setInterval("moveImage()", 100) }
```

Функцию preloadImages мы переименовали в setupAnimation. А внутри нее, кроме всего прочего, мы устанавливаем таймер, вызывающий функцию moveImage каждые 100 миллисекунд.

```
function moveImage() {
  var rad;
  angle += 1;
  counter += 1;
  liveImage.src = "liveimage" + counter + ".gif";
  rad = angle * Math.PI / 180;
  liveImage.style.pixelLeft = xbegin + radius * Math.sin(rad);
  liveImage.style.pixelTop = ybegin + radius * Math.cos(rad);
  if (angle >= 360) angle = 0;
  if (counter >= 4) counter = 0; }
```

Это конец нашего скрипта. Его следует поместить в заголовок HTML-документа.

```
<BODY . . . onLoad="setupAnimation();">
```

И не забудьте добавить в тег тела документа эту строчку.

Если вам когда-нибудь понадобится прервать анимацию, просто вызовите метод clearInterval. Он уничтожит определенный нами объект таймера.

```
window.clearInterval(timer);
```

Пара слов о двух других методах управления таймерами. Метод setTimeout просто однократно вызывает функцию или вычисляет выражение по исте-

чении установленного интервала. А метод `clearTimeout` уничтожает этот таймер, пока он еще не сработал.

Пример страницы с анимацией

Теперь самое время создать что-то такое, что можно увидеть воочию.

Давайте сделаем не просто страницу, а страницу с небольшой выдумкой. В частности, поэкспериментируем со свойством `zIndex` разных элементов и посмотрим, что из этого получится. (Напоминаю, что свойство `zIndex` отвечает за порядок, в котором элементы Web-страницы перекрывают друг друга.) Ради этого мы даже не будем городить огород со сложной анимацией, а ограничимся простой, где элемент страницы — рисунок — движется по прямой линии.

Код страницы

Здесь мы построчно опишем код страницы.

```
<HTML>
<HEAD>
<TITLE>Анимация на Web-странице</TITLE>
<STYLE>
DIV { font-size: 72; font-weight: bold }
```

Сделаем неподвижные элементы страницы покрупнее.

```
#div1 { top: 50; left: 50; position: absolute; z-index: 1 }
#div2 { top: 50; left: 100; position: absolute; z-index: -1 }
#div3 { top: 50; left: 150; position: absolute; z-index: 1 }
#div4 { top: 50; left: 200; position: absolute; z-index: -1 }
```

Это стили неподвижных элементов. В принципе, их можно было встроить прямо в теги, но я вынес их в таблицу для наглядности. Обратите внимание на значения атрибута `z-index`. Стиль "живого" элемента мы встроим прямо в тег, чтобы впоследствии им можно было манипулировать через объект `style`.

```
</STYLE>
<!-- Здесь у нас будут скрипты -->
</HEAD>
<BODY onLoad="setupAnimation();">
```

Не забудем при загрузке страницы вызвать функцию, выполняющую подготовительные действия.

```
<DIV ID="div1">J</DIV>
<DIV ID="div2">a</DIV>
```

```
<DIV ID="div3">v</DIV>
<DIV ID="div4">a</DIV>
```

Это неподвижные элементы. А вот подвижный рисунок:

```
<DIV ID="livediv" STYLE=" top: 70; left: 0; width=30;
↳position: absolute"><IMG SRC="tips.gif"></DIV>
```

Обратите внимание на определение стиля: мы должны определить все свойства, которыми будем манипулировать через объект `style`. Иначе результат будет непредсказуемым.

```
</BODY>
</HTML>
```

Собственно, вот и все. На этот раз у нас получилась довольно простая страничка.

Скрипты

Код скриптов не преподнесет нам сюрпризов.

```
<SCRIPT>
var dx, timer;
dx = 2;

function moveImage() {
  livediv.style.pixelLeft += dx;
  if (livediv.style.pixelLeft + livediv.style.pixelWidth >=
  ↳document.body.clientWidth) dx = -dx;
  if (livediv.style.pixelLeft <= 0) dx = -dx; }

function setupAnimation() {
  timer = window.setInterval("moveImage()", 100) }
</SCRIPT>
```

Рис. 3.6. Web-страница с движущимся рисунком

Скрипты вышли не очень большими и совсем не сложными. Мы почти все теоретические вопросы рассмотрели в предыдущей главе, поэтому я даже не буду комментировать скрипты.

Объедините HTML-код страницы и код скриптов, сохраните получившийся файл под именем 3.5.htm и откройте его в Web-обозревателе. К сожалению, рис. 3.6 не может передать движение.

Если вы хотите чего-либо посложнее, можете использовать программы более сложной анимации, рассмотренные выше. Все они работают. Вам остается только вставить их в код страницы. Думаю, вы справитесь с этим без моей помощи.

Объект *layer*

Объект `layer` представляет собой слой — свободно позиционированный элемент Web-страницы, созданный с помощью тегов `<LAYER>` или `<ILAYER>`. Этот объект поддерживается только Navigator, как, собственно, и сами слои.

Как вы уже знаете, практически всеми элементами Web-страницы можно управлять из скрипта, используя объектную модель документа. К несчастью, это в полной мере относится только к Internet Explorer. Объектная модель Navigator значительно слабее; реальной мощью обладают только гиперссылки, формы (о формах см. ниже) и слои. Именно слои и их использование в "живых" Web-страницах мы сейчас и рассмотрим.

Доступ к слоям

Все слои, определенные в документе, доступны как элементы коллекции `layers`.

```
document.layers[2]
document.layers["somelayer"]
```

Navigator поддерживает прямой доступ к слоям по именам. Имена слоям даются с помощью атрибута `NAME` тегов `<LAYER>` и `<ILAYER>`:

```
document.somelayer
```

Для доступа к слою, находящемуся внутри другого слоя, используется такой способ:

```
document.layers["outerlayer"].layers["innerlayer"]
```

Свойства и методы

Некоторые из поддерживаемых объектом `layer` свойств перечислены в табл. 3.14, методов — в табл. 3.15. Полный список свойств и методов объекта слоя приведен в приложении 3.

Таблица 3.14. *Свойства объекта* `layer`

Свойство	Описание
`above`	Возвращает ссылку на слой, находящийся над текущим (т. е. выше в z-последовательности). Если таких нет, возвращается `null`
`background`	Ссылка на объект `Image`, представляющий фоновый рисунок текущего слоя. Вы можете использовать свойство `src` этого объекта для задания или получения адреса файла рисунка. Этому свойству можно присвоить ссылку на другой объект `Image`. Если для текущего слоя не задан фоновый рисунок, возвращается `null`
`below`	Возвращает ссылку на слой, находящийся под текущим (т. е. ниже в z-последовательности). Если таких нет, возвращается `null`
`bgcolor`	Цвет фона слоя
`document`	Возвращает ссылку на объект `document`, представляющий содержимое слоя
`left`	Горизонтальная координата левого верхнего угла слоя в пикселах
`name`	Имя тега, заданное в атрибуте `NAME`
`pageX`	Горизонтальная координата слоя относительно остального документа в пикселах
`pageY`	Вертикальная координата слоя относительно остального документа в пикселах
`parentLayer`	Возвращает ссылку на родительский слой или объект `window`, если родительского слоя нет
`siblingAbove`	Возвращает ссылку на слой, находящийся над текущим (т. е. выше в z-последовательности) и имеющий того же родителя. Если таких нет, возвращается `null`
`siblingBelow`	Возвращает ссылку на слой, находящийся под текущим (т. е. ниже в z-последовательности) и имеющий того же родителя. Если таких нет, возвращается `null`
`src`	Возвращает адрес файла документа, отображаемого в слое
`top`	Вертикальная координата левого верхнего угла слоя в пикселах
`visibility`	Видимость слоя: `show` показывает его, `hide` скрывает, а `inherit` наследует видимость от родителя (это поведение по умолчанию)
`window`	Возвращает ссылку на объект `window` (или `frame`), где находится текущий слой
`x`	То же самое, что `left`
`y`	То же самое, что `top`

Таблица 3.14 *(окончание)*

Свойство	Описание
`zIndex`	Порядок перекрытия слоев. Слои с большим значением этого свойства перекрывают слои с меньшим значением. По умолчанию равно нулю

Таблица 3.15. *Методы объекта* `layer`

Метод	Описание
`load({Адрес файла нового документа}, {Новая ширина слоя в пикселах})`	Позволяет заменить содержимое слоя. В качестве первого параметра принимает интернет-адрес документа, который будет отображен в текущем слое. В качестве второго параметра указывается новая ширина текущего слоя в пикселах
`moveAbove({Имя слоя})`	Перемещает текущий слой выше слоя, имя которого передано в качестве параметра
`moveBelow({Имя слоя})`	Перемещает текущий слой ниже слоя, имя которого передано в качестве параметра
`moveBy({X}, {Y})`	Перемещает слой на X пикселов вправо и на Y пикселов вниз. Для перемещения влево и вверх задайте отрицательные значения X и Y
`moveTo({X}, {Y})`	Перемещает слой в точку, заданную координатами X и Y. Координаты отсчитываются относительно родительского слоя (если такой есть)
`moveToAbsolute({X}, {Y})`	Перемещает слой в точку, заданную координатами X и Y. Координаты отсчитываются относительно документа
`resizeBy({X}, {Y})`	Увеличивает слой на X пикселов по горизонтали и на Y пикселов по вертикали. Для его уменьшения задайте отрицательные значения X и Y
`resizeTo({X}, {Y})`	Увеличивает слой до размера, заданного значениями X и Y

Все описанные в таблицах свойства и методы, надеюсь, понятны без дополнительного разъяснения. Поэтому рассмотрим сразу же пример, где будут использоваться слои.

Пример страницы с анимацией для Navigator

Давайте перепишем нашу страничку с анимированным рисунком так, чтобы она была совместима с Navigator. (К несчастью, она при этом потеряет совместимость с Internet Explorer.)

Код страницы

```
<HTML>
<HEAD>
<TITLE>Анимация на Web-странице</TITLE>
<!-- Здесь у нас будут скрипты -->
</HEAD>
<BODY onLoad="setupAnim();" STYLE="font-size: 72pt; font-weight: bold">
```

Сделаем текст нашей страницы как можно более крупным. Эти настройки унаследуют все слои.

```
<LAYER NAME="layer1" LEFT="50" TOP="20" Z-INDEX="2">J</LAYER>
<LAYER NAME="layer2" LEFT="100" TOP="20" Z-INDEX="0">a</LAYER>
<LAYER NAME="layer3" LEFT="150" TOP="20" Z-INDEX="2">v</LAYER>
<LAYER NAME="layer4" LEFT="200" TOP="20" Z-INDEX="0">a</LAYER>
```

Это неподвижные слои. А вот подвижный слой с рисунком:

```
<LAYER NAME="livelayer" LEFT="0" TOP="60" Z-INDEX="1">
<IMG SRC="tips.gif">
</LAYER>
```

Как видите, по сравнению с предыдущим примером мы немного изменили геометрические параметры слоев с тем, чтобы они лучше ложились на страницу. К тому же, атрибут `Z-INDEX` слоя принимает иные значения, чем атрибут стиля `z-index` рассмотренного выше примера для Internet Explorer.

```
</BODY>
</HTML>
```

Вот это да! Код странички для Navigator получился меньше. Может, его разработчики этого и добивались?..

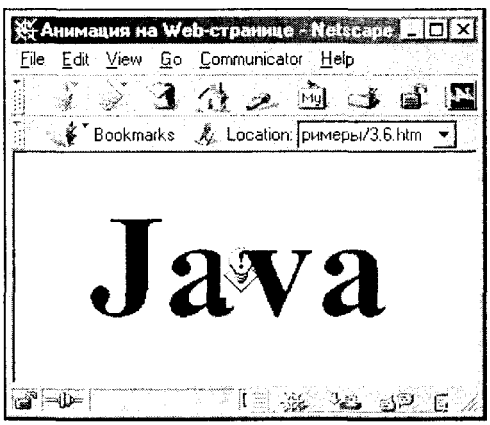
Скрипты

```
<SCRIPT>
var dx, timer;
dx = 2;

function moveImage() {
  document.livelayer.left += dx;
```

```
  if (document.livelayer.left >= document.width) dx = -dx;
  if (document.livelayer.left <= 0) dx = -dx; }

function setupAnim() {
  timer = window.setInterval("moveImage()", 100) }
</SCRIPT>
```

А на скриптах остановимся подробнее.

В них можно выделить три отличия. Во-первых, мы не можем ни определить, ни установить ширину слоя, если не задали ее непосредственно в теге <LAYER> (или <ILAYER>) атрибутом WIDTH. Web-обозреватель установит ширину слоя такой, чтобы вместить в него все содержимое. Мы можем только изменить ее методом load при загрузке в слой нового документа, но наша страница — не тот случай. Во-вторых, мы всегда указываем ссылку на объект document. Internet Explorer позволял опускать ее, а Navigator не позволяет. И в-третьих, чтобы определить ширину документа, мы обращаемся к свойству width объекта document, не поддерживаемому Internet Explorer.



Объедините код страницы и код скриптов, сохраните получившийся файл под именем 3.6.htm и откройте его в Navigator. Должно получиться нечто, похожее на рис. 3.7, но движущееся.

Рис. 3.7. Web-страница с движущимся рисунком, совместимая с Navigator

И возрадовались поклонники Navigator: "Ура! Пришел и на нашу улицу праздник!" А если серьезно, Navigator тоже на многое способен, более того, некоторые вещи на нем делаются проще, чем на Internet Explorer. Это закон природы: достоинства одних оборачиваются недостатками других, и наоборот.

Объект *navigator*

Объект navigator служит для доступа к самой программе Web-обозревателя. Не путайте его с объектом window, представляющим текущее окно Web-обозревателя, и названием программы Netscape Navigator.

Свойства и методы

Поддерживаемые объектом navigator свойства перечислены в табл. 3.16.

Таблица 3.16. *Свойства объекта* navigator

Свойство	Описание
appCodeName	Возвращает имя кода программы Web-обозревателя. И для Internet Explorer, и для Navigator вернет строку "Mozilla"
appMinorVersion	Возвращает младшую цифру номера версии программы Web-обозревателя. Например, для Internet Explorer 5.0 вернет "0", для 5.5 — "5". Поддерживается только Internet Explorer начиная с 4.0
appName	Возвращает имя программы Web-обозревателя, например, "Netscape" или "Microsoft Internet Explorer"
appVersion	Возвращает версию программы Web-обозревателя. Полное описание возвращаемого значения приведено после таблицы
browserLanguage	Возвращает код языка программы Web-обозревателя. Поддерживается только Internet Explorer начиная с 4.0
cookieEnabled	Возвращает true, если Web-обозревателю разрешен пользователем прием cookie. Поддерживается только Internet Explorer начиная с 4.0
cpuClass	Возвращает класс процессора клиентского компьютера, например, "x86" или "Alpha". Поддерживается только Internet Explorer начиная с 4.0
language	Возвращает код языка программы Web-обозревателя. Поддерживается только Navigator начиная с 4.0
onLine	Возвращает true, если клиент в настоящий момент подключен к Интернету (находится в режиме on-line), и false, если отключен (off-line). Поддерживается только Internet Explorer начиная с 4.0
platform	Возвращает название клиентской платформы, например, "Win32"
systemLanguage	Возвращает код языка операционной системы клиента. Поддерживается только Internet Explorer начиная с 4.0
userAgent	Возвращает строку, идентифицирующую Web-обозреватель клиента. Является комбинацией значений свойств appCodeName и appVersion. Подробнее описано после таблицы
userLanguage	То же самое, что browserLanguage. Поддерживается только Internet Explorer начиная с 4.0

Объект navigator поддерживает, кроме того, метод javaEnabled(), возвращающий true, если Web-обозревателю разрешено пользователем выполнение сценариев JavaScript.

А теперь поговорим подробнее о свойстве appVersion, точнее, о возвращаемом им значении. Дело в том, что у разных программ оно будет разным.

У Navigator оно имеет следующий формат:

```
{Версия} [{Язык}] ({Операционная система}; U|I)
```

Здесь {Версия} представляет собственно версию Web-обозревателя, {Язык} — язык программы (может отсутствовать), {Операционная система} — обозначение операционной системы клиента, например, "Win95", "Win16" или "WinNT", буква "U" — американскую версию программы, а "I" — интернациональную. Например,

```
4.0 [ru] (Win95; I)
```

А у Internet Explorer формат свойства appVersion заметно отличается.

```
{Совместимая версия Navigator} (compatible; {Версия};
↳{Операционная система})
```

Здесь {Операционная система} может принимать значения "Windows 3.1", "Windows 3.11", "Windows 95" или "Windows NT".

```
2.0 (compatible; 3.01; Win95)
```

Свойство userAgent возвращает значение, имеющее формат:

```
{Значение appCodeName}/{Значение appVersion}
```

То есть, для двух предыдущих примеров мы получим следующие значения:

```
Mozilla/4.0 (Win95; I)
Mozilla/2.0 (compatible; 3.01; Win95)
```

Простейший скрипт для определения Web-обозревателя

Как вы уже заметили (не могли не заметить), Internet Explorer и Navigator отличаются настолько, что для них приходится готовить разные Web-страницы. Internet Explorer поддерживает такие возможности, которые не поддерживает Navigator, и наоборот. Возникает проблема: как сделать страницы, совместимые с разными программами Web-обозревателей?

Для этого можно использовать свойства объекта navigator. А именно, свойства appName и appVersion.

Ниже приведен код скрипта, осуществляющего это. Он снабжен обширными построчными комментариями, так что сможете легко изменить его под свои нужды.

```
if (navigator.appName.indexOf("Netscape") != -1)
  window.alert("Netscape Navigator!")
```

То есть, если в значении, возвращенном свойством `appName`, есть строка "Netscape", значит это Navigator.

Конечно, вместо вывода окна предупреждения вы можете использовать любой код. Например, вызвать `location.href` для перехода на страницу, созданную специально для Navigator.

```
else
  window.alert("Microsoft Internet Explorer!");
```

Это простейший скрипт. В более сложном случае вы можете проверить свойство `appVersion` на предмет версии программы Web-обозревателя. Или даже использовать регулярные выражения JavaScript для "разбора" возвращенной строки. Ниже мы приведем пример такого скрипта. А пока что вот вам код небольшой Web-страницы, определяющей тип программы Web-обозревателя и перенаправляющей клиента на другие страницы в соответствии с этим типом.

```
<HTML>
<HEAD>
<TITLE>Определение Web-обозревателя</TITLE>
<SCRIPT>
function determineIt() {
  if (navigator.appName.indexOf("Netscape") != -1)
    location.href="index_nn.htm"
  else
    location.href="index_ie.htm"; }
</SCRIPT>
</HEAD>
<BODY onLoad="determineIt();">
Ваш Web-обозреватель не поддерживает сценарии JavaScript.
</BODY>
</HTML>
```

Этот код настолько прост, что не нуждается в дополнительных комментариях. Вы можете использовать его везде, где нужно реагировать на разные типы Web-обозревателей и перенаправлять их на разные страницы. Если же Web-обозреватель не поддерживает JavaScript, на экране появится предупреждающая об этом надпись.

Объект *screen*

Объект `screen` служит для доступа к характеристикам видеосистемы компьютера клиента.

Свойства и методы

Поддерживаемые объектом `screen` свойства перечислены в табл. 3.17. Методов этот объект не поддерживает.

Таблица 3.17. *Свойства объекта* `screen`

Свойство	Описание
`availHeight`	Возвращает высоту полезной области экрана без панели задач и подобных ей элементов графического интерфейса системы
`availWidth`	Возвращает ширину полезной области экрана без панели задач и подобных ей элементов графического интерфейса системы
`colorDepth`	Возвращает глубину цвета. Для 16 цветов возвращается 2, для 256 — 8, для 16,7 миллионов цветов (режим High Color) — 32
`height`	Возвращает полную высоту экрана
`width`	Возвращает полную ширину экрана

Пример страницы, определяющей характеристики Web-обозревателя

А теперь самое время рассмотреть какой-нибудь сложный и полезный пример. Пусть это будет Web-страница, перечисляющая характеристики программы Web-обозревателя и операционной системы клиента.

На этот раз мы не будем делить код страницы на HTML и JavaScript, поскольку HTML-код будет совсем простым. Зато часть, написанная на JavaScript, будет воистину огромной.

```
<HTML>
<HEAD>
<TITLE>Характеристики Web-обозревателя</TITLE>
</HEAD>
<BODY>
<SCRIPT>
```

На этот раз мы поместим наши скрипты прямо в тело HTML-документа.

```
var sv, aname, re, result;
document.write("Имя кода программы: " + navigator.appCodeName);
document.write("<BR>");
aname = navigator.appName;
document.write("Имя программы: " + aname);
document.write("<BR>");
```

Здесь все понятно. Мы записываем в тело документа строки, представляющие имена свойств объекта `navigator` и их значения. Каждая строка отделяется друг от друга тегом разрыва `<BR>`. Пока что идут свойства, одинаково поддерживаемые и Internet Explorer, и Navigator...

```
sv = navigator.appVersion;
```

Это свойство возвращает в качестве значения сложную строку, формат которой вдобавок зависит от конкретной версии программы. Поэтому для "разбора" строки применим регулярные выражения:

```
if (aname.indexOf("Netscape") != -1) {
  re = /([0-9\.]+)\s*.*\s+\((.+);\s+(I|U)\)/i;
```

А теперь создаем регулярное выражение:

```
result = sv.match(re);
```

и выполняем "разбор" строки, возвращенной методом `appVersion`. Наше регулярное выражение содержит три подвыражения, которые и будут содержать нужную нам информацию о версии и платформе клиента, на которой работает Web-обозреватель.

```
document.write("Версия программы: " + RegExp.$1);
document.write("<BR>");
document.write("Платформа клиента: " + RegExp.$2);
document.write("<BR>");
if (RegExp.$3 == "I")
  document.write("Интернациональная версия")
else
  document.write("Американская версия");
```

Мы используем глобальный объект `RegExp` и его свойства, чтобы получить отдельные части "разобранной" регулярным выражением строки.

```
document.write("<BR>");
document.write("Язык: " + navigator.language); }
```

А это свойство специфично для Navigator:

```
else {
  re = /([0-9\.]+)\s+\(compatible;\s+.+\s+([0-9\.]+);\s+(.+)\)/i;
  result = sv.match(re);
```

```
  document.write("Версия программы: " + RegExp.$2);
  document.write("<BR>");
  document.write("Совместима с Navigator " + RegExp.$1);
  document.write("<BR>");
  document.write("Платформа клиента: " + RegExp.$3);
  document.write("<BR>");
  document.write("Процессор: " + navigator.cpuClass);
  document.write("<BR>");
  document.write("Язык: " + navigator.browserLanguage); }
```

Здесь, в принципе, то же самое, только для Internet Explorer.

```
document.write("<BR>");
document.write("Разрешение экрана: " + screen.availWidth + "*" +
↳screen.availHeight);
```

И в заключение обращаемся к объекту `screen`, чтобы получить данные о разрешении экрана клиентского компьютера.

```
</SCRIPT>
</BODY>
</HTML>
```

Все. Мы это сделали!

Сохраните полученный код в файле под именем 3.7.htm и откройте его поочередно в Internet Explorer и Navigator. Результаты вы увидите соответственно на рис. 3.8 и 3.9.

Рис. 3.8. Страница со сведениями о Web-обозревателе, открытая в Internet Explorer

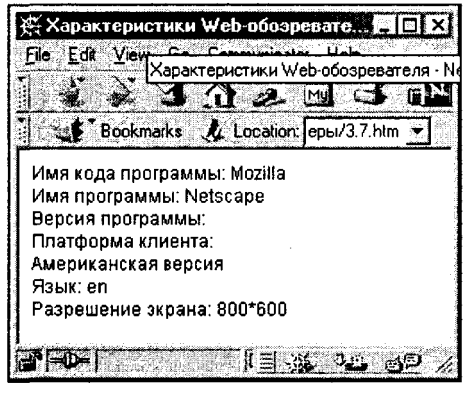


Рис. 3.9. Страница со сведениями о Web-обозревателе, открытая в Navigator

На самом деле, здесь все довольно просто. Единственную сложность могут представлять регулярные выражения: если в них встретится ошибка, ее

очень трудно найти. Поэтому я рекомендую вам строить регулярные выражения по частям: сначала одно подвыражение, потом, после проверки правильности работы первого, второе и т. д. И не забывайте при этом включить в код скрипта команды вывода промежуточных результатов — в случае ошибки это поможет вам понять, что же в вашем скрипте работает неправильно.

Объект *history*

Объект `history` представляет интерфейс к списку истории Web-обозревателя, т. е. списку всех Web-страниц, просмотренных пользователем в течение времени, указанного в настройках.

Все поддерживаемые этим объектом свойства и методы перечислены в табл. 3.18 и 3.19 соответственно.

Таблица 3.18. *Свойства объекта* `history`

Свойство	Описание
`current`	Возвращает интернет-адрес документа, загруженного в настоящее время. Поддерживается только Navigator начиная с 3.0
`length`	Возвращает размер списка истории
`next`	Возвращает интернет-адрес следующего в списке истории документа. Поддерживается только Navigator начиная с 3.0
`previous`	Возвращает интернет-адрес предыдущего в списке истории документа. Поддерживается только Navigator начиная с 3.0

Таблица 3.19. *Методы объекта* `history`

Метод	Описание
`back()`	Загружает в окно Web-обозревателя предыдущий документ из списка истории
`forward()`	Загружает в окно Web-обозревателя следующий документ из списка истории
`go({Адрес})`	Загружает в окно Web-обозревателя документ из списка истории, интернет-адрес которого наиболее близок к переданному в качестве параметра. Поддерживается только Navigator
`go({Позиция})`	Перемещается в списке истории на позицию, номер которой передан в качестве параметра. Поддерживается только Internet Explorer начиная с 4.0

Таблица 3.19 (окончание)

Метод	Описание
`go([Приращение])`	Перемещается в списке истории на количество позиций, переданное в качестве параметра. Можно задавать как положительные, так и отрицательные значения приращения. Поддерживается только Navigator

Работа с фреймами

А теперь рассказ пойдет о работе с фреймами. Поддержка фреймов в объектной модели документа осуществляется объектами `frame` и `frameset` и коллекцией `frames`.

Коллекция *frames*. Работа с фреймами

Коллекция `frames` является внутренней коллекцией объектов `document` и `window`. Так что вы можете обращаться к тому из них, к которому в данный момент обратиться проще:

```
document.frames(2)
window.frames("contentframe")
```

Каждое из этих выражений вернет ссылку на соответствующий фрейму объект `window`. Как вы помните, фрейм подобен небольшому окошку внутри большого окна документа; и это окошко ведет себя во многом независимо от других таких же окошек-фреймов. Поэтому вполне логично описать фрейм как объект `window`.

Вы можете запрашивать геометрические размеры фрейма, используя свойства объекта `window`:

```
frameWidth = frames(1).document.body.clientHeight;
frameWidth = document.frames(1).innerHeight;
```

Первое из этих выражений будет работать в Internet Explorer, второе — в Navigator. И оба вернут величину высоты клиентской области фрейма, т. е. без учета границ и отступов.

Вы можете загружать во фрейм другие документы:

```
frames("contentsframe").location.href = "chapter_2.html";
```

Это выражение будет работать в обоих Web-обозревателях.

Ниже приведен небольшой фрагмент HTML-кода двух гиперссылок, загружающих два разных документа в один и тот же фрейм. Только одна из них использует средства "чистого" HTML, а другая скрипт.

```
<A HREF="chapter_1.html" TARGET="contentsframe">Часть 1</A>
<A HREF="javascript:frames('contentsframe').location.href =
↳'chapter_2.html'">Часть 1</A>
```

И под конец, чтобы завершить эту главу, приведем код небольшой функции, определяющей, помещен ли во фрейм какой-либо документ, и, если он отсутствует, загружающей его. Вы можете использовать эту функцию в своих Web-страницах.

```
<SCRIPT>
function loadDocument (documentURL, frameName) {
  if (frames(frameName).location.href == documentURL)
    frames(frameName).location.href = documentURL; }
</SCRIPT>
. . .
<A HREF="javascript:loadDocument('chapter_1.html', 'contentsframe')">
↳Часть 1</A>
```

Работа с объектами фреймов и набором фреймов

А здесь мы опишем более "продвинутую" работу с фреймами. Как вы знаете, каждый элемент страницы можно представить объектом. Точно такими же объектами с большим набором свойств и методов могут быть представлены фреймы и наборы фреймов.

Сразу скажу, что большинство из того, о чем пойдет речь дальше, относится к Internet Explorer.

Итак, мы уже умеем загружать во фрейм другой документ:

```
frames("contentsframe").location.href = "chapter_2.html";
```

Здесь мы обратились к фрейму как к окну. А можно сделать и так:

```
contentsframe.src = "chapter_2.html";
```

То есть, мы обратились к фрейму как к элементу страницы и изменили его свойство `src`, связанное с атрибутом `SRC` тега `<FRAME>`.

Мы также можем изменить значения отступа между границей и содержимым фрейма:

```
contentsframe.marginHeight = 5;
contentsframe.marginWidth = 10;
```

или запретить ему прокручиваться:

```
contentsframe.scrolling = "no";
```

В общем, мы можем делать с фреймом все, что угодно. И для этого даже не обязательно иметь перед глазами справочник всех его свойств — достаточно просто помнить все атрибуты тега <FRAME>. (Конечно, если вы захотите применить к фрейму какие-либо методы, придется обратиться к справочнику, например, приложению 3 этой книги.)

Если отдельным фреймом можно "крутить" как заблагорассудится, то как насчет набора фреймов <FRAMESET>? Можем ли мы обращаться с ним так же легко? Конечно!

Например, мы можем изменить толщину рамки фреймов, входящих в набор:

```
commonframeset.border = "2";
```

или изменить ее цвет:

```
commonframeset.borderColor = "blue";
```

Более того. Мы можем изменить относительные размеры фреймов, входящих в этот набор:

```
commonframeset.cols = "100, *";
```

Мы только что задали новые размеры фреймов-колонок.

```
commonframeset.rows = "20%, 80%";
```

А теперь — фреймов-строк.

Как видите, объектами фреймов и наборов фреймов можно манипулировать как угодно. Как и любым элементом Web-страницы, загруженной в Internet Explorer.

Динамические свойства Internet Explorer

Динамические свойства — это новая возможность, появившаяся только в версии 5.0 Internet Explorer. Она позволит придать вашим Web-страницам еще больше динамизма и одновременно значительно упростить код. Достигается это тем, что свойству (любому свойству любого объекта) присваивается не готовое значение, а результат вычисления некоторого выражения.

Например, рассмотрим такой пример:

```
par1.style.width = document.body.style.pixelWidth / 2;
```

Здесь мы берем ширину документа, делим его пополам и присваиваем получившееся значение ширине некоторого абзаца. Вы видите, что для этого нам, возможно, понадобится отдельный скрипт.

А что вы скажете на это?

```
<P STYLE="width:expression(document.body.style.pixelWidth / 2)">
↳Это наш текст.</P>
```

Здесь мы задали вычисление необходимого выражения и присвоение его значения атрибуту (не свойству!) прямо в определении встроенного стиля. Согласитесь, что так намного компактнее и удобнее, чем создавать специальный скрипт.

Это и есть динамическое свойство.

Теперь давайте сформулируем только что узнанное. Итак, для того чтобы присвоить какому-либо свойству значение, вычисленное выражением, нужно воспользоваться следующим форматом.

```
{Имя атрибута стиля}:expression({Выражение})
```

Этот синтаксис поддерживается всеми элементами страницы и всеми атрибутами стиля.

Четыре новых метода

Все элементы страниц, входящие в объектную модель Internet Explorer, поддерживают четыре метода, специально определенные для работы с динамическими свойствами.

Метод `setExpression` предназначен для установки динамического свойства. Он может иметь два разных формата. При использовании первого формата называется свойство элемента страницы, чем достигается привязка выражения к названному свойству элемента. (Естественно, свойство должно быть доступно для чтения и записи.)

```
setExpression("{Имя свойства}", "{Выражение}"[, "{Язык}"])
```

Здесь в качестве языка можно задать "JavaScript", но это не обязательно.

```
par1.setExpression("innerText", "getSomeText(123)");
```

Здесь мы привязываем к свойству `innerText` абзаца функцию, результат вычисления которой будет присваиваться этому свойству.

При использовании второго формата вызывается метод `setExpression` объекта `style` (`currentStyle`), чем достигается привязка выражения к атрибуту стиля элемента страницы:

```
style.setExpression("{Имя атрибута}", "{Выражение}"[, "{Язык}"])
```

Например,

```
par1.style.setExpression("color", "getSomeColor()");
```

Если есть метод, привязывающий к атрибуту или свойству выражение, то должен быть также и метод, возвращающий текст этого выражения. Это `getExpression`. Его формат таков:

```
getExpression("{Имя свойства}")
```

Этот метод может быть вызван как для объекта элемента страницы, так и для style:

```
par1.getExpression("innerText");
```

Метод removeExpression позволит вам удалить динамическое свойство:

```
removeExpression("{Имя свойства}")
par1.removeExpression("innerText");
```

Есть еще один полезнейший метод, позволяющий перевычислить все выражения, привязанные к атрибутам и свойствам всех объектов. Обычно нужные выражения перевычисляются, когда Web-обозреватель сочтет необходимым, например, после изменений значений переменных, атрибутов или свойств, входящих в привязанные выражения. Но иногда могут произойти коллизии из-за того, что какое-то выражение, привязанное к динамическому свойству, использует значение другого динамического свойства, и первое выражение перевычисляется перед вторым. В этом случае страница будет отображена неверно. Для исключения подобных коллизий и предусмотрен метод recalc объекта document.

```
document.recalc([true|false]);
```

Как видите, этот метод может принимать один необязательный параметр. Если он равен true (значение по умолчанию), то перевычислятся только те выражения, аргументы которых изменились после предыдущего перевычисления. Если же он равен false, перевычислятся все выражения.

Пример страницы с центрированным элементом

Я хочу рассмотреть практический пример использования динамических свойств для решения проблем Web-дизайна, неразрешимых обычным, "чистым" HTML. Пусть это будет страница с рисунком, который всегда, при любых изменениях размера окна Web-обозревателя располагается точно в его центре.

Рассмотрим код нашей страницы построчно.

```
<HTML>
<HEAD>
<TITLE>Использование динамических свойств</TITLE>
<SCRIPT>
function setupElement() {
  centered.style.setExpression("top", "document.body.clientHeight / 2 -
↳centered.offsetHeight / 2");
  centered.style.setExpression("left", "document.body.clientWidth / 2 -
    ↳centered.offsetWidth / 2"); }
```

Здесь мы присваиваем элементу, который должен находиться в центре, динамические свойства, устанавливающие его местоположение. При каждом изменении размеров окна Web-обозревателя, в котором отображается документ, меняются координаты этого элемента.

```
</SCRIPT>
</HEAD>
<BODY onLoad="setupElement();">
<DIV ID="centered" STYLE="position: absolute; top: 0; left: 0">
```

Для того чтобы управлять местоположением элемента страницы, его свойство `position` должно быть установлено в `absolute` (или `relative`). Также необходимо установить свойства `top` или `left`, чтобы впоследствии легко получить к ним доступ через объект `style`.

```
<IMG SRC="hlplogo.gif">
</DIV>
</BODY>
</HTML>
```

Сохраните этот код в файле под именем 3.8.htm и загрузите его в Internet Explorer. То, что вы увидите, показано на рис. 3.10.

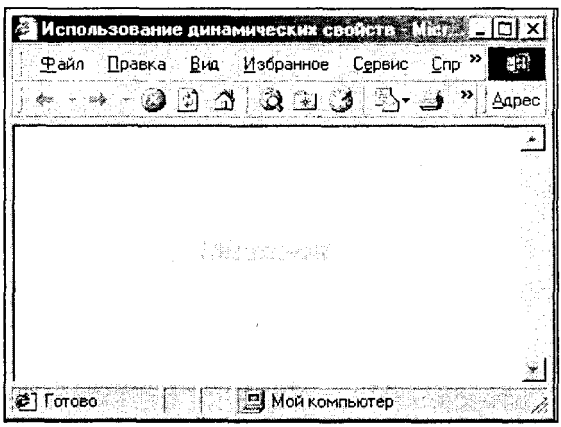


Рис. 3.10. Страница с центрированным элементом

Таким образом, применение динамических свойств упрощает решение многих задач Web-дизайна. Можно создавать Web-страницы с множеством свободно позиционированных элементов, координаты которых вычисляются специально написанными выражениями. В результате при любых изменениях размеров окна они будут позиционироваться правильно. Такое не позволяют делать даже настольные издательские программы, так что у Web-дизайнеров есть повод для гордости.

Пример страницы с динамическим содержимым

В конце, чтобы завершить обсуждение динамических свойств, рассмотрим пример Web-страницы, где выражения, привязанные к свойствам, определяют не внешний вид элементов, а их содержимое. Пусть эта страница отображает текущие дату и время, причем их значения будут изменяться. То есть, мы сделаем часы на Web-странице.

```
<HTML>
<HEAD>
<TITLE>Текущие дата и время</TITLE>
<SCRIPT>
function setupElement() {
  now.style.setExpression("top", "document.body.clientHeight / 2 –
    ↳now.offsetHeight / 2");
  now.style.setExpression("left", "document.body.clientWidth / 2 –
    ↳now.offsetWidth / 2");
```

Начало страницы организовано точно так же, как и в предыдущем примере.

```
now.setExpression("innerText", "currentDate()");
```

Задаем динамическое свойство, отображающее текстовое представление даты и времени.

```
window.setInterval("document.recalc();", 500); }
```

И не забываем каждые полсекунды обновлять это самое представление, чтобы наши часы "шли". Иначе пользы от них не будет. Интервал в полсекунды выбран из соображений точности и производительности: время будет отображаться относительно точно, почти секунда в секунду, и при этом наши скрипты не будут слишком "нагружать" клиентский компьютер.

```
function currentDate() {
  var d;
  d=new Date();
  return d.toLocaleString (); }
```

Ну а это уже совсем просто. Функция взята из самого первого нашего примера динамической страницы с минимальными изменениями.

```
</SCRIPT>
</HEAD>
<BODY onLoad="setupElement();">
<DIV ID="now" STYLE="font-size: 24pt; position: absolute; top: 0;
↳left: 0"></DIV>
</BODY>
</HTML>
```

Остается сохранить код в файле под 3.9.htm и загрузить его в Internet Explorer. А если вы не хотите этого делать, посмотрите на рис. 3.11.

Рис. 3.11. Страница-"часы"

Конечно же, это простейший пример использования динамических свойств. Я уверен, что вы сумеете придумать что-нибудь более сложное и несравнимо более красивое.

Вычисляемые атрибуты Navigator

После воздаяния заслуженных почестей Internet Explorer обратим наш снисходительный взор на Navigator. Конечно, он не поддерживает динамических свойств и многих других возможностей своего младшего (или уже старшего?) брата. Но кое-что динамичное он может вам предложить. Это так называемые вычисляемые атрибуты тегов.

Скажем, мы определили некий элемент Web-страницы с использованием тега `<H1>`:

```
<H1>Это заголовок</H1>
```

Как видите, это обычный статичный HTML-код без всяких признаков жизни. И после того, как он загружен в Navigator, мы больше не сможем его изменить.

Однако мы можем изменить его ДО того, как он загружен.

```
<SCRIPT>
var someText;
someText = "Это заголовок";
</SCRIPT>
. . .
<H1>&{someText};</H1>
```

Поясню, что за скрипт вы только что прочли. Вначале мы определяем переменную someText и присваиваем ей значение "Это заголовок". После этого в теле HTML-документа мы определяем заголовок первого уровня, и в качестве его содержимого подставляем значение этой переменной. Обратите внимание на формат записи, который используется.

Мы можем определять таким образом значения любых атрибутов любых тегов и любой текст.

```
<TABLE WIDTH="&{tableWidth};">
<I>Сегодня &{CurrentDate};</I>
```

Однако запомните, что определенный таким образом элемент страницы нединамичен. То есть, он не может изменить свое содержимое или свой вид после изменения значения переменной. Это основная беда Navigator.

Создание совместимых Web-страниц

Мы уже многому научились: познакомились с объектной моделью документа, рассмотрели основные объекты, предусматриваемые для управления документом и его составными частями из сценариев JavaScript, рассмотрели большинство необходимых для этого свойств и методов и создали несколько примеров. Казалось бы, все прекрасно, и будущее сулит нам только хорошее. Но нет. Проклятая несовместимость различных программ Web-обозревателей кому угодно отравит жизнь, а уж тем более Web-дизайнерам и Web-программистам. Слишком они разные: Netscape Navigator и Microsoft Internet Explorer...

Так можно ли побороть эту проблему, разрушить неодолимый барьер между этими программами-конкурентами, помирить друг с другом две несовместимые объектные модели? Или мир так и останется расколотым пополам?

Можно. Но только путем компромиссов.

Я дам несколько советов начинающим Web-дизайнерам, желающим привнести "жизнь" в свои творения. И если вы хотите, чтобы ваши Web-страницы просматривались или хотя бы выглядели нормально в разных программах Web-обозревателей, следуйте им.

1. Постарайтесь вообще обойтись без динамики в Web-страницах, если это возможно. Помните, что для пользователя главное — получить информацию, а не просмотреть ваш очередной супердинамический шедевр. Излишние "навороты" только затруднят чтение текста и ничего не дадут взамен.
2. Используйте альтернативные технологии придания страницам динамики, независимые от разных программ Web-обозревателей, например, Macromedia Flash. Страница, созданная с помощью Flash, будет красивой,

динамичной и совместимой со всеми Web-обозревателями, поддерживающими расширения (а таких сейчас большинство). Описание Flash выходит за рамки моей книги; поищите хорошую литературу по этому вопросу — сейчас ее много.

3. Перенаправляйте пользователя на разные страницы в зависимости от используемого им программного обеспечения. Хорошим решением является и генерация совместимой Web-страницы "на лету" с помощью серверных программ. Описание разработки серверных программ опять же не является задачей этой книги.
4. И наконец, если три предыдущих совета вам не подходят, пишите совместимые страницы. Используйте только те возможности, которые поддерживаются обеими программами или, по крайней мере, стремитесь к тому, чтобы страницы выглядели достойно в любой из них, даже если та или иная возможность данной программой не поддерживается. Комбинируйте HTML-код так, чтобы каждая программа читала только "свой" участок. Вообще, универсальных рецептов нет; вам придется или экспериментировать самим, или использовать чужой опыт.

Я советую вам крепко запомнить эти четыре принципа. Ведь самый плохой мир все же лучше самой хорошей войны, даже если это бескровная война стандартов DOM.

Ну что ж, теперь самое время дать пример динамической Web-страницы, совместимой и с Internet Explorer, и с Navigator. Давайте возьмем за образец нашу страницу с анимированным рисунком, точнее, обе ее версии, и попробуем сделать ее "дружелюбной" к разным программам. А для этого мы попробуем объединить несовместимый код.

Ниже приведен код новой страницы.

```
<HTML>
<HEAD>
<TITLE>Анимация на Web-странице</TITLE>
<STYLE>
DIV { font-size: 72; font-weight: bold }
#div1 { top: 50; left: 50; position: absolute; z-index: 1 }
#div2 { top: 50; left: 100; position: absolute; z-index: -1 }
#div3 { top: 50; left: 150; position: absolute; z-index: 1 }
#div4 { top: 50; left: 200; position: absolute; z-index: -1 }
</STYLE>
<SCRIPT>
var dx, timer, isNavigator;
dx = 2;
```

Здесь мы добавили новую переменную `isNavigator`, идентифицирующую программу Web-обозревателя.

```
function moveImage() {
  if (isNavigator) {
    document.livelayer.left += dx;
    if (document.livelayer.left >= document.width) dx = -dx;
    if (document.livelayer.left <= 0) dx = -dx; }
  else {
    livediv.style.pixelLeft += dx;
    if (livediv.style.pixelLeft + livediv.style.pixelWidth >=
    ↳document.body.clientWidth) dx = -dx;
    if (livediv.style.pixelLeft <= 0) dx = -dx; } }
```

Если страница загружена в Navigator, исполняется один код, если в Internet Explorer — другой.

```
function setupAnim() {
  timer = window.setInterval("moveImage()", 100);
  isNavigator = (navigator.appName.indexOf("Netscape") != -1); }
```

А эта функция теперь не только устанавливает таймер, но и определяет программу Web-обозревателя.

```
</SCRIPT>
</HEAD>
<BODY onLoad="setupAnim();" STYLE="font-size: 72pt; font-weight: bold">

<LAYER NAME="layer1" LEFT="50" TOP="20" Z-INDEX="2">
<NOLAYER>
<DIV ID="div1" STYLE="top: 50; left: 50; position: absolute; z-index: 1">
</NOLAYER>
J
<NOLAYER>
</DIV>
</NOLAYER>
</LAYER>
```

Какое-то дикое нагромождение тегов... Но на самом деле, все просто. Мы берем за основу Navigator, задаем для него слои, а весь несовместимый с ним (но совместимый с Internet Explorer) код помещаем в теги <NOLAYER>. Navigator обрабатывает теги <LAYER> и игнорирует содержимое тегов <NOLAYER>. Internet Explorer игнорирует теги <LAYER> и <NOLAYER> как неизвестные и обрабатывает содержимое тегов <NOLAYER>.

```
<LAYER NAME="layer2" LEFT="100" TOP="20" Z-INDEX="0">
<NOLAYER>
<DIV ID="div2" STYLE="top: 50; left: 100; position: absolute;
↳z-index: -1">
</NOLAYER>
```

```
a
<NOLAYER>
</DIV>
</NOLAYER>
</LAYER>

<LAYER NAME="layer3" LEFT="150" TOP="20" Z-INDEX="2">
<NOLAYER>
<DIV ID="div3" STYLE="top: 50; left: 150; position: absolute;
↳z-index: 1">
</NOLAYER>
v
<NOLAYER>
</DIV>
</NOLAYER>
</LAYER>

<LAYER NAME="layer4" LEFT="200" TOP="20" Z-INDEX="0">
<NOLAYER>
<DIV ID="div4" STYLE="top: 50; left: 200; position: absolute;
↳z-index: -1">
</NOLAYER>
a
<NOLAYER>
</DIV>
</NOLAYER>
</LAYER>

<LAYER NAME="livelayer" LEFT="0" TOP="60" Z-INDEX="1">
<NOLAYER>
<DIV ID="livediv" STYLE="top: 40; left: 0; width: 30;
↳position: absolute">
</NOLAYER>
<IMG SRC="tips.gif">
<NOLAYER>
</DIV>
</NOLAYER>
</LAYER></BODY>
</HTML>
```

Вот и все. Теперь можете сохранить этот код в файле 3.10.htm и поочередно открыть его в Internet Explorer и Navigator. Вы увидите то же самое, что изображено на рис. 3.6 и 3.7.

Повторяю, что в создании совместимых Web-страниц нет стандартных решений. Здесь в полной мере может проявиться мастерство Web-программиста, знатока особенностей реализации HTML в разных программах

Web-обозревателей. Конечно, существуют какие-то наработки, которые можно использовать как образец, но не более того. Поэтому, если вы хотите "помирить" разные, не очень-то совместимые друг с другом программы, придется поработать головой. А в результате может ничего и не получиться...

Да, программирование — это технология. А в каждой технологии, кроме голой науки и техники, есть место творчеству. И тем более — в технологии Web-дизайна. И уж, безусловно, — когда не существует простого пути реализации какой-либо нетривиальной задачи. Например, сделать Web-страницу, одинаково хорошо работающую и в Internet Explorer, и в Navigator.

Долго ли еще будет существовать эта объектно-документная разноголосица? Вряд ли...

Я пишу эти строки, когда Navigator и саму фирму Netscape уже можно считать достоянием истории. Мир захватил Internet Explorer; около 95% пользователей Интернета пользуются последней версией этой программы — 5.5. Даже недавний выпуск Navigator 6.0 не смог переломить ситуацию в пользу Netscape. Можно по-разному относиться к этому, но ясно одно: Web-программистам станет проще. Так же как Windows в свое время установила стандарт пользовательской оболочки, а потом и операционной системы, так и Internet Explorer установил стандарт программы Web-обозревателя, которого все отныне будут придерживаться.

И еще. Web-дизайнер, как и всякий творец, свободен в своем творчестве. Только он решает, что и для кого творить. Решайте сами и вы. Я только даю вам в руки инструмент — пользуйтесь им так, как подсказывает ваша интуиция и опыт.

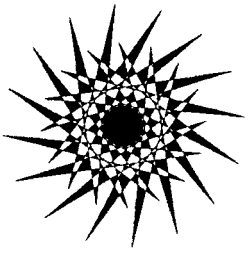
Хотите знать больше?

Ух, сколько динамических Web-страниц мы наделали — глаза разбегаются! Казалось, больше и нечего изучать, все понятно. Но нет. На самом деле, вся эта динамика — просто подобие жизни, но не сама жизнь.

Всякое живое существо тем или иным образом реагирует на воздействия окружающей среды. А нашим Web-страницам (за редким исключением) дела до нее нет. Ведь все программы (в том числе, и Web-страницы) создаются не просто так, а для пользователя. А значит, должны взаимодействовать с ним.

В следующей главе мы изучим понятие события, научимся обрабатывать события и создадим множество страниц, взаимодействующих с пользователем посредством событий.

ГЛАВА 4



События

Все, что мы создали в предыдущей главе книги, было динамичным: Web-страницы меняли свое содержимое, текст менял свой стиль, элементы летали взад-вперед, и даже сам Web-обозреватель приоткрывал нам свои секреты. Ничего схожего со статичным по самой своей природе HTML. Как говорится, всюду жизнь.

Но можно ли назвать живым существо, совершенно не реагирующее на внешние воздействия? Нет. Точно так и любая программа, в том числе и Web-страница. Без взаимодействия с пользователем они подобна йогу, впавшему в нирвану. Она не откликается ни на одно событие, произошедшее во внешнем, реальном мире...

Стоп! Ключевое слово в этой фразе: "событие". Давайте поговорим о событиях.

Обработка событий

Понятие события

Наступление того или иного *события* в системе может быть вызвано каким-либо действием пользователя или возникновением некоего условия внутри самой системы. В нашем случае событие может наступить при щелчке пользователя мышью по странице или какому-либо ее элементу, изменении размеров окна Web-обозревателя, нажатии на клавишу или окончании загрузки страницы. Можно дать такое определение события: это сигнал, формируемый операционной системой или прикладной программой и имеющий смысл "что-то произошло".

Большинство программ, работающих под Microsoft Windows, являются *событийно-управляемыми*. Каждая такая программа после запуска начинает

ждать адресованных ей событий. После того как событие получено, программа как-то реагирует на него или игнорирует его, в результате чего событие передается на обработку операционной системе. Программа может отреагировать на щелчок по кнопке, на выбор пункта меню, ввод символа в поле ввода и может проигнорировать сворачивание или разворачивание главного окна или изменение системного формата даты (если, конечно, на это не нужно как-то реагировать). Для того чтобы завершиться, программе передается специальное событие закрытия приложения, на которое ей опять-таки придется реагировать.

Web-страницы значительно отличаются от обычных Windows-программ. Но, тем не менее, они также могут реагировать на события: на щелчок мышью, на нажатие клавиши, на изменение размеров окна или на окончание загрузки Web-страницы. Различных событий, на которые может ответить Web-скрипт, очень много, и они с лихвой покроют потребности Web-программиста.

Как же Web-страница может ответить на то или иное событие? Очень просто. Сейчас мы это опишем. Но, прежде всего, немного освежим наши знания объектов вообще и объектной модели документа в частности.

Итак, объекты могут содержать внутри себя свойства — своего рода внутренние переменные, и методы — функции для работы с этими свойствами. В предыдущей главе мы много пользовались и тем и другим. Так вот: в JavaScript методы представляют собой обычные свойства, хранящие указатель на функцию, которая и реализует нужный метод. (Подробнее см. в главе 2, посвященной JavaScript.)

А теперь предположим, что в объекте, скажем, документа (`document`) предусмотрено несколько свойств, которым можно присвоить указатели на функции, выполняемые при наступлении того или иного события. И что, используя специальные соглашения, можно написать функции, откликающиеся на нужные события, и присвоить их этим свойствам.

Давайте рассмотрим один из наших примеров Web-страницы, отображающей текущую дату.

```
<HTML>
<HEAD>
<TITLE>Сегодня</TITLE>
<SCRIPT>
function writeDate() {
var d;
d=new Date();
dateishere.innerText = d.toLocaleString(); }
</SCRIPT>
</HEAD>
<BODY onload="writeDate()">
```

```
<P ID="dateishere">Здесь будет дата</P>
</BODY>
</HTML>
```

В этом примере используется обработка события. Не верите? Тогда посмотрите на следующую строчку:

```
<BODY onload="writeDate()">
```

А ведь это и есть присвоение событию `onload` функции-обработчика `writeDate`. Мы применили этакий "псевдоатрибут" `onload` для того, чтобы задать обработчик события `onLoad`, возникающего по окончании загрузки страницы.

Можно прибегнуть и к другому приему:

```
<HTML>
<HEAD>
<TITLE>Сегодня</TITLE>
<SCRIPT>
window.onload = writeDate;
function writeDate() {
var d;
d=new Date();
dateishere.innerText = d.toLocaleString(); }
</SCRIPT>
</HEAD>
<BODY>
<P ID="dateishere">Здесь будет дата</P>
</BODY>
</HTML>
```

Здесь мы *явно присвоили свойству* `onload` *объекта* окна указатель на функцию `writeDate`, говоря тем самым, что она будет выполняться при наступлении этого события. Можете использовать любой подход.

Точно таким же образом можно привязать обработчик события к другому элементу страницы. Например, рисунку:

```
<IMG SRC="someimage.gif"
↳onclick="document.location.href='somepage.htm';">
```

Здесь мы присвоили событию `onclick` рисунка обработчик, перенаправляющий пользователя на другую страницу. То есть, фактически, мы создали аналог гиперссылки, но средствами DOM и JavaScript, а не HTML.

Я уже говорил, что события могут возникать по самым разным причинам. Ниже мы перечислим некоторые из них и дадим краткое пояснение:

- *щелчок мышью по какому-либо из элементов страницы или по самой странице (фону).* Может использоваться для перехода на другую страницу

(аналог применения гиперссылок), выводу какого-либо пояснения (подобно тому, как мы выводили краткие пояснения при щелчках на гиперссылках в одном из примеров предыдущего раздела) и пр.;

- *нажатие клавиши клавиатуры.* Возможные области применения те же самые, что и в предыдущем случае;
- *перемещение мыши над элементами страницы или самой страницей.* Используется очень часто для создания так называемых "горячих" или "чувствительных" изображений-гиперссылок (английские Web-дизайнеры именуют их "rollover images" или просто "rollovers") и во многих других случаях. Ниже мы рассмотрим пример создания таких "горячих" гиперссылок;
- *окончание загрузки Web-страницы или файла изображения.* Мы уже использовали это событие для различных начальных установок.

Конечно, существуют и другие события, специфичные для того или иного вида элементов страницы или объектов. Но мы рассмотрим их позднее. А пока давайте перечислим события, поддерживаемые всеми элементами страницы.

Общие события

Так называют события, поддерживаемые всеми элементами страницы. Разумеется, это справедливо для Internet Explorer. Navigator же поддерживает их для очень небольшого круга объектов, а именно для документа, окна, рисунков, гиперссылок и некоторых других. Более полная информация по событиям и их поддержке разными объектами приведена в приложении 3.

Большинство общих событий рассмотрено в табл. 4.1.

Таблица 4.1. *Общие события*

Событие	Описание
`onclick`	Наступает при щелчке мышью на элементе страницы или на самой странице
`oncontextmenu`	Наступает, когда пользователь щелкает по странице или одному из ее элементов правой кнопкой мыши, чтобы вывести контекстное меню. Поддерживается только Internet Explorer начиная с 5.0
`ondblclick`	Наступает при двойном щелчке мышью на элементе. Может применяться для создания элементов, не реагирующих на одинарный щелчок, но реагирующих на двойной
`onhelp`	Наступает, когда пользователь нажимает клавишу <F1>. Поддерживается только Internet Explorer начиная с 4.0

Таблица 4.1 (окончание)

Событие	Описание
onkeydown	Наступает, когда пользователь нажимает и удерживает клавишу на клавиатуре
onkeypress	Наступает, когда пользователь нажимает клавишу на клавиатуре. От onkeydown отличается тем, что возвращает значение кода символа в кодировке Unicode
onkeyup	Наступает, когда пользователь отпускает нажатую ранее клавишу на клавиатуре
onmousedown	Наступает, когда пользователь нажимает кнопку мыши на элементе страницы или на самой странице
onmousemove	Наступает, когда пользователь перемещает мышь над элементом страницы или над самой страницей. Это событие вызывается при любом перемещении мыши
onmouseout	Наступает, когда пользователь убирает мышь с элемента страницы
onmouseover	Наступает, когда пользователь помещает мышь над элементом страницы
onmouseup	Наступает, когда пользователь отпускает ранее нажатую кнопку мыши
onpropertychange	Наступает, когда изменяется значение какого-либо атрибута тега или стиля или свойство элемента страницы. Поддерживается только Internet Explorer начиная с 5.0
onscroll	Наступает, когда пользователь прокручивает содержимое элемента страницы, документа, окна или фрейма. Поддерживается только Internet Explorer начиная с 4.0
onselectstart	Наступает, когда пользователь начинает выделять какой-либо текст на странице или в одном из ее элементов. Поддерживается только Internet Explorer начиная с 4.0

Как видите, многие события вызываются подряд друг за другом. Например, при перемещении мыши над элементом страницы последовательность наступления событий будет такой:

onmouseover,
onmousemove,
onmouseout.

Заметьте, что onmousemove наступает при любом, даже самом малом перемещении мыши.

При щелчке мышью на объекте последовательность событий такова:

```
onmousedown,
onmouseup,
onclick.
```

т. е. код, привязанный к `onmousedown` и `onmouseup`, будет выполняться прежде кода, привязанного к `onclick`.

Если пользователь щелкает дважды, то последовательность событий следующая:

```
onmousedown,
onmouseup,
onclick,
ondblclick.
```

Как видите, при двойном щелчке раньше события `ondblclick` наступает `onclick`. Это можно использовать в скриптах и, вместе с тем, придется это учитывать, чтобы не прореагировать на обычный щелчок, если нужна реакция на двойной.

При нажатии клавиши на клавиатуре последовательность наступления событий такова:

```
onkeydown,
onkeypress,
onkeyup.
```

При этом событие `onkeypress` наступает постоянно, пока пользователь не отпустит клавишу. Это происходит вследствие так называемого автоповтора, когда, нажав и удерживая клавишу, можно добиться многократного и очень частого ее срабатывания. Это очень полезная возможность, которую можно также применять в скриптах.

Событие `oncontextmenu` вы можете использовать для того, чтобы запретить показ контекстного меню при щелчке правой кнопкой мыши. Это может понадобиться, например, для того, чтобы скрыть от пользователя HTML-код вашей страницы (в контекстном меню есть пункт "В виде HTML", открывающий исходный код страницы в Блокноте). Для этого достаточно написать следующий код:

```
<DIV ID="secretDiv" oncontextmenu="return false;">...</DIV>
```

Я уже говорил, что Navigator поддерживает события далеко не для всех элементов страницы. В табл. 4.2 приведен список объектов, для которых поддерживается то или иное событие. Полную информацию о событиях и их поддержке в Navigator см. в приложении 3.

Таблица 4.2. Поддержка событий в Navigator

Событие	Объекты, которые его поддерживают
onclick	Гиперссылки
ondblclick	Гиперссылки
onkeydown	Документ (document), изображения, гиперссылки
onkeypress	Документ (document), изображения, гиперссылки
onkeyup	Документ (document), изображения, гиперссылки
onmousedown	Документ (document), гиперссылки
onmousemove	Не поддерживается ни одним объектом. Требует перехвата события, о котором см. ниже.
onmouseout	Гиперссылки, карты-изображения
onmouseover	Гиперссылки
onmouseup	Документ (document), гиперссылки

Простейший пример Web-страницы

Теперь давайте рассмотрим пример Web-страницы, в которой используются обработчики событий. Пока что это будет самая простая страница, реагирующая на перемещения и щелчки мыши и выделение текста в абзацах. Она будет включать в себя три абзаца, расположенные в таблице для удобства форматирования. При помещении курсора мыши над каждым из этих абзацев в статусной строке появится содержащийся в нем текст, при уводе мыши с него — обычный текст по умолчанию. При попытке выделения текста любого из абзацев изменится его стиль.

Эта страница, как и многие другие, созданные нами, поддерживается только Internet Explorer.

```
<HTML>
<HEAD>
<TITLE>События 1</TITLE>
<STYLE>
P { font-size: 24pt }
.par { color: black; background-color: white }
```

Стиль .par мы применим к обычному, невыделенному абзацу.

```
.par_sel { color: white; background-color: blue }
```

А это — стиль выделенного абзаца. Текст будет похож на выделенный: белые символы на синем фоне.

```
</STYLE>
<SCRIPT>
function parOver(par) { window.status = par.innerText; }
```

Функция `parOver` помещает в строку состояния окна Web-обозревателя текст соответствующего абзаца. Здесь для нас нет ничего нового.

```
function parOut(par) {
  window.status = window.defaultStatus;
  par.className = "par"; }
```

Снова все знакомо. При уводе мыши с абзаца в строку состояния помещается стандартный текст по умолчанию, выводимый Web-обозревателем. Также следует предусмотреть смену стиля текста на обычный, невыделенный, если перед этим мы пытались выделить текст, и скрипт изменил его стиль.

```
function parSel(par) { par.className = "par_sel"; }
```

Опять же все знакомо: мы просто меняем стиль текста абзаца, который пользователь пытается выделить.

```
</SCRIPT>
</HEAD>
<BODY>
<TABLE>
<TR><TD><P ID="par1" CLASS="par" onmouseout="parOut(par1);"
↳onmouseover="parOver(par1);"
↳onselectstart="parSel(par1);">Абзац 1</P></TD></TR>
```

Первый из наших абзацев. Три обработчика событий, которым присвоены три функции. В каждую функцию передается имя абзаца, заданное атрибутом `ID`. Остальные два абзаца почти точно такие же:

```
<TR><TD><P ID="par2" CLASS="par" onmouseout="parOut(par2);"
↳onmouseover="parOver(par2);"
↳onselectstart="parSel(par2);">Абзац 2</P></TD></TR>
<TR><TD><P ID="par3" CLASS="par" onmouseout="parOut(par3);"
↳onmouseover="parOver(par3);"
↳onselectstart="parSel(par3);">Абзац 3</P></TD></TR>
</TABLE>
</BODY>
</HTML>
```

Это весь код. Сохраните его под именем 4.1.htm и откройте в Internet Explorer. У вас должно получиться то, что показано на рис. 4.1. Попробуйте перемещать курсор мыши над абзацами, попытайтесь выделить текст одного из них и посмотрите, что из этого получится.

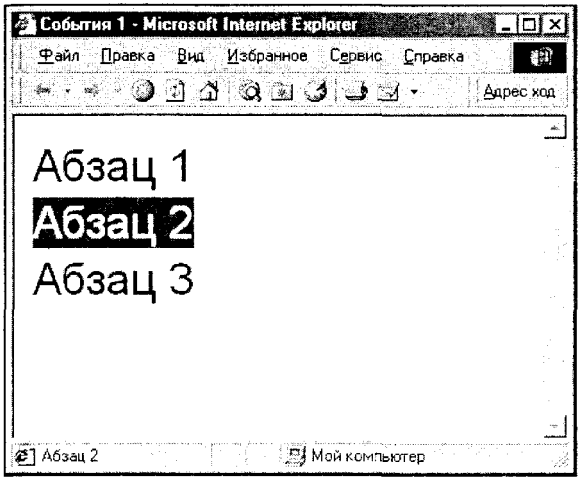


Рис. 4.1. Web-страница, реагирующая на события

Итак, наша первая истинно динамичная, "живая" страница готова. Но прежде чем продолжить в том же духе и сделать что-нибудь посложнее, рассмотрим еще один очень важный момент.

Динамические стили Internet Explorer

Internet Explorer позволяет динамически изменять стили элементов страницы или всего документа в ответ на наступление какого-либо события. Все это выполняется очень просто.

Скажем, мы хотим, чтобы при перемещении курсора над заголовком менялся цвет его шрифта. Для этого пишем следующий код:

```
<H1 onmouseover="this.style.color = 'green';">Это заголовок</H1>
```

Здесь мы используем переменную `this`, чтобы сослаться на объект, вызвавший наступление события. Как видите, этот код ничем не отличается от любого другого выражения JavaScript. Вместо него можно было подставить вызов функции-обработчика, выполняющей те же самые действия. Но приведенный выше код — согласитесь! — намного компактнее.

Продолжим менять стили элементов. Теперь давайте сделаем гиперссылку, но не средствами HTML, а средствами JavaScript. Для этого перехватим два события: `onmouseover` и `onclick`:

```
<P onmouseover="this.style.textDecoration = 'underline';"
↳onclick="location.href = 'otherpage.htm';">Это гиперссылка</P>
```

Этот абзац текста будет работать как гиперссылка, хотя на самом деле он ею не является. Вся функциональность гиперссылки добавляется исключительно методами JavaScript. При наведении курсора мыши текст абзаца подчер-

кивается, а при щелчке происходит переход на другую страницу. Вы можете добавить и другие возможности, скажем, изменение цвета текста при щелчке.

И наконец, глупая шутка: документ, чье содержимое сразу же после загрузки... пропадает:

```
<BODY onload="this.style.visibility = 'hidden';">
```

Никогда так не делайте, посетители вашего сайта, скорее всего, не оценят ваш юмор.

Объект *event*

Очень часто вместе с событием передаются дополнительные данные. В частности, вместе с событием `onclick` передаются координаты курсора мыши и номер кнопки, нажатой пользователем, а вместе с `onkeypress` — код нажатой пользователем клавиши. Для того чтобы скрипт смог получить доступ к этим данным, предусмотрен объект `event`.

Ссылку на объект `event` в Internet Explorer можно получить, обратившись к свойству `event` объекта `window`:

```
x = window.event.clientX;
```

В Navigator ссылка на объект `event` передается в функцию обработки события в качестве параметра:

```
function handleEvent(someEvent) { x = someEvent.pageX; }
```

Как видите, поддержка объекта `event` в Internet Explorer и Navigator различается очень существенно. Поэтому мы рассмотрим ее отдельно для каждой из этих программ.

Объект *event* в Internet Explorer

Сначала рассмотрим реализацию объекта `event` в Internet Explorer.

Свойства

Свойства, поддерживаемые объектом `event` в Internet Explorer, перечислены в табл. 4.3. Методов этот объект не поддерживает.

Таблица 4.3. *Свойства объекта* `event` *(Internet Explorer)*

Свойство	Описание
`altKey`	Возвращает `true`, если была нажата клавиша <Alt>
`altLeft`	Возвращает `true`, если была нажата левая клавиша <Alt>, и `false`, если правая

***Таблица 4.3** (продолжение)*

Свойство	Описание
button	Возвращает номер кнопки мыши, нажатой пользователем. Список возвращаемых значений приведен в табл. 4.4
cancelBubble	Задает, прерывать или не прерывать дальнейшее прохождение событий. О прохождении событий и управлении им см. ниже
clientX	Возвращает горизонтальную координату курсора мыши относительно клиентской области окна (без учета рамок, заголовка, строки меню, панелей инструментов и строки состояния)
clientY	Возвращает вертикальную координату курсора мыши относительно клиентской области окна (без учета рамок, заголовка, строки меню, панелей инструментов и строки состояния)
ctrlKey	Возвращает true, если была нажата клавиша <Ctrl>
ctrlLeft	Возвращает true, если была нажата левая клавиша <Ctrl>, и false, если правая
fromElement	Возвращает ссылку на элемент, с которого переместился курсор мыши при наступлении события onmouseover или onmouseout
keyCode	Возвращает код нажатой клавиши клавиатуры в кодировке Unicode
offsetX	Возвращает горизонтальную координату курсора мыши относительно элемента страницы, вызвавшего это событие
offsetY	Возвращает вертикальную координату курсора мыши относительно элемента страницы, вызвавшего это событие
propertyName	Возвращает имя атрибута тега или стиля или свойства элемента страницы, значение которого изменилос
repeat	Возвращает true, если событие onkeypress наступило повторно вследствие того, что пользователь удерживает клавишу нажатой, и false — в противном случае
returnValue	Задает, будет ли выполняться действие по умолчанию для элемента страницы. Подробнее см. ниже.
screenX	Возвращает горизонтальную координату курсора мыши относительно экрана
screenY	Возвращает вертикальную координату курсора мыши относительно экрана
shiftKey	Возвращает true, если была нажата клавиша <Shift>
shiftLeft	Возвращает true, если была нажата левая клавиша <Shift>, и false, если правая

Таблица 4.3 (окончание)

Свойство	Описание
srcElement	Возвращает ссылку на элемент страницы, вызвавший наступление события
toElement	Возвращает ссылку на элемент страницы, на который пользователь перемещает курсор мыши
type	Возвращает имя события без приставки "on"
x	Возвращает горизонтальную координату курсора мыши относительно родительского элемента
y	Возвращает вертикальную координату курсора мыши относительно родительского элемента

Таблица 4.4. Номера кнопок мыши, возвращаемых свойством button

№	Описание
0	Ничего не было нажато. Значение по умолчанию
1	Была нажата левая кнопка
2	Была нажата правая кнопка
3	Были одновременно нажаты левая и правая кнопки
4	Была нажата средняя кнопка
5	Были одновременно нажаты левая и средняя кнопки
6	Были одновременно нажаты правая и средняя кнопки
7	Были одновременно нажаты все кнопки

Здесь стоит дать дополнительные пояснения по свойству returnValue. Это свойство позволяет управлять действием по умолчанию, определенным для элемента страницы. Значение true вызывает выполнение действия по умолчанию, а false — отменяет его. Например, для гиперссылки действие по умолчанию — переход на другую страницу. И мы можем его отменить.

```
<A HREF="somepage.htm" click="window.event.returnValue = false;">Другая
↳страница</A>
```

В этом небольшом примере мы отменяем действие по умолчанию для гиперссылки. И пользователь больше никогда не попадет на страницу somepage.htm, если мы не позволим ему.

Есть еще один способ отменить для элемента страницы действие по умолчанию.

```
<A HREF="somepage.htm" click="return false;">Другая страница</A>
```

Как видите, здесь мы просто возвращаем `false`. С тем же результатом, что и в предыдущем случае.

Прохождение событий

А сейчас мы рассмотрим очень важную тему — *прохождение событий* по объектам, входящим в объектную модель документа, и управление им. Понимая, как это происходит, вы сможете добиться нужного результата несравнимо меньшим количеством кода и избегнуть многих досадных ошибок. Прохождение событий поддерживается только Internet Explorer.

Давайте рассмотрим небольшой пример Web-страницы.

```
<HTML>
<HEAD>
<TITLE>События 2</TITLE>
<STYLE>
#obj { background-color: gray; left: 20; top: 20; width: 200;
↳height: 100; position: absolute }
#pic { left: 30; top: 30; position: absolute }
</STYLE>
<SCRIPT>
function picClick() { window.alert("Рисунок!!!"); }
function objClick() { window.alert("Элемент страницы!!!"); }
function pageClick() { window.alert("Страница!!!"); }
</SCRIPT>
</HEAD>
<BODY onclick="pageClick();">
<DIV ID="obj" onclick="objClick();">
<IMG SRC="hlplogo.gif" ID="pic" onclick="picClick();">
</DIV>
</BODY>
</HTML>
```

Вряд ли понимание этого кода вызовет у вас затруднения. Мы просто создаем свободно позиционированный элемент, а в нем — свободно позиционированный рисунок. К телу документа, к элементу и к рисунку привязаны обработчики события `onclick`, выводящие на экран окна-предупреждения. Фон элемента страницы мы сделали серым, чтобы его было сразу заметно.

Сохраните эту страницу в файле под именем 4.2.htm и откройте в Internet Explorer. Вы должны увидеть то же самое, что показано на рис. 4.2.

Теперь попробуйте щелкнуть по пустому полю страницы. Что при этом случится? Правильно, на экране появится окно-предупреждение, сообщающее о том, что пользователь щелкнул по телу документа. Никакого сюрприза — все работает так, как было нами запланировано.

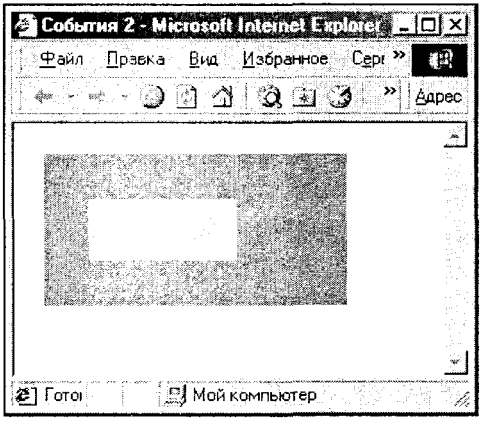


Рис. 4.2. Пример управления прохождением событий

Но щелкните теперь по серому полю свободно позиционированного элемента (не по рисунку). Да, на экране появится предупреждение, что пользователь щелкнул по серому элементу. Закройте его, нажав кнопку **OK**, и на экране появится еще одно предупреждение, что пользователь якобы щелкнул еще и по телу документа. В чем дело?

Оказывается, что каждое событие передается вверх по объектной иерархии. Говоря простыми словами, сначала оно передается элементу, вызывавшему событие (в нашем случае — серому элементу), а потом — его родителю (телу документа). И каждый из этих объектов может обработать это событие по-своему. Что они и делают.

В нашем случае событие `onclick` сначала передается серому элементу, который обрабатывает его, выводя на экран окно-предупреждение. После этого оно передается телу документа, которое тоже обрабатывается им. Вот поэтому мы и получаем сразу два предупреждения подряд.

А если теперь щелкнуть по рисунку? Пожалуйста! Только теперь предупреждений будет три: от самого рисунка, от серого элемента и тела документа. То есть, в этом случае событие прошло все объекты, составляющие страницу: рисунок, серый элемент и тело документа. И каждый из них его обработал.

Вы скажете: ну и придумали эти разработчики, с обработкой событий и так ничего не понятно, так они еще туману напускают. На самом деле, прохождение (иногда еще говорят "всплытие") событий — отличная вещь, если распорядиться ею с умом. Но иногда действительно бывает нужно, чтобы события не проходили дальше по объектной иерархии. Для этого объект `event` предусматривает свойство `cancelBubble`. По умолчанию его значение равно `false`, что означает полное прохождение событий. Если вы не хотите, чтобы события шли дальше, просто присвойте ему `true`.

Давайте исправим наш пример так, чтобы в любом случае выдавалось всего одно предупреждение. Здесь приведен только код скриптов, которым нужно заменить исходные скрипты:

```
<SCRIPT>
function picClick() {
  window.alert("Рисунок!!!");
  window.event.cancelBubble = true; }
function objClick() {
  window.alert("Элемент страницы!!!");
  window.event.cancelBubble = true; }
function pageClick() {
  window.alert("Страница!!!");
  window.event.cancelBubble = true; }
</SCRIPT>
```

Измените код страницы, сохраненный в файле 4.2.htm, загрузите его в Internet Explorer и пощелкайте по разным местам страницы. Вы увидите, что в любом случае выдается только одно предупреждение.

Страница с перетаскиваемыми элементами

А этот пример будет действительно сложным. Мы создадим Web-страницу с несколькими элементами, которые можно перетаскивать. Также мы добавим код, отображающий в строке состояния окна Web-обозревателя текущие координаты курсора мыши.

Поскольку текст нашей страницы не очень велик, приведем его полностью.

```
<HTML>
<HEAD>
<TITLE>События 3</TITLE>
<STYLE>
DIV { background-color: teal;
  position: absolute }
```

Чтобы несколько раз не задавать одни и те же стилевые настройки для каждого элемента, мы вынесли их в таблице стилей.

```
</STYLE>
<SCRIPT>
function bodyMouseMove() {
```

Перетаскивание элементов страницы реализует одна-единственная функция — обработчик события `onmousemove` тела документа. Здесь мы в полной мере используем преимущества прохождения событий. Даже если событие будет передано элементу, впоследствии оно в любом случае попадет к телу документа, которое его и обработает.

```
var obj;
window.status = "X: " + window.event.clientX + "; Y: "+
↪window.event.clientY;
```

Приведенный выше фрагмент кода показывает координаты курсора мыши в строке состояния.

```
obj = window.event.srcElement;
if (obj.tagName != "BODY" && window.event.button == 1) {
```

Если событие было передано впервые не телу документа, выполняем код перетаскивания:

```
obj.style.pixelLeft = window.event.x - obj.style.pixelWidth / 2;
obj.style.pixelTop = window.event.y - obj.style.pixelHeight / 2; } }
```

Позиционируем элемент так, чтобы его центр совпадал с курсором мыши.

```
</SCRIPT>
</HEAD>
<BODY onmousemove="bodyMouseMove();">
<DIV ID="div1" STYLE="left: 20; top: 20; width: 30; height: 40"></DIV>
<DIV ID="div2" STYLE="left: 120; top: 20; width: 30; height: 40"></DIV>
<DIV ID="div3" STYLE="left: 20; top: 120; width: 30; height: 40"></DIV>
<DIV ID="div4" STYLE="left: 120; top: 120; width: 30; height: 40"></DIV>
```

Это элементы страницы, которые мы и будем перетаскивать с места на место. Обратите внимание на определения стилей: некоторые свойства будут унаследованы из таблицы стилей.

```
</BODY>
</HTML>
```

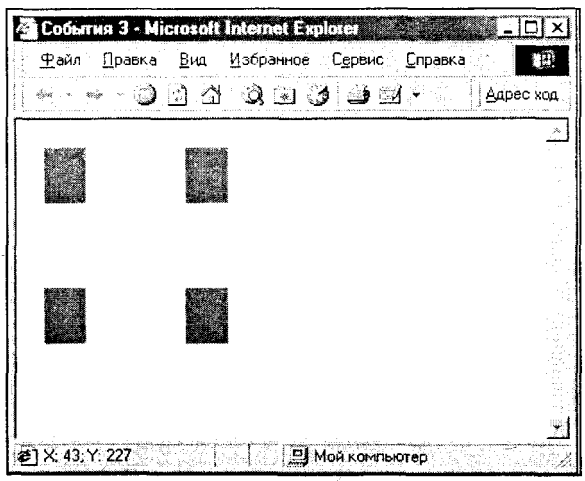


Рис. 4.3. Страница с перетаскиваемыми элементами

Вот и весь код. Сохраните его в файле 4.3.htm и откройте в Internet Explorer. То, что появится на экране, показано на рис. 4.3. Попробуйте переместить какой-либо из ее элементов и посмотрите, что получится.

Страница с рисунком-указателем

Очень часто на Web-сайтах можно видеть такую картину: небольшой рисунок "ползает" следом за курсором мыши. Он словно помогает курсору указывать на что-либо, чтобы пользователь, не дай бог, этого не пропустил. Маленький рисунок-указатель выглядит весьма эффектно, не загружает компьютер и не требует слишком уж сложного кодирования. Сейчас мы рассмотрим, как такое можно сделать.

Как обычно, в качестве примера создадим небольшую страницу.

```
<HTML>
<HEAD>
<TITLE>Рисунок-указатель</TITLE>
<STYLE>
IMG { position: absolute }
```

В принципе, эту стилевую установку можно было перенести во встроенный стиль. Ведь у нас только один элемент со свободным позиционированием.

```
</STYLE>
<SCRIPT>
var timID, x, y, ix, iy;
```

Объявляем глобальные переменные таймера и координат.

```
function setupAnim() { timID = window.setInterval("imgMove()", 50); }
```

Эта функция выполняется при загрузке страницы и создает таймер.

```
function imgMove() {
  if (x >= ix) {
    if (x - ix > 4)
      ix += 4
    else
      ix = x; }
```

Движение нашего рисунка-указателя осуществляется небольшими приращениями по 4 пиксела (или меньше).

```
else {
  if (ix - x > 4)
    ix -= 4
  else
    ix = x; }
if (y >= iy) {
```

```
  if (y - iy > 4)
    iy += 4
  else
    iy = y; }
else {
  if (iy - y > 4)
    iy -= 4
  else
    iy = y; }
```

Весь этот код вычисляет приращение движения, сообразуясь с направлением движения курсора мыши.

```
pImage.style.pixelLeft = ix - pImage.style.pixelWidth;
pImage.style.pixelTop = iy; }
```

А здесь мы собственно устанавливаем новые координаты рисунка-указателя. Причем, устанавливаем их таким образом, чтобы на месте курсора находился правый верхний угол рисунка.

```
function bodyMouseMove() {
  x = window.event.clientX;
  y = window.event.clientY; }
```

Сохраняем координаты курсора мыши для использования в процедуре движения рисунка, т. к. там мы уже не будем иметь доступа к объекту event.

```
</SCRIPT>
</HEAD>
<BODY onload="setupAnim();" onmousemove="bodyMouseMove ();">
<IMG SRC="tips.gif" ID="pImage" STYLE="top: 0; left: 0; width: 30">
</BODY>
</HTML>
```

Вот и весь код. Сохраните его в файле 4.4.htm и откройте в Internet Explorer. То, что вы увидите, показано на рис. 4.4; к несчастью, бумага не передает движения, как при любом перемещении курсора мыши наш рисунок "спешит" к нему.

Для использования рисунка-указателя в реальной Web-странице вам, возможно, придется поэкспериментировать с параметрами интервала таймера и шага движения рисунка. Исходите из того, как много процессорного времени отнимает ваша суперанимированная страница и насколько плавно движется рисунок-указатель. И помните, что главное в Web-странице — не чудеса анимации, не шикарное оформление, а информация, что размещена на ней.

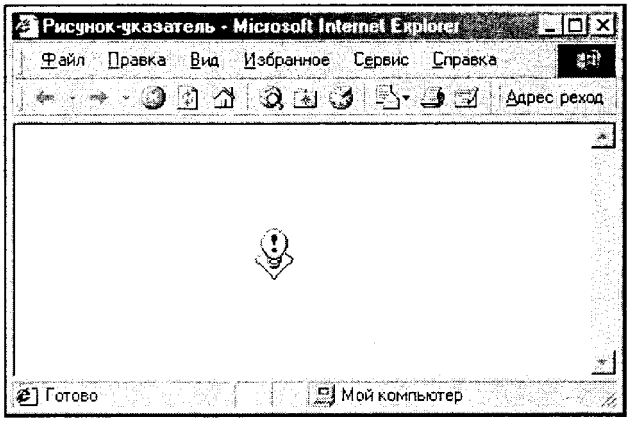


Рис. 4.4. Страница с рисунком-указателем

Объект *event* в Navigator

А теперь настала пора поговорить о том, как объект `event` реализован в Navigator.

Свойства

Свойства, поддерживаемые объектом `event` в Navigator, перечислены в табл. 4.5. Методов этот объект не поддерживает.

Таблица 4.5. *Свойства объекта* `event` *(Navigator)*

Свойство	Описание
`height`	Возвращает высоту окна или фрейма
`layerX`	Возвращает либо ширину объекта, либо горизонтальную координату курсора мыши относительно слоя, вызвавшего событие
`layerY`	Возвращает либо высоту объекта, либо вертикальную координату курсора мыши относительно слоя, вызвавшего событие
`modifiers`	Возвращает строку, представляющую нажатую клавишу-модификатор. Доступно одно из трех возвращаемых значений: `"ALT_MASK"`, `"SHIFT_MASK"`, `"CONTROL_MASK"`
`pageX`	Возвращает горизонтальную координату курсора мыши относительно страницы
`pageY`	Возвращает вертикальную координату курсора мыши относительно страницы
`screenX`	Возвращает горизонтальную координату курсора мыши относительно экрана

Таблица 4.5 (окончание)

Свойство	Описание
screenY	Возвращает вертикальную координату курсора мыши относительно экрана
target	Возвращает строку, представляющую имя объекта, которому было послано событие
type	Возвращает строку, представляющую тип события
which	Возвращает либо код нажатой клавиши, либо номер нажатой кнопки мыши. В последнем случае 1 обозначает левую кнопку, 2 — среднюю, а 3 — правую
width	Возвращает ширину окна или фрейма
x	То же самое, что layerX
y	То же самое, что layerY

Как видите, объекты event в Internet Explorer и Navigator поддерживаются совершенно по-разному. Конечно, есть общие свойства, и это можно использовать при создании совместимых Web-страниц. Но в дальнейшем имейте в виду эти различия. (Хотя о чем я говорю: несовместимость двух популярнейших программ Web-обозревателей уже стала притчей во языцех.)

Navigator не поддерживает прохождение событий, как Internet Explorer. Вместо этого он предусматривает так называемый перехват событий, который мы рассмотрим ниже.

Перехват событий

Как правило, событие передается тому объекту, который его вызвал. Но иногда необходимо обрабатывать все события в одном месте, скажем, в объекте document. В Internet Explorer с этим нет никаких проблем: просто определяются необходимые функции-обработчики и привязываются потом к нужным событиям объекта document. При этом события, происходящие в системе, рано или поздно будут переданы документу и обработаны им. В Navigator все обстоит несколько сложнее: если вы хотите обрабатывать все события в одном месте, вам придется специально дать понять это Web-обозревателю. И реализуется это с помощью так называемого *перехвата событий.*

Перехватывать события в Navigator умеют объекты документа (document), окна (window) и, с некоторыми ограничениями, слоя (layer). Все эти объекты поддерживают ряд методов, которые мы опишем ниже.

Для того чтобы документ или окно получили возможность перехватывать события, произошедшие в дочерних элементах, нужно вызвать метод captureEvents. Он имеет следующий формат:

```
captureEvents({Список типов событий, разделенных вертикальной чертой})
```

Типы событий записываются как свойства объекта event, большими буквами, без приставки "on". Например, чтобы дать документу возможность перехватывать все одинарные и двойные щелчки мыши, нужно написать следующее выражение:

```
document.captureEvents(event.CLICK | event.DBLCLICK);
```

После этого вы можете определить обработчик событий объекта тела документа, который будет перехватывать все щелчки мыши:

```
function bodyClicks (someEvent) . . .
document.onclick = bodyClicks;
```

Если вы хотите вернуть для обработки событие объекту, который его вызвал, воспользуйтесь методом routeEvent. Если объект, вызвавший событие, не имеет соответствующего обработчика, Web-обозреватель попытается найти другой объект, перехватывающий события (например, окно или фрейм).

```
routeEvent({Ссылка на объект event})
document.routeEvents(someEvent);
```

Иногда бывает необходимо направить событие для обработки объекту, не имеющему никакого отношения ни к вызвавшему событие объекту, ни к объекту-перехватчику событий. Для этого можно использовать метод handleEvent. Этот метод поддерживается всеми объектами документа Navigator: телом документа, окнами, фреймами, изображениями, гиперссылками, слоями и т. п.

```
handleEvent({Ссылка на объект event})
document.image1.handleEvents(someEvent);
```

Если вы хотите отказаться от перехвата какого-то типа событий, используйте метод releaseEvents.

```
releaseEvents({Список типов событий, разделенных вертикальной чертой})
```

Вы можете отказаться от перехвата всех типов событий, которые были зарегистрированы ранее:

```
document.releaseEvents(event.CLICK | event.DBLCLICK);
```

или только некоторых из них:

```
document.releaseEvents(event.DBLCLICK);
```

А теперь мы рассмотрим пример динамической Web-страницы, реагирующей на события и совместимой с Navigator.

Простейшая "живая" страничка для Navigator

Создадим страницу, содержащую два слоя и отображающую координаты курсора мыши относительно слоя, в котором находится курсор мыши, или относительно окна, если курсор находится вне слоев. Код этой страницы довольно сложен.

```
<HTML>
<HEAD>
</TITLE>События 4</TITLE>
<SCRIPT>
var inLayer = false;
```

Эта переменная будет хранить `true`, если курсор мыши находится над одним из слоев.

```
function bodyMouseMove(ev) {
  if (inLayer)
    window.status = "Слой — X: " + ev.layerX + "; Y: " + ev.layerY
  else
    window.status = "X: " + ev.pageX + "; Y: " + ev.pageY; }
```

Эта функция просто отображает в строке состояния окна программы текущие координаты курсора мыши относительно окна или слоя, в зависимости от того, находится ли курсор над слоем или вне его.

```
document.captureEvents(Event.MOUSEMOVE);
```

Перехватываем событие движения мыши. Как вы помните, в Navigator оно по умолчанию не привязано ни к какому объекту, так что для нормальной его обработки нужно воспользоваться техникой перехвата.

```
document.onmousemove = bodyMouseMove;
```

Присваиваем указатель на функцию-обработчик событию.

```
</SCRIPT>
</HEAD>
<BODY>
<LAYER NAME="layer1" LEFT="20" TOP="20" WIDTH="100" BGCOLOR="silver"
↳onmouseover="inLayer = true;" onmouseout="inLayer = false;">
Создадим страницу, содержащую в себе два слоя и отображающую координаты
курсора мыши относительно слоя, в котором он находится, или
относительно окна, если курсор находится вне слоев.
</LAYER>
<LAYER NAME="layer2" LEFT="160" TOP="20" WIDTH="100" BGCOLOR="yellow"
↳onmouseover="inLayer = true;" onmouseout="inLayer = false;">
Код этой страницы довольно сложен и в полной мере использует технику
перехвата событий.
```

```
</LAYER>
</BODY>
</HTML>
```

Остальной код не вызовет никаких проблем. Это просто два слоя, содержащих текст.

Сохраните код страницы в файле 4.5.htm и откройте его в Navigator. Вы увидите то, что показано на рис. 4.5. Попробуйте перемещать курсор мыши по странице и слоям и смотрите, что будет показано в строке состояния.

Рис. 4.5. "Живая" страница для Navigator

Страница с рисунком-указателем для Navigator

Теперь давайте перепишем сделанную нами ранее страничку с рисунком-указателем, "спешащим" к курсору мыши. Собственно, переделок будет не очень много.

```
<HTML>
<HEAD>
<TITLE>Рисунок-указатель</TITLE>
<SCRIPT>
var timID, x, y, ix = 0, iy = 0;
```

```
function setupAnim() { timID = window.setInterval("imgMove()", 50); }

function imgMove() {
  if (x >= ix) {
    if (x - ix > 4)
      ix += 4
    else
      ix = x; }
  else {
    if (ix - x > 4)
      ix -= 4
    else
      ix = x; }
  if (y >= iy) {
    if (y - iy > 4)
      iy += 4
    else
      iy = y; }
  else {
    if (iy - y > 4)
      iy -= 4
    else
      iy = y; }
  document.pLayer.left = ix - 30;
  document.pLayer.top = iy; }
```

Свободно позиционированный рисунок мы заменили слоем, т. к. в Navigator только слои могут свободно позиционироваться. И заметьте, что мы непосредственно указали ширину слоя, т. к. у объекта `layer` нет свойства `width`.

```
function bodyMouseMove(ev) {
  x = ev.pageX;
  y = ev.pageY; }

document.captureEvents(Event.MOUSEMOVE);
document.onmousemove = bodyMouseMove;
```

Не забываем перехватить событие перемещения мыши.

```
</SCRIPT>
</HEAD>
<BODY onload="setupAnim();">
<LAYER NAME="pLayer" TOP="100" LEFT="100" WIDTH="30">
<IMG SRC="tips.gif">
</LAYER>
</BODY>
</HTML>
```

Вот и все. Опять же остальной код не должен вызвать у вас вопросов. Мы просто заменили свободно позиционированный рисунок слоем.

Сохраните код страницы в файле под именем 4.6.htm и откройте его в Navigator. Должно получиться что-то, похожее на рис. 4.6.

Простота переделки подобного кода может натолкнуть начинающего Web-программиста на создание страницы, одинаково работающей и в Navigator, и в Internet Explorer. Что ж, это вполне возможно. Возьмите за основу последний пример, рассмотренный в предыдущей главе, и попробуйте!

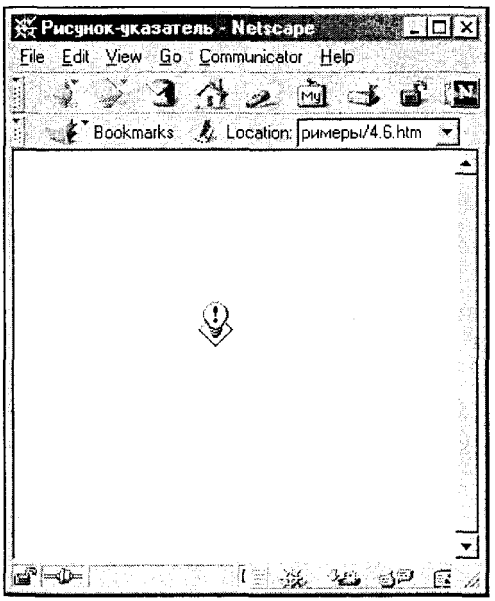


Рис. 4.6. Страница с рисунком-указателем для Navigator

Написание совместимых Web-страниц

Здесь мы рассмотрим написание Web-страниц, совместимых и с Internet Explorer, и с Navigator. И одновременно выясним, как реализуются "горячие" изображения-гиперссылки.

Очень часто Web-дизайнеры используют на своих сайтах "горячие" изображения-гиперссылки, меняющиеся при наведении на них курсора мыши. Эти изображения не требуют сложного кодирования и неплохо оживляют даже самую простенькую страничку (конечно, если их умело использовать). Единственный недостаток: необходимость иметь по два изображения одинакового размера на каждую гиперссылку.

Мы создадим простейшую страничку с тремя "горячими" изображениями-гиперссылками. И сделаем ее совместимой с Internet Explorer и с Navigator.

```
<HTML>
<HEAD>
```

```
<TITLE>&quot;Горячие&quot; изображения-гиперссылки</TITLE>
<SCRIPT>
var img2;
img2 = new Image;
img2.src = "hlplogo.gif";
```

Здесь мы создаем объект `Image` и загружаем в него файл изображения, которое будет появляться при наведении курсора мыши на гиперссылку. При этом Web-обозреватель сохраняет его в промежуточной памяти и впоследствии берет оттуда вместо того, чтобы долго загружать из Интернета. Аналогично мы поступали в предыдущей главе, когда рассматривали анимацию элементов страницы.

```
function changeImage(imageName, phase) {
  if (phase == 1)
    document.images[imageName].src = "hlplogo.gif"
  else
    document.images[imageName].src = "tips.gif"; }
```

Выше приведена функция, меняющая изображение на гиперссылке в зависимости от положения курсора мыши. Первым параметром передается имя изображения, вторым фаза: 1 означает обычное состояние, а 2 — "горячее", когда курсор мыши расположен над гиперссылкой.

```
</SCRIPT>
</HEAD>
<BODY>
<TABLE WIDTH="100%" HEIGHT="100%">
```

Изображения-гиперссылки находятся в трех ячейках таблицы, "растянутой" на всю страницу.

```
<TR ALIGN="center" VALIGN="middle">
<TD>
<A HREF="dummy.htm" onmouseover="changeImage('img1', 1);"
↳onmouseout="changeImage('img1', 2);">
```

Именно гиперссылка и реализует обработку событий.

```
<IMG SRC="tips.gif" NAME="img1" HEIGHT="20" WIDTH="50">
```

Рисунок просто украшает собой страницу. Обратите внимание на то, что мы жестко задаем размеры рисунка. Я сделал так потому, что у меня не нашлось двух рисунков одинакового размера, но это стоит делать в любом случае.

```
</A>
</TD>
<TD>
```

```
<A HREF="dummy.htm" onmouseover="changeImage('img2', 1);"
onmouseout="changeImage('img2', 2);">
<IMG SRC="tips.gif" NAME="img2" HEIGHT="20" WIDTH="50">
</A>
</TD>
<TD>
<A HREF="dummy.htm" onmouseover="changeImage('img3', 1);"
onmouseout="changeImage('img3', 2);">
<IMG SRC="tips.gif" NAME="img3" HEIGHT="20" WIDTH="50">
</A>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

Остальной код не содержит ничего нового. Мы просто повторяем нашу "горячую" гиперссылку еще два раза с разными параметрами обработчиков событий.

Сохраните код страницы в файле 4.7.htm и откройте его в Internet Explorer и Navigator. То, что должны вы увидеть, показано соответственно на рис. 4.7 и 4.8.

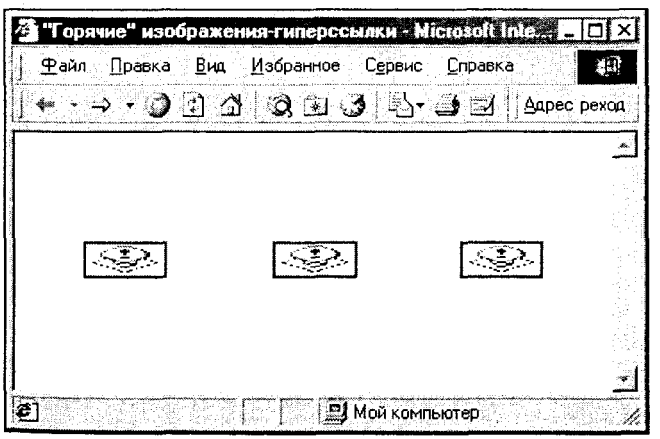


Рис. 4.7. Страница с "горячими" изображениями-гиперссылками (Internet Explorer)

Попробуйте поместить курсор мыши над одной из этих гиперссылок и посмотрите, как меняется изображение. Это работает и в Internet Explorer, и в Navigator, потому что мы руководствовались при создании этой страницы четвертым принципом создания совместимых страниц. Мы использовали только те возможности, которые поддерживают обе популярные программы Web-обозревателей.

Рис. 4.8. Страница с "горячими" изображениями-гиперссылками (Navigator)

События других объектов

В этом разделе будут описаны события, поддерживаемые другими объектами, входящими в объектную модель документа.

События объекта тела документа

Объект тела документа, образуемый тегом <BODY>, поддерживает ряд событий, перечисленных в табл. 4.6.

***Таблица 4.6.** События объекта тела документа*

Событие	Описание
onafterprint	Наступает сразу после распечатки документа или вывода его на предварительный просмотр. Поддерживается только Internet Explorer начиная с 5.0
onbeforeprint	Наступает непосредственно перед распечаткой документа или выводом его на предварительный просмотр. Поддерживается только Internet Explorer начиная с 5.0
onbeforeunload	Наступает непосредственно перед выгрузкой текущего документа, например, при переходе на другую страницу или закрытии окна. Поддерживается только Internet Explorer начиная с 4.0
onload	Наступает сразу по окончании загрузки документа

Таблица 4.6 (окончание)

Событие	Описание
onunload	Наступает непосредственно перед выгрузкой текущего документа, например, при переходе на другую страницу или закрытии окна. В Internet Explorer наступает после onbeforeunload

Нужно заметить, что события onload и onunload поддерживаются и Internet Explorer, и Navigator. Но если в Internet Explorer эти события принадлежат объекту тела документа body, то в Navigator — самому объекту document.

```
document.body.onload = bodyOnLoadForIE();
```

Так выглядит выражение присвоения функции-обработчика в Internet Explorer.

```
document.onload = bodyOnLoadForN();
```

А так — в Navigator.

Событие onload мы уже активно использовали в предыдущих разработках. Это одно из самых часто применяемых событий.

При помощи событий onunload и onbeforeunload можно, скажем, переназначить пользователя на другую страницу или выполнить какие-либо завершающие действия.

События onbeforeprint и onafterprint можно обработать, если требуется изменить текст или стиль печатного текста. В этом случае можно написать нечто подобное:

```
function bodyBeforePrint() {
  document.body.currentStyle.fontSize = "12pt"; }
function bodyAfterPrint() {
  document.body.currentStyle.fontSize = "9pt"; }
. . .
<BODY onbeforeprint="bodyBeforePrint()" onafterprint="bodyAfterPrint()">
```

События объекта *document*

Объект document в Navigator поддерживает события onload и onunload. В Internet Explorer реализовано только одно дополнительное событие, которое может нас заинтересовать, — onstop. Оно наступает, когда загрузка документа по тем или иным причинам останавливается, например, после нажатия пользователем кнопки **Стоп** на панели инструментов или разрыва соединения. Его можно использовать, скажем, чтобы сигнализировать о некорректной загрузке документа.

События объекта *window*

Объект окна программы Web-обозревателя поддерживает события, перечисленные в табл. 4.7.

Таблица 4.7. *События объекта* `window`

Событие	Описание
`onafterprint`	Наступает сразу после распечатки документа или вывода его на предварительный просмотр. Поддерживается только Internet Explorer начиная с 5.0
`onbeforeprint`	Наступает непосредственно перед распечаткой документа или выводом его на предварительный просмотр. Поддерживается только Internet Explorer начиная с 5.0
`onbeforeunload`	Наступает непосредственно перед выгрузкой текущего документа, например, при переходе на другую страницу или закрытии окна. Поддерживается только Internet Explorer начиная с 4.0
`onblur`	Наступает, когда окно перестает быть активным
`onerror`	Наступает, когда при загрузке документа происходит ошибка
`onfocus`	Наступает, когда окно становится активным
`onload`	Наступает сразу по окончании загрузки документа
`onresize`	Наступает, когда размеры окна изменяются
`onunload`	Наступает непосредственно перед выгрузкой текущего документа, например, при переходе на другую страницу или закрытии окна. В Internet Explorer наступает после `onbeforeunload`

Как видите, `window` поддерживает почти те же события, что и `body` (тело документа).

Хотите знать больше?

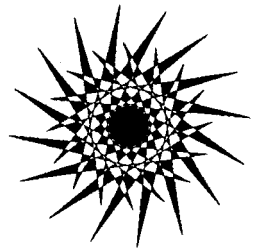
Воистину, всюду жизнь... Даже Web-страницы — и те реагируют на действия пользователя. Что-то там ползает за курсором, что-то меняет свой стиль, что-то дергается, мелькают какие-то картинки на гиперссылке, даже в глазах рябит. Неужели ради этого и создавались скриптовые языки, а программисты ломали головы над новыми версиями Web-обозревателей?

Конечно, нет!

Это просто украшения, красивости, штучки-дрючки, бантики сбоку страницы. JavaScript и объектная модель документа способны на большее. Они могут по-настоящему взаимодействовать с пользователем, например, принимать от него и обрабатывать какие-то данные. И они это делают! Как? Очень просто!

В следующей главе мы рассмотрим Web-формы, работу с выделенным текстом, операции drag-n-drop и поддержку Буфера обмена Windows. Процессу познания JavaScript нет конца!

ГЛАВА 5



Взаимодействие с пользователем

А сейчас мы рассмотрим то, ради чего и затевалась вся эта кутерьма вокруг JavaScript и DOM. Мы научимся взаимодействовать с пользователем, получать от него и обрабатывать различные данные. Мы изучим операции ввода-вывода: одни из важнейших операций, изучаемых любым новичком в любом языке программирования.

Может показаться, что мы уже выполняли какие-то операции ввода-вывода в предыдущей главе, посвященной обработке событий. Но это не так. Здесь мы изучим полноценный ввод-вывод данных, тот самый, который выполняют обычные приложения Windows. Для этого существуют разнообразные возможности:

- Web-формы;
- диалоговые HTML-окна;
- работа с выделенным текстом;
- работа с Буфером обмена;
- операции drag-n-drop.

И начнем мы с Web-форм.

Web-формы

Мы научимся создавать *формы ввода* — аналог "обычных" диалоговых окон, куда пользователь вводит данные с помощью полей ввода, всплывающих меню, флажков и прочих хорошо знакомых вам элементов интерфейса Windows. Эти данные можно затем обработать на стороне клиента или переслать серверному приложению для каких-либо целей.

Что такое форма ввода? Ничего особенного, просто обычная Web-страница, на которой, кроме элементов управления, могут быть размещены: любой

текст, рисунки, таблицы и прочие стандартные HTML-элементы. Сами поля ввода, всплывающие меню, флажки и т. д. и т. п. вставляются с помощью специальных HTML-тегов, которые мы здесь изучим.

Форма

Но первым делом поговорим об объекте формы, образующим саму форму ввода. Все элементы управления, ее составляющие, находятся внутри тега <FORM> и являются дочерними по отношению к нему. Собственно форма и определяет, что делать с данными дальше.

Что делать с данными дальше?

Данные, введенные пользователем в форму, можно использовать двумя способами, о которых мы уже говорили. Во-первых, их можно обработать на стороне клиента, с помощью JavaScript-сценария, и выдать пользователю в обработанном виде. Например, можно спросить у пользователя имя и потом поприветствовать. Или, что более полезно, спросить дату рождения и выдать знак зодиака, под которым пользователя угораздило родиться. В общем, применений много; я не сомневаюсь, что вы найдете и другие способы использовать данные пользователя на стороне клиента.

Во-вторых, данные можно передать на Web-сервер, в серверную программу, которая может, в свою очередь, сгенерировать Web-страницу специально для этого пользователя, занести переданные данные в базу, предложить купить какой-либо товар и многое другое. Форма сама передаст данные серверу. Разработчику Web-страницы нужно будет только указать способ передачи. А таких способов два, и они довольно сильно отличаются друг от друга.

Но прежде чем начать говорить об их отличиях, расскажем, в чем они схожи.

Для того чтобы получить какую-либо Web-страницу или иной файл, скажем, рисунок или ZIP-архив, Web-обозреватель передает Web-серверу HTTP-запрос, в котором указывает интернет-адрес этого файла. Конечно, кроме адреса, в HTTP-запросе передается и другая информация: сведения о программе Web-обозревателя, некоторые настройки операционной системы пользователя и др. Если Web-обозревателю нужно передать серверу какие-то введенные пользователем в форму данные, он также передает их в HTTP-запросе одним из двух методов.

Первый из них называется `get`. Данные, введенные пользователем, передаются серверу в виде неотъемлемой части интернет-адреса. В главе 3, посвященной взаимодействию HTML и JavaScript, мы рассмотрели объект `location` и шесть составных частей интернет-адреса. Помните?

```
http://www.supershop.ru/catalogs/search.exe?good=tapochkifortarakans
```

Английский с рязанширским акцентом... Так вот, ?good=tapochkifortarakans (то, что я подчеркнул) — это и есть введенные пользователем данные. Они всегда находятся в самом конце интернет-адреса и отделяются от остальных частей вопросительным знаком.

Вы также можете заметить, что пересылаемые данные имеют формат

```
{Имя параметра}={Значение параметра}.
```

Часто таких выражений больше, чем одно. В этом случае они отделяются друг от друга амперсандом ("коммерческим И") "&".

```
http://www.veniki_shop.ru/goods/buy.exe?cat=1&good=56
```

Метод `get` имеет два преимущества и один огромный недостаток. Преимущества: серверу передается минимальный объем данных, которые очень просто обработать (скажем, с помощью регулярного выражения). Недостаток: поскольку интернет-адрес не может быть длиннее 256 символов, вы не сможете переслать таким образом сколь-нибудь значительный объем данных. Метод `get` идеален, например, для поисковых систем, поскольку ключевые слова обычно имеют небольшой размер, для интернет-магазинов, товары в которых, как правило, пронумерованы, и вообще везде, где объем данных невелик. В противном случае вам следует использовать второй метод — `post`.

Метод `post` отличается от метода `get` тем, что данные передаются отдельным HTTP-запросом в том же самом формате

```
{Имя параметра}={Значение параметра}.
```

Само собой, на длину запроса здесь нет 256-символьного ограничения, и можно переслать на сервер какой угодно объем данных. В частности, некоторые Web-сайты позволяют пользователям загружать на сервер целые файлы, иной раз довольно "увесистые". Конечно, реализуется это методом `post`. Однако при пересылке этим методом передается значительно больший объем служебных данных, и от программиста, пишущего серверную программу, требуются дополнительные усилия по их обработке.

Метод `post` идеален для больших регистрационных форм, досок объявлений, вообще для пересылки больших текстов и файлов на сервер.

Тег *<FORM>...</FORM>*

А теперь пора приняться за собственно форму.

Web-форма ввода данных создается с помощью парного тега `<FORM>`. Внутри этого тега размещаются теги, вставляющие в форму элементы управления.

```
<FORM ACTION="{Адрес серверной программы}" METHOD="get|post"
[ENCTYPE="{Тип MIME передаваемых данных}"] [DISABLED]
```

```
[TARGET="{Имя фрейма}|_self|_parent|_top|_blank"]>
. . . Теги элементов управления
</FORM>
```

Здесь перечислены только атрибуты, специфичные для тега <FORM>. Кроме этого, он также поддерживает общие атрибуты ID, NAME, CLASS и STYLE.

Давайте перечислим по порядку все атрибуты, поддерживаемые этим тегом.

Атрибут ACTION позволяет указать адрес серверной программы, обрабатывающей переданные формой данные. А атрибут METHOD, как вы уже поняли, указывает метод передачи данных.

```
<FORM ACTION="http://www.veniki_shop.ru/goods/buy.exe" METHOD="get">
```

Атрибут ENCTYPE задает MIME-тип передаваемых данных. Если он отсутствует, то используется значение по умолчанию "application/x-www-form-urlencoded".

Атрибут DISABLED задает изначальную недоступность формы. Вы можете впоследствии сделать ее доступной из скрипта. Этот атрибут поддерживается для формы только Internet Explorer начиная с версии 5.5.

Атрибут TARGET знаком вам по тегу гиперссылки. Он указывает, куда будет выводиться информация, переданная в ответ серверной программой. Вы можете задать либо имя фрейма, либо одно из предопределенных значений, перечисленных в табл. 1.7.

Помимо элементов управления, форма может содержать любые другие элементы HTML. Например, это могут быть таблицы, служащие для оформления, рисунки, текст с различным форматированием. Как вы увидите позже, без этого трудно обойтись при создании любой достаточно сложной формы.

Элементы управления

После тега формы можно начать изучение тегов, образующих элементы управления.

Тег *<INPUT>*

С помощью одинарного тега <INPUT> в форму вставляется большинство элементов управления, знакомых нам по интерфейсу Windows: поля ввода, кнопки, флажки и т. п. Формат этого тега приведен ниже:

```
<INPUT TYPE="button|checkbox|file|hidden|image|password|radio|reset|
↳submit|text"
[ALIGN="left|right|top|texttop|middle|absmiddle|baseline|bottom|
↳absbottom"]
[CHECKED] [MAXLENGTH="{Максимальное количество символов}"]
[SIZE="{Размер поля ввода}"] [SRC="{Адрес файла рисунка}"]
```

```
[VALUE="{Начальное значение}"] [TABINDEX="{Порядок обхода}"]
[DISABLED] [READONLY]>
```

Здесь перечислены только атрибуты, специфичные для тега <INPUT>. Кроме этого, он также поддерживает общие атрибуты ID, NAME, CLASS и STYLE.

Тип элемента управления задается значением атрибута TYPE. В табл. 5.1 приведены все возможные значения и их описания.

Таблица 5.1. *Значения атрибута* TYPE *тега* <INPUT>

Значение	Описание
button	Обычная командная кнопка
checkbox	Флажок
file	Поле ввода имени файла и кнопка **Открыть**. Позволяет отправить файл на Web-сервер.
hidden	"Скрытое" поле. Оно никак не отображается на форме, но его значение посылается вместе с остальными данными. Может использоваться, например, для уникальной идентификации Web-страницы
image	Изображение. Аналогично по действию кнопке submit
password	Текстовое поле для ввода паролей. Вводимый текст отображается в виде звездочек
radio	Радиокнопка (кнопка-переключатель)
reset	Командная кнопка, аналогичная button, при нажатии на которую вся форма очищается. Называется также кнопкой reset
submit	Командная кнопка, аналогичная button, при нажатии на которую происходит отправка данных, введенных в форму. Называется также кнопкой submit
text	Поле ввода

Назначение остальных атрибутов тега <INPUT> сильно зависит от типа элемента управления. В табл. 5.2 указано, за что отвечает тот или иной атрибут в зависимости от значения атрибута TYPE. Более подробное описание будет дано после таблицы.

Таблица 5.2. *Описание атрибутов тега* <INPUT>

Атрибут	Значение атрибута TYPE	Описание
ALIGN	image	Вертикальное выравнивание рисунка. Ведет себя так же, как аналогичный атрибут тега <IMG> (см. главу 1)

Таблица 5.2 (окончание)

Атрибут	Значение атрибута TYPE	Описание
CHECKED	checkbox, radio	Флажок или радиокнопка включены по умолчанию
DISABLED	Все	Элемент управления изначально запрещен. Поддерживается только Internet Explorer начиная с 4.0
MAXLENGTH	password, text	Максимальное количество символов, которое может быть введено в текстовое поле или область редактирования. Может быть больше, чем SIZE; в таком случае текст будет прокручиваться
NAME|ID	radio	Имя группы радиокнопок
	Остальные	Имя элемента управления
READONLY	Все	Элемент управления доступен только для чтения
SIZE	password, text	Видимый размер поля ввода или области редактирования
TABINDEX	Все	Порядок обхода элементов управления при последовательном нажатии клавиши <Tab>. Поддерживается только Internet Explorer начиная с 4.0
VALUE	button	Текст, отображаемый на кнопке
	checkbox, hidden, radio	Значение, посылаемое вместе с основными данными
	reset	Текст, отображаемый на кнопке. Если не указан, отображается **Сброс** или что-то подобное
	submit	Текст, отображаемый на кнопке. Если не указан, отображается **Отправить** или что-то подобное
	file, password, text	Значение по умолчанию

А теперь рассмотрим создание форм более подробно.

Каждая форма должна содержать кнопку submit, создаваемую атрибутом TYPE="submit". При нажатии на эту кнопку происходит отправка всех введенных в форму данных серверной программе, адрес которой указан в атрибуте ACTION тега <FORM>. Также форма может содержать кнопку reset, создаваемую атрибутом TYPE="reset"; при нажатии на нее форма очищается, т. е. все элементы управления формы получают значения по умолчанию.

Я уже говорил, что пересылаемые данные имеют формат

```
{Имя параметра}={Значение параметра}.
```

Именем параметра является имя элемента управления. Например,

```
<INPUT TYPE="text" ID="username1">
<INPUT TYPE="text" ID="username2">
```

Из формы с такими полями ввода серверная программа получит следующие данные:

```
username1=Вася
username2=Пупкин
```

С помощью атрибута VALUE можно задать для текстовых полей значение по умолчанию:

```
<INPUT TYPE="text" ID="username1" VALUE="Введите свое имя">
<INPUT TYPE="text" ID="username2" VALUE="Введите свою фамилию">
```

Атрибуты SIZE и MAXLENGTH помогут вам задать размер поля ввода и максимальное количество символов, которое можно в него ввести:

```
<INPUT TYPE="text" ID="address" SIZE="40" MAXLENGTH="256"
VALUE="Введите адрес вашего сайта">
```

В данном случае мы задали размер поля ввода меньше, чем количество допустимых символов. Текст в нашем поле ввода будет прокручиваться.

Поле ввода пароля password отличается от обычного только тем, что вводимый текст отображается звездочками.

```
<INPUT TYPE="text" ID="username">
<INPUT TYPE="password" ID="userpassword">
```

С флажками и группами радиокнопок дело обстоит чуть сложнее.

```
<INPUT TYPE="checkbox" ID="adult" VALUE="yes">Вам уже исполнилось 18?
```

Здесь мы создали флажок, который во включенном состоянии отправит следующие данные:

```
adult=yes
```

В противном случае он не отправит ничего. И серверная программа должна предусмотреть такой случай.

Вы можете опустить атрибут VALUE. Тогда будет использовано значение по умолчанию "on".

```
<INPUT TYPE="checkbox" ID="adult">Вам уже исполнилось 18?
adult=on
```

Радиокнопки всегда встречаются группами — а иначе их не имеет смысла использовать — и в обычных Windows-программах, и в Web-формах. Вся группа должна иметь одно и то же значение атрибута NAME:

```
Укажите ваш пол
<INPUT TYPE="radio" ID="gender" VALUE="male">Мужской
<INPUT TYPE="radio" ID="gender" VALUE="female">Женский
```

Если выбрана первая радиокнопка, будет послана такая строка:

```
gender=male
```

Если вторая — то

```
gender=female
```

Если же ни одна из радиокнопок не выбрана, никаких данных послано не будет. Как и в случае с флажком.

Заметьте, что текст надписи для флажка и радиокнопки мы указывали отдельно.

Атрибут CHECKED позволяет указать, что флажок или одна из радиокнопок в группе включена по умолчанию.

```
<INPUT TYPE="checkbox" ID="adult" CHECKED>Вам уже исполнилось 18?
<INPUT TYPE="radio" ID="gender" VALUE="male" CHECKED>Мужской
<INPUT TYPE="radio" ID="gender" VALUE="female">Женский
```

Также вы можете вставить в форму обычную командную кнопку, которая может выполнять какое-либо действие, отличное от отправки данных и очистки формы. (Естественно, такие кнопки имеет смысл использовать только с привязанным к ним скриптом, что мы рассмотрим чуть позже.)

```
<INPUT TYPE="button" ID="recalcdata" VALUE="Пересчитать">
```

Здесь значение атрибута VALUE, в отличие от флажков и радиокнопок, задает подпись на кнопке.

Для кнопок submit и reset атрибут VALUE не является обязательным. По умолчанию на них отображается стандартный текст, различающийся у разных программ Web-обозревателей. (В частности, у последних версий Internet Explorer отображается соответственно **Отправить** и **Сброс**.) Но вы можете задать другой текст подписи, указав его в атрибуте VALUE, как и для обычной кнопки.

Кстати, кнопки submit и reset тоже посылают свою строку данных. Она имеет вид:

```
{Значение атрибута NAME|ID}={Значение атрибута VALUE|ID}
```

Но если один из этих атрибутов не задан, ничего не посылается.

Вместо кнопки submit вы можете использовать рисунок image:

```
<INPUT TYPE="image" ID="okimage" SRC="ok.gif">
```

В этом случае посылаются две строки, имеющие следующий формат:

```
{Значение атрибута NAME|ID}.x={Координата X}
{Значение атрибута NAME|ID}.y={Координата Y}
```

Обе координаты отсчитываются от левого верхнего угла рисунка.

"Скрытое" поле hidden может использоваться для внутренних целей серверной программы. Например, если программа генерирует для каждого пользователя Web-сайта отдельную персональную страницу с формой, то в "скрытое" поле она может включить уникальное имя пользователя.

```
<INPUT TYPE="hidden" ID="hiddenusername" VALUE="vasya_pupkin">
```

И впоследствии, когда пользователь отправит введенные данные обратно серверной программе, она будет знать, от кого пришли эти данные.

Элемент file позволяет отправить серверной программе любой файл и представляет собой поле ввода, где вводится имя файла, и кнопку **Открыть**, выводящую на экран стандартный диалог открытия файла Windows. При этом значение атрибута METHOD тега <FORM> должно быть равно "post", а значение атрибута ENCTYPE того же тега — "multipart/form-data".

Тег <TEXTAREA>...</TEXTAREA>

Парный тег <TEXTAREA> размещает в форме область редактирования текста. Он имеет следующий формат:

```
<TEXTAREA [ROWS="{Количество строк}"] [COLS="{Количество столбцов}"]
[WRAP="off|soft|hard"] [TABINDEX="{Порядок обхода}"] [DISABLED]
[READONLY]>
. . . Начальное значение
</TEXTAREA>
```

Здесь перечислены только атрибуты, специфичные для тега <TEXTAREA>. Кроме этого, он поддерживает общие атрибуты ID, NAME, CLASS и STYLE.

Атрибуты ROWS и COLS задают количество строк и столбцов текста соответственно в области редактирования. Если содержимое не вмещается в заданные размеры, появляются полосы прокрутки. Имейте в виду, что при отображении текста в области редактирования используется моноширинный шрифт.

Атрибут WRAP задает, как будут вести себя слишком длинные строки, не помещающиеся по ширине в область редактирования. Доступны три значения: "off" (значение по умолчанию) ничего с ними не делает, "soft" просто "разрывает" их, располагая текст в несколько строк, а "hard" не только

"разрывает", но и вставляет в местах "разрыва" символы возврата каретки и перевода строки. То есть, два первых значения ничего не делают с введенным текстом; длинные строки пересылаются "как есть", а третье позволяет разбить их на более короткие.

Атрибуты TABINDEX, DISABLED и READONLY вам уже знакомы; я не буду описывать их повторно.

Скажем, вы можете организовать на своей Web-странице область редактирования для ввода сведений о пользователе:

```
<TEXTAREA COLS="60" ROWS="20" WRAP="soft">
Введите любой текст.
</TEXTAREA>
```

Теги <SELECT>...</SELECT> и <OPTION>

Парный тег <SELECT> позволяет создать на Web-странице список или всплывающее меню, из которых пользователь может выбрать нужный пункт. Каждый из доступных пунктов задается тегом <OPTION>.

Сначала рассмотрим тег <SELECT>. Он имеет следующий формат:

```
<SELECT SIZE="{Количество одновременно отображаемых пунктов}"
[MULTIPLE] [TABINDEX="{Порядок обхода}"] [DISABLED]>
. . . Теги отдельных пунктов
</SELECT>
```

Кроме этого, тег <SELECT> поддерживает общие атрибуты ID, NAME, CLASS и STYLE.

Атрибут SIZE позволяет задать, сколько пунктов списка будет показано одновременно. Если задан один пункт, отображается всплывающее меню, если больше — список.

Атрибуты TABINDEX и DISABLED вам уже знакомы.

Внутри тега <SELECT> размещаются теги <OPTION>, представляющие отдельные пункты списка. Тег <OPTION> имеет следующий формат:

```
<OPTION [VALUE="{Значение пункта}"] [SELECTED]>
. . . Текст данного пункта
```

Тег <OPTION> поддерживает, кроме того, общие атрибуты ID и CLASS.

Атрибут VALUE представляет значение, которое будет послано серверной программе, если был выбран данный пункт списка. Если атрибут VALUE опущен, посылается текст этого пункта.

Давайте рассмотрим пример, который пояснит сказанное. Создадим для этого следующий список:

```
<SELECT ID="fruit" SIZE="3">
<OPTION>Апельсин
<OPTION>Мандарин
<OPTION VALUE="lemon">Лимон
</SELECT>
```

Если пользователь выберет первый или второй пункт списка, форма пошлет такие данные:

```
fruit=Апельсин
```

или

```
fruit=Мандарин
```

(Разумеется, они будут посланы в закодированном виде, т. к. протокол HTTP органически не переносит русских букв. Но давайте не обращать внимания на эту его "русофобию".)

А если пользователь выберет третий пункт, будет послано следующее:

```
fruit=lemon
```

Атрибут SELECTED позволяет указать, какой пункт списка будет выбран изначально.

Теги оформления Internet Explorer

Internet Explorer предлагает также несколько тегов, позволяющих немного украсить Web-формы. Это текстовые метки и группирующие рамки, аналогичные стандартным меткам и рамкам Windows.

Тег <LABEL>...</LABEL>

Парный тег <LABEL> позволяет привязать к элементу формы текстовое описание. Пользователь может активизировать элемент, просто щелкнув по этому описанию. Также пользователь может использовать клавишу-ускоритель, которую следует нажимать одновременно с клавишей <Alt>.

Тег <LABEL> имеет следующий формат:

```
<LABEL FOR="{Имя элемента управления, к которому привязана метка}"
[ACCESSKEY="{Клавиша-ускоритель}"] [DISABLED]>
. . . Текст метки
</LABEL>
```

Тег метки также поддерживает общие атрибуты CLASS, ID и STYLE.

Атрибут FOR задает имя элемента формы, к которому привязана метка. Имейте в виду, что это имя в теге элемента должно быть указано при помощи атрибута ID, а не NAME!

Атрибут ACCESSKEY задает клавишу-ускоритель для доступа к элементу формы. Например, если предполагается использовать клавишу <A>, то атрибут

будет иметь вид ACCESSKEY="A". Не забывайте, что клавишу-ускоритель нужно нажимать одновременно с клавишей <Alt>.

С атрибутом DISABLED вы уже знакомы. Единственное отличие: в этом случае он разрешает или запрещает доступ и к метке, и к элементу формы, с которым метка связана.

Рассмотрим небольшой пример. Пусть мы имеем текстовое поле, к которому нужно привязать метку:

```
<LABEL FOR="txtName" ACCESSKEY="И">Имя пользователя</LABEL>
<INPUT TYPE="text" ID="txtName" SIZE="40">
```

Теги <FIELDSET>...</FIELDSET> и <LEGEND>...</LEGEND>

Парный тег <FIELDSET> позволяет собрать выбранные элементы формы в группу, ограниченную группирующей рамкой. С помощью парного тега <LEGEND> можно задать заголовок, который будет отображен в верхней части этой рамки.

Рассмотрим сначала тег <FIELDSET>. Он имеет следующий формат:

```
<FIELDSET [DISABLED]>
. . . Элементы управления, помещаемые в группу
</FIELDSET>
```

Тег группы также поддерживает общие атрибуты CLASS, ID и STYLE.

Атрибут DISABLED вам уже знаком. Здесь он отвечает за разрешение или запрещение доступа ко всей группе.

Формат тега <LEGEND> совсем прост:

```
<LEGEND>Текст заголовка группы</LEGEND>
```

И, конечно же, поддерживаются общие атрибуты CLASS, ID и STYLE.

Небольшой пример формы

В заключение рассмотрим небольшой пример формы, который вы можете использовать как шаблон для своих форм. Это будет форма регистрации в некой базе данных, которых сейчас хватает во Всемирной паутине.

```
<HTML>
<HEAD>
<TITLE>Регистрация</TITLE>
</HEAD>
<BODY>
<H1>Регистрация в базе данных</H1>
<FORM ACTION="dummy.htm" METHOD="get">
```

Здесь начинается наша форма. В качестве места назначения и метода передачи данных пользователя можно, в принципе, указать все что угодно: мы ничего отправлять не будем.

```
<TABLE>
```

Применим таблицу для красивого оформления формы.

```
<TR><TD>
<LABEL FOR="txtName">Имя пользователя</LABEL>
<INPUT TYPE="text" ID="txtName" MAXLENGTH="40" SIZE="40">
</TD></TR>
```

Используем специфический для Internet Explorer тег <LABEL>. Navigator все равно его проигнорирует.

В дальнейшем, если вы будете писать скрипт, который должен работать и в Internet Explorer, и в Navigator, используйте для задания имени атрибут NAME, а не ID, потому что Navigator не поддерживает ID. Здесь же мы использовали атрибут ID только для того, чтобы привязать к полю ввода имени пользователя метку, которую Navigator все равно не поддерживает. Возможно, вам даже придется задавать имя элемента формы дважды: атрибутами ID и NAME.

```
<TR><TD>
<LABEL FOR="txtPassword">Пароль</LABEL>
<INPUT TYPE="password" ID="txtPassword" MAXLENGTH="40" SIZE="40">
</TD></TR>
```

Для ввода пароля используем специальное поле ввода пароля.

```
<TR><TD>
<FIELDSET>
```

Элементы формы, в которые вводится дата рождения пользователя, оформим в виде группы.

```
<LEGEND>Дата рождения</LEGEND>
```

Не забываем указать заголовок группы.

```
<TABLE WIDTH="100%">
```

Внутренняя таблица. С ее помощью мы располагаем на странице элементы управления, входящие в группу.

```
<TR>
<TD WIDTH="20%">
<LABEL FOR="txtDay">Число</LABEL>
<INPUT TYPE="text" ID="txtDay" MAXLENGTH="2" SIZE="2">
</TD>
<TD WIDTH="60%">
```

```
<LABEL FOR="cboMonth">Месяц</LABEL>
<SELECT ID="cboMonth" SIZE="1">
<OPTION>Январь
<OPTION>Февраль
<OPTION>Март
<OPTION>Апрель
<OPTION>Май
<OPTION>Июнь
<OPTION>Июль
<OPTION>Август
<OPTION>Сентябрь
<OPTION>Октябрь
<OPTION>Ноябрь
<OPTION>Декабрь
<SELECT>
</TD>
```

Код, реализующий всплывающее меню (или список), очень велик.

```
<TD WIDTH="20%">
<LABEL FOR="txtYear">Год</LABEL>
<INPUT TYPE="text" ID="txtYear" MAXLENGTH="4" SIZE="4">
</TD>
</TR>
</TABLE>
```

Закрываем внутреннюю таблицу.

```
</TD></TR>
</TABLE>
```

Закрываем внешнюю таблицу.

```
</FORM>
```

Закрываем форму.

```
</BODY>
</HTML>
```

Код этой страницы довольно велик, и это притом, что она не содержит никаких скриптов. Писать скрипты, работающие с формой и ее элементами, мы научимся чуть позднее. А пока сохраните этот код в файле 5.1.htm и откройте в любом Web-обозревателе. То, что вы увидите в Internet Explorer, показано на рис. 5.1, в Navigator — на рис. 5.2.

Обе популярнейшие программы Web-обозревателей отображают нашу форму правильно. Navigator просто проигнорировал неизвестные ему теги и отобразил только текст подписей к элементам формы.

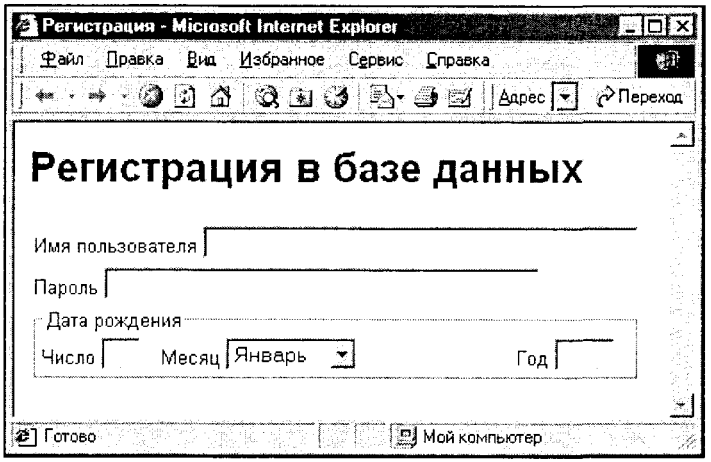


Рис. 5.1. Web-форма (Internet Explorer)

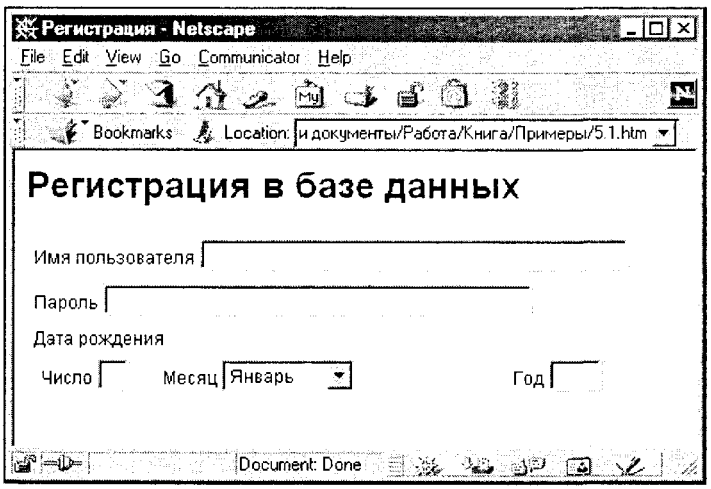


Рис. 5.2. Web-форма (Navigator)

Написание скриптов, работающих с формами

А теперь пора узнать, как можно использовать формы и их элементы в JavaScript-сценариях.

Объект формы

Все формы текущего документа доступны через коллекцию `forms`. Она поддерживается обеими популярнейшими программами Web-обозревателей.

```
document.forms("frmSubmit")
```

К форме можно обращаться и по имени, как и к любому элементу Web-страницы.

```
document.frmSubmit
```

Повторю еще раз. Если вы пишете исключительно для Internet Explorer, то можете задавать имена формы и всех ее элементов в атрибуте ID. Navigator же требует задания имени в атрибуте NAME. Поэтому, если вы хотите, чтобы ваша страница работала в обеих программах, используйте атрибут NAME.

Объект формы поддерживает ряд свойств, методов и событий. Ниже мы их все перечислим.

Все свойства, поддерживаемые объектом формы, описаны в табл. 5.3.

Таблица 5.3. *Свойства объекта формы*

Свойство	Описание
action	Аналогично атрибуту ACTION тега <FORM>
disabled	Аналогично атрибуту DISABLED тега <FORM>. Возвращает значение true или false. Поддерживается только Internet Explorer начиная с 5.5
elements	Возвращает ссылку на коллекцию elements, описываемую ниже
encoding	Аналогично атрибуту ENCTYPE тега <FORM>
method	Аналогично атрибуту METHOD тега <FORM>
name	Аналогично атрибуту NAME тега <FORM>
target	Аналогично атрибуту TARGET тега <FORM>

Объект формы поддерживает и все общие свойства.

Объект формы также поддерживает два метода. Первый из них — submit() — позволяет выполнить отправку данных серверной программе и аналогичен нажатию кнопки submit. Второй — reset() — позволяет очистить форму и аналогичен нажатию кнопки reset.

Событий, поддерживаемых объектом формы, также два. Это onsubmit и onreset, наступающие при посылке данных и очистке формы соответственно. Вы можете отменить соответствующее действие, вернув из функции-обработчика события false.

Элементы формы

Все элементы текущей формы доступны через коллекцию elements:

```
document.frmSubmit.elements("txtName")
document.frmSubmit.txtName
```

Заметьте, что обязательно нужно указать ссылку на саму форму. А иначе Web-обозреватель будет искать указанный вами элемент управления в теле документа, игнорируя все формы и, в конце концов, не найдя его, вернет null.

Свойства, поддерживаемые объектами элементов формы, перечислены в табл. 5.4, методы — в табл. 5.5, события — в табл. 5.6. Кроме того, поддерживаются все общие свойства, методы и события.

Таблица 5.4. *Свойства элементов формы*

Свойство	Объект	Описание
align	image	Аналогично атрибуту ALIGN тега <INPUT>
checked	checkbox, radio	Возвращает true, если флажок или радиокнопка находится во включенном состоянии
defaultChecked	checkbox, radio	Аналогично атрибуту CHECKED тега <INPUT>. Возвращает значение true или false
defaultValue	file, password, text	Возвращает начальное значение, заданное атрибутом VALUE тега <INPUT>
disabled	Все	Аналогично атрибуту DISABLED тега <INPUT>. Возвращает значение true или false. Поддерживается только Internet Explorer начиная с 4.0
form	Все	Возвращает ссылку на форму, в которой находится данный элемент управления
indeterminate	checkbox	Флажок находится в неопределенном состоянии (закрашен серым). Возвращает значение true или false. Поддерживается только Internet Explorer начиная с 4.0
maxLength	password, text	Аналогично атрибуту MAXLENGTH тега <INPUT>. Поддерживается только Internet Explorer начиная с 4.0
name	Все	Аналогично атрибуту NAME тега <INPUT>
readOnly	password, text	Аналогично атрибуту READONLY тега <INPUT>. Возвращает значение true или false. Поддерживается только Internet Explorer начиная с 4.0
tabIndex	Все	Аналогично атрибуту TABINDEX тега <INPUT>. Поддерживается только Internet Explorer начиная с 4.0
type	Все	Аналогично атрибуту TYPE тега <INPUT>

Таблица 5.4 (окончание)

Свойство	Объект	Описание
value	Все	Значение текущего элемента формы. То же самое, что и данные, которые он отправит серверной программе

Таблица 5.5. Методы элементов формы

Метод	Объект	Описание
blur()	Все	Убирает фокус ввода с текущего элемента формы
focus()	Все	Помещает фокус вода на текущий элемент формы
select()	password, text	Выделяет текст в текстовом поле

Таблица 5.6. События элементов формы

Событие	Объект	Описание
onblur	Все	Наступает, когда текущий элемент формы теряет фокус ввода
onchange	password, text	Наступает после того, как пользователь изменил данные в текстовом поле и либо переместил фокус ввода на другой элемент формы, либо отправил данные формы серверной программе. Наступает перед событием onblur
onclick	button, checkbox, radio, reset, submit	Наступает, когда пользователь щелкает по кнопке. Вообще-то, это общее событие, поддерживаемое всеми элементами страницы в Internet Explorer и гиперссылками в Navigator. Я упомянул его здесь потому, что это одно из самых часто перехватываемых событий
onfocus	Все	Наступает, когда текущий элемент формы получает фокус ввода

Область редактирования текста поддерживает свой набор свойств, описанных в табл. 5.7. Методы и события поддерживаются в основном те же, что и рассмотренные ранее (см. табл. 5.5 и 5.6). Разумеется, поддерживаются также все общие свойства, методы и события.

Таблица 5.7. *Свойства объекта области редактирования*

Свойство	Описание
cols	Аналогично атрибуту COLS тега <TEXTAREA>. Поддерживается только Internet Explorer начиная с 4.0
defaultValue	Возвращает начальное значение, помещенное в область редактирования
disabled	Аналогично атрибуту DISABLED тега <TEXTAREA>. Возвращает значение true или false. Поддерживается только Internet Explorer начиная с 4.0
form	Возвращает ссылку на форму, в которой находится данный элемент управления
name	Аналогично атрибуту NAME тега <TEXTAREA>
readOnly	Аналогично атрибуту READONLY тега <TEXTAREA>. Возвращает значение true или false. Поддерживается только Internet Explorer начиная с 4.0
rows	Аналогично атрибуту ROWS тега <TEXTAREA>. Поддерживается только Internet Explorer начиная с 4.0
tabIndex	Аналогично атрибуту TABINDEX тега <TEXTAREA>. Поддерживается только Internet Explorer начиная с 4.0
type	Возвращает тип элемента формы. Для области редактирования всегда равно "textarea"
value	Значение текущей области редактирования. То же самое, что и данные, которые она отправит серверной программе
wrap	Аналогично атрибуту WRAP тега <TEXTAREA>. Поддерживается только Internet Explorer начиная с 4.0

Свой набор свойств, описанных в табл. 5.8, поддерживает объект списка. Наборы методов и событий не отличаются от рассмотренных в табл. 5.6 и 5.7 соответственно. Поддерживаются также все общие свойства, методы и события.

Таблица 5.8. *Свойства объекта списка*

Свойство	Описание
disabled	Аналогично атрибуту DISABLED тега <SELECT>. Возвращает значение true или false. Поддерживается только Internet Explorer начиная с 4.0
form	Возвращает ссылку на форму, в которой находится данный список

***Таблица 5.8** (окончание)*

Свойство	Описание
length	Возвращает количество пунктов, имеющихся в данном списке
multiple	Аналогично атрибуту MULTIPLE тега <SELECT>. Возвращает значение true или false. Поддерживается только Internet Explorer начиная с 4.0
name	Аналогично атрибуту NAME тега <TEXTAREA>
options	Возвращает ссылку на коллекцию пунктов, имеющихся в данном списке
selectedIndex	Номер выбранного пользователем пункта. (Нумерация пунктов начинается с нуля.)
size	Аналогично атрибуту SIZE тега <SELECT>. Поддерживается только Internet Explorer начиная с 4.0
tabIndex	Аналогично атрибуту TABINDEX тега <SELECT>. Поддерживается только Internet Explorer начиная с 4.0
type	Возвращает тип элемента формы. Для списка равно "select-multiple", если в теге <SELECT> присутствует атрибут MULTIPLE, или "select-one" — в противном случае
value	Значение пункта, выбранного в списке. То же самое, что и данные, которые он отправит серверной программе

Объект пункта списка имеет совсем небольшой набор специфических свойств, описанных в табл. 5.9. Дополнительных методов и событий у него нет. И опять же, поддерживаются общие свойства, методы и события.

***Таблица 5.9.** Свойства объекта пункта списка*

Свойство	Описание
defaultSelected	Аналогично атрибуту SELECTED тега <OPTION>. Возвращает значение true или false
index	Позиция пункта в списке
selected	Возвращает true, если пункт выбран в списке
text	Текст пункта списка
value	Аналогично атрибуту VALUE тега <OPTION>. Если этот атрибут не задан, возвращается пустая строка

Дополнительные элементы формы, предлагаемые Internet Explorer, также поддерживают свой набор свойств, но мы не будем его рассматривать. Они редко используются в скриптах, т. к. предназначены в основном для украшения. Полное описание всех элементов форм приведено в приложении 1, а список поддерживаемых ими свойств, методов и событий — в приложении 3.

Простейшая форма со скриптом

Рассмотрим простейший пример использования скриптов с формами и элементами управления. Пусть это будет форма, спрашивающее имя пользователя и выводящая на экран приветствие.

```
<HTML>
<HEAD>
<TITLE>Приветствующая форма</TITLE>
<SCRIPT>
function showMe() {
```

Это функция, выводящая приветствие.

```
window.alert("Приветствую тебя, " + document.frm.txtName1.value + " " +
↳document.frm.txtName2.value + "!"); }
```

Обратите внимание, как мы получаем доступ к элементам формы. Форма — это своего рода контейнер, где размещаются элементы управления, и скрипт может получить к ним доступ, только сначала указав ссылку на саму форму. Запомните это.

```
</SCRIPT>
</HEAD>
<BODY>
<FORM ACTION="dummy.htm" METHOD="get" NAME="frm"
↳onsubmit="showMe(); return false;">
```

Задаем имя формы через атрибут `NAME` для совместимости с Navigator. А чтобы отменить посылку данных, вернем из обработчика события `onsubmit` `false`.

```
<TABLE>
<TR>
<TD COLSPAN="2">Введите свои данные...</TD>
</TR>
<TR>
<TD><LABEL FOR="txtName1">Имя</LABEL></TD>
<TD><INPUT TYPE="text" NAME="txtName1" ID="txtName1"></TD>
```

Здесь мы задаем имя поля ввода с атрибутом NAME (для совместимости с Navigator) и атрибутом ID (для привязки к нему текстовой метки).

```
</TR>
<TR>
<TD><LABEL FOR="txtName2">Фамилия</LABEL></TD>
<TD><INPUT TYPE="text" NAME="txtName2" ID="txtName2"></TD>
</TR>
<TR>
<TD><INPUT TYPE="submit" NAME="cmdOK" VALUE="OK"></TD>
<TD><INPUT TYPE="reset" NAME="cmdCancel" VALUE="Отмена"></TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```

Остальной код не должен вызвать у вас вопросов. Сохраните его под именем 5.2.htm и откройте в любой программе Web-обозревателя. Попробуйте ввести что-нибудь в поля ввода и нажать кнопку **OK**. На рис. 5.3 показано, что вы увидите в Internet Explorer. Navigator выдаст вам то же самое.

Конечно, пользы от этой формы немного. Но наш простейший пример поможет вам понять, как реализуется взаимодействие между формой, ее элементами, скриптом и HTML-кодом формы. А в следующем примере мы рассмотрим нечто более полезное, а именно проверку правильности ввода пользователя.

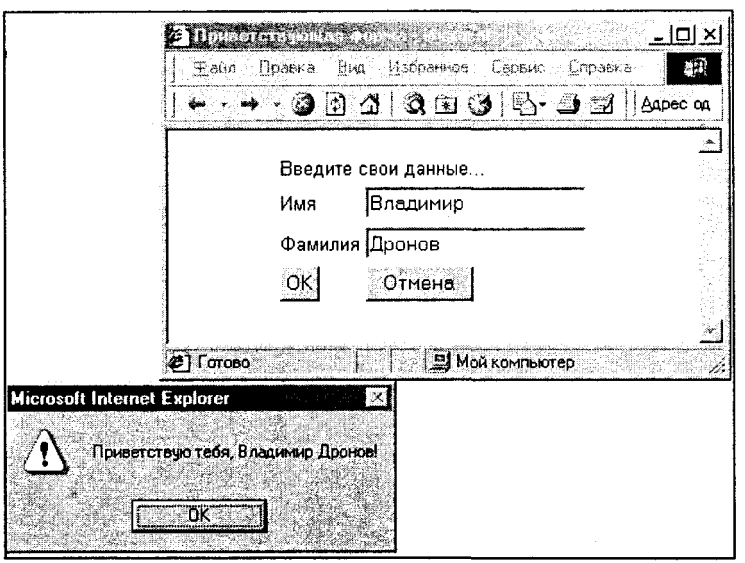


Рис. 5.3. Форма, приветствующая пользователя по имени

Пример формы с контролем правильности ввода данных

Давайте создадим форму, спрашивающую у пользователя его имя, фамилию и адрес электронной почты. Такие формы вы можете увидеть на многих Web-страницах в Интернете; они используются для регистрации в базах данных и подписки на рассылки. Но наша форма будет не просто запрашивать данные пользователя, но и проверять их корректность. В частности, правильность электронного почтового адреса будет контролироваться регулярным выражением.

```
<HTML>
<HEAD>
<TITLE>Форма регистрации с проверкой</TITLE>
<SCRIPT>
function submitForm () {
  var name1, name2, address, re, dataRight = true, message;
  re = /([0-9a-zA-Z\.-_]+)@([0-9a-zA-Z\.-_]+)/;
```

С помощью этого регулярного выражения мы проверим адрес электронной почты пользователя.

```
name1 = document.frm.txtName1.value;
name2 = document.frm.txtName2.value;
address = document.frm.txtAddress.value;
```

Помещаем значения элементов форм в локальные переменные.

```
if (name1.length == 0) {
  dataRight = false;
  message = "Введите имя."; }
```

Проверяем, введено ли имя.

```
else {
  if (name2.length == 0) {
    dataRight = false;
    message = "Введите фамилию."; }
```

Проверяем, введена ли фамилия.

```
else {
  if (address.match(re) == null) {
    dataRight = false;
    message = "Неверный или отсутствующий адрес e-mail."; } } }
```

Проверяем, введен ли адрес e-mail и правильно ли он введен. Если регулярное выражение не совпадет со значением поля адреса, метод `match` вернет `null`.

```
if (dataRight)
  window.alert("Данные верны и успешно отправлены.")
else
  window.alert(message); }
```

Отображаем соответствующее случаю предупреждение.

```
</SCRIPT>
</HEAD>
<BODY>
<FORM ACTION="dummy.htm" METHOD="get" NAME="frm"
↳onsubmit="submitForm(); return false;">
```

Независимо от результата проверки правильности ввода, в данном примере мы всегда отменяем действие по умолчанию, т. е. посылку данных формой, поскольку серверного приложения, готового принять и обработать эти данные, у нас все равно нет. Конечно, если вы будете использовать в дальнейшем серверные приложения и писать для них формы ввода, вам нужно будет в зависимости от результата проверки запрещать или размещать посылку данных, возвращая `false` или `true`.

```
<TABLE>
<TR>
<TD COLSPAN="2">Введите свои данные...</TD>
</TR>
<TR>
<TD><LABEL FOR="txtName1">Имя</LABEL></TD>
<TD><INPUT TYPE="text" NAME="txtName1" ID="txtName1"></TD>
</TR>
<TR>
<TD><LABEL FOR="txtName2">Фамилия</LABEL></TD>
<TD><INPUT TYPE="text" NAME="txtName2" ID="txtName2"></TD>
</TR>
<TR>
<TD><LABEL FOR="txtAddress">E-mail</LABEL></TD>
<TD><INPUT TYPE="text" NAME="txtAddress" ID="txtAddress"></TD>
</TR>
<TR>
<TD><INPUT TYPE="submit" NAME="cmdOK"></TD>
<TD><INPUT TYPE="reset" NAME="cmdCancel"></TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```

Остальной код почти такой же, как в предыдущем примере. Сохраните его в файле 5.3.htm и откройте в любом Web-обозревателе. Введите что-нибудь

в поля ввода, оставьте одно из них пустым и нажмите кнопку посылки данных. На рис. 5.4 показано, что получилось у меня.

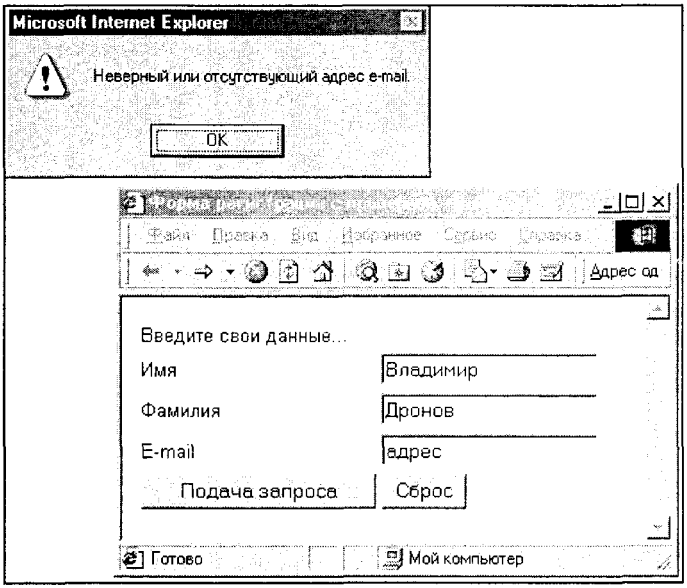


Рис. 5.4. Форма с проверкой правильности ввода данных

Вы можете подумать, что формы ввода и элементы управления используются только для посылки данных серверным программам. Отнюдь! Формы — да, но не элементы управления. Дело в том, что элементы управления вовсе не обязательно должны находиться внутри формы. Вы можете расположить их прямо на странице, привязать к их событиям обработчики и делать с их помощью все, что угодно. Например, менять цвет шрифта документа.

Далее вы узнаете, как можно управлять внешним видом страницы, используя так называемые "независимые" элементы управления, т. е. не привязанные к форме. Следующий пример более похож на традиционную Windows-программу, а не Web-страницу.

Новый пример страницы с изменяющимися стилями

В главе 3, посвященной взаимодействию HTML и JavaScript, мы создали пример Web-страницы с элементом, стиль текста которого изменялся при щелчках по соответствующим гиперссылкам. Давайте перепишем эту страницу так, чтобы использовать самые "современные" технологии — элементы управления, не привязанные к форме. И заодно добавим поле ввода и кнопку, с помощью которых можно будет изменить текст.

К несчастью, в Navigator это все равно не будет работать.

```
<HTML>
<HEAD>
<TITLE>Изменяющиеся стили</TITLE>
<STYLE>
#sample { font-weight: normal;
  font-style: normal;
  text-decoration: none }
</STYLE>
<SCRIPT>
function setupControls() {txtSample.value = sample.innerText; }
```

Эта функция будет вызвана сразу после загрузки страницы. Она поместит в поле ввода `txtSample` начальный текст.

```
function switchBold() {
  if (chkBold.checked)
```

Обратите внимание на то, как мы обращаемся к нашим флажкам. Теперь они не принадлежат форме, значит, имя формы указывать не нужно.

```
    sample.style.fontWeight = 'bold'
else
  sample.style.fontWeight = 'normal'; }
```

Это новая функция включения-отключения "жирности" шрифта текста элемента. Остальные две функции точно такие же.

```
function switchItalic() {
  if (chkItalic.checked)
    sample.style.fontStyle = 'italic'
  else
    sample.style.fontStyle = 'normal'; }

function switchUnder() {
  if (chkUnder.checked)
    sample.style.textDecoration = 'underline'
  else
    sample.style.textDecoration = 'none'; }

function setText() { sample.innerText = txtSample.value; }
```

А эта функция срабатывает при нажатии на кнопку и перемещает текст из поля ввода в элемент `sample`.

```
</SCRIPT>
</HEAD>
<BODY onload="setupControls();">
<TABLE>
<TR>
```

```
<TD VALIGN="top" BGCOLOR="#FFFFE0" WIDTH="150">
<INPUT TYPE="checkbox" ID="chkBold" NAME="chkBold"
↳onclick="switchBold();">
<LABEL FOR="chkBold">Полужирный</LABEL>
```

Подпись помещается всегда за флажком — это стандарт пользовательского интерфейса Windows. Не нужно его нарушать.

```
</TD>
<TD WIDTH="10" ROWSPAN="4">
</TD>
<TD BGCOLOR="#FFFFE0" WIDTH="*" ROWSPAN="4">
<P ID="sample">Здесь вы можете видеть результат щелчков по
флажкам.</P>
</TD>
</TR>
<TR>
<TD VALIGN="top" BGCOLOR="#FFFFE0" WIDTH="150">
<INPUT TYPE="checkbox" ID="chkItalic" NAME="chkItalic"
↳onclick="switchItalic();">
<LABEL FOR="chkItalic">Курсив</LABEL>
</TD>
</TR>
<TR>
<TD VALIGN="top" BGCOLOR="#FFFFE0" WIDTH="150">
<INPUT TYPE="checkbox" ID="chkUnder" NAME="chkUnder"
↳onclick="switchUnder();">
<LABEL FOR="chkUnder">Подчеркнутый</LABEL>
</TD>
</TR>
<TR>
<TD VALIGN="top" BGCOLOR="#FFFFE0" WIDTH="150">
<LABEL FOR="txtSample">Ввод текста</LABEL>
<INPUT TYPE="text" ID="txtSample" NAME="txtSample">
<INPUT TYPE="button" NAME="cmdSet" onclick="setText();" VALUE="Изменить">
```

А здесь — наоборот, сначала метка, а потом поле ввода. И это тоже правило интерфейса Windows.

```
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

Вот и весь код. Надо сказать, он сильно уменьшился по сравнению с первоначальной версией. Сохраните его под именем 5.4.htm и откройте в Internet

Explorer. Пощелкайте по флажкам, попробуйте изменить текст, содержащийся в поле ввода. На рис. 5.5 показано то, что может у вас получиться.

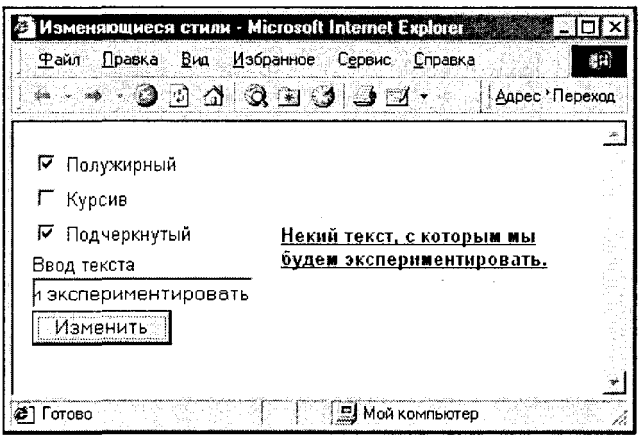


Рис. 5.5. Новая страница с изменяющимися стилями

Вот и все о формах и элементах управления. Вы узнали, как их можно использовать для передачи данных серверным приложениям, для изменения внешнего вида и содержимого Web-страницы, как проконтролировать данные, введенные пользователем, на правильность. На самом деле, формы и элементы управления можно использовать многими способами и решать с их помощью разнообразные задачи. Можно, например, создать Web-страницу, являющуюся интерфейсом к какой-нибудь программе, работающей на Web-сервере. Или встроить на свою страницу игру "15" (такие примеры есть). Все зависит только от фантазии Web-программиста.

А мы рассмотрим еще одну возможность, предлагаемую Internet Explorer, — это модальные и немодальные Web-окна.

Web-окна

Модальные Web-окна

Модальными диалоговыми окнами или просто диалогами в Windows называются окна, отображаемые поверх других окон этого же приложения и не дающие пользователю перейти в них, пока он не выполнит некоторое действие. Чтобы закрыть такое окно, пользователь должен либо ввести какие-то данные и нажать кнопку **OK**, либо отказаться от ввода и нажать кнопку **Отмена** или кнопку закрытия окна (если она есть). Модальные окна очень часто используются в Windows-приложениях для того, чтобы запросить у пользователя ввод каких-либо данных, необходимых для продолжения работы с приложением.

Internet Explorer начиная с версии 4.0 предлагает Web-программистам возможность создавать модальные Web-окна или, другими словами, обычные Web-формы, отображаемые в отдельном окне Web-обозревателя и не дающие пользователю переключиться в другие окна этого же приложения, пока он не введет какие-либо данные или не откажется от их ввода. Модальные Web-окна можно использовать для тех же целей, что и обычные модальные диалоги Windows-приложений, например для регистрации пользователя перед входом на сайт или запроса его личных данных для записи в базу данных.

Работа с модальными окнами

Для поддержки модальных Web-окон предусмотрен набор свойств объекта `window`. Эти свойства перечислены в табл. 5.10. Более подробное описание свойств и их использования мы дадим немного позже.

Таблица 5.10. *Свойства поддержки модальных Web-окон объекта* `window`

Свойство	Описание
`dialogArguments`	Возвращает значение или массив значений, переданных модальному Web-окну при его создании методом `window.open`. Доступно только в модальных Web-окнах
`dialogHeight`	Высота модального Web-окна
`dialogLeft`	Горизонтальная координата левого верхнего угла модального Web-окна
`dialogTop`	Вертикальная координата левого верхнего угла модального Web-окна
`dialogWidth`	Ширина модального Web-окна
`returnValue`	Возвращает значение или массив значений, переданных модальным Web-окном

Кроме того, объект `window` поддерживает метод `showModalDialog`, с помощью которого модальное Web-окно выводится на экран. Он имеет следующий формат:

```
window.showModalDialog({Адрес страницы формы}[[,{Аргументы}],
↳{Свойства окна}]);
```

Параметр `{Адрес страницы формы}` задает интернет-адрес Web-страницы с формой, которая будет показана в модальном Web-окне. Параметр `{Аргументы}` задает список аргументов, которые будут переданы в модальное Web-окно; доступ к ним осуществляется через свойство `window.dialogArguments`. А тре-

тий параметр метода следует описать подробнее. В нем передаются свойства создаваемого Web-окна, описанные в табл. 5.11.

Таблица 5.11. *Свойства модального Web-окна*

Свойство окна	Описание
`center=yes|no`*	Если `yes`, то создаваемое окно будет находиться в центре экрана. Значение по умолчанию — `yes`
`dialogHeight:{Значение}`**	Высота создаваемого окна
`dialogHide=yes|no`*	Если `yes`, то создаваемое окно будет скрываться при печати или предварительном просмотре. Значение по умолчанию — `no`
`dialogLeft:{Значение}`**	Горизонтальная координата левого верхнего угла создаваемого окна
`dialogTop:{Значение}`**	Вертикальная координата левого верхнего угла создаваемого окна
`dialogWidth:{Значение}`**	Ширина создаваемого окна
`edge=sunken|raised`	Задает вид границы окна: вдавленный (`sunken`) или выпуклый (`raised`). Значение по умолчанию — `raised`
`help=yes|no`*	Если `yes`, то создаваемое окно будет отображать в строке заголовка кнопку вызова контекстной справки. Значение по умолчанию — `yes`
`resizable=yes|no`*	Если `yes`, то пользователь сможет изменять размеры создаваемого окна. Значение по умолчанию — `no`
`scroll=yes|no`*	Если `yes`, то создаваемое окно будет отображать полосы прокрутки, если содержимое в нем не помещается. Значение по умолчанию — `yes`
`status=yes|no`*	Если `yes`, то создаваемое окно будет отображать строку состояния. Значение по умолчанию может в разных случаях быть различным, поэтому лучше задать это свойство явно
`unadorned=yes|no`*	Если `yes`, то создаваемое окно будет "неукрашено". Что это значит, выяснить не удалось, т. к. никакой разницы между "украшенным" и "неукрашенным" окном не заметно. Значение по умолчанию — `no`

* Вместо значений `yes` и `no` вы можете использовать `1` и `0` или `on` и `off`.

** Для указания единицы измерения размеров окна применяйте соответствующее обозначение, скажем, `px` для пикселов или `mm` для миллиметров. Вы также можете использовать относительные единицы измерения `em` и `ex`. В Internet Explorer версии 4.0 единицей измерения по умолчанию является пиксел `px`, в версии 5.0 — `em`.

Для того чтобы создать модальное окно, нужно написать примерно такое выражение.

```
window.showModalDialog("dialog.htm", ["param1", "param2], "scroll=no");
```

Здесь мы передали нашему модальному окну массив из двух параметров "param1" и "param2" и отключили у него полосы прокрутки. Модальное окно сможет получить доступ к этим параметрам, обратившись к свойству window.dialogArguments:

```
var par1, par2;
par1 = window.dialogArguments[0];
par2 = window.dialogArguments[1];
```

Для закрытия модального окна мы можем использовать метод window.close:

```
window.close();
```

В свою очередь, чтобы модальное окно смогло передать какие-то данные обратно в вызвавший его скрипт, нужно записать вызывающее выражение таким образом:

```
var varResult; res1, res2, res3;
varResult = window.showModalDialog("dialog.htm", ["param1", "param2],
↳"scroll=no");
res1 = varResult[0];
res2 = varResult[1];
res3 = varResult[2];
```

А само модальное окно должно будет воспользоваться свойством window.returnValue:

```
window.returnValue = [result1, result2, result3];
window.close();
```

Пример модального диалогового окна

Давайте рассмотрим небольшой пример Web-страницы, выводящей модальное Web-окно с полями ввода, где пользователь должен будет ввести свои имя и фамилию. По нажатии кнопки **OK** в главной странице отображается приветственный текст. Также предусмотрим переменные для хранения введенных имени и фамилии и при каждом отображении модального Web-окна будем подставлять их в поля ввода.

Сначала приведем код основной страницы, где будет отображаться приветствие.

```
<HTML>
<HEAD>
<TITLE>Приветствую вас!</TITLE>
```

```
<SCRIPT>
var name1 = "", name2 = "";
```

Это переменные для промежуточного хранения имени и фамилии пользователя.

```
function showNameDialog() {
```

Эта функция выводит модальное окно на экран.

```
var varResult;
varResult = window.showModalDialog("5.6.htm", [name1, name2],
↳"dialogHeight:120px; dialogWidth:300px; help=no; status=no");
```

Передаем методу `showModalDialog` имя файла модального окна, сохраненные имя и фамилию, введенные пользователем ранее, и ряд свойств, в частности размеры создаваемого окна. И присваиваем возвращенное этим методом значение локальной переменной — оно нам пригодится в дальнейшем.

```
if (varResult == null)
```

Проверяем возвращенное значение.

```
hello.innerText = "Не хотите здороваться — не надо..."
```

Если пользователь нажал кнопку **Отмена**, выводим соответствующий случаю текст.

```
else {
  name1 = varResult[0];
  name2 = varResult[1];
  hello.innerText = "Приветствую вас, " + name1 + " " + name2 +
  ↳"!"; } }
```

Иначе приветствуем пользователя по всей форме.

```
</SCRIPT>
</HEAD>
<BODY>
<TABLE WIDTH="100%">
<TR ALIGN="center" VALIGN="middle">
<TD>
<P ID="hello" STYLE="font-size: 18pt">Но сначала представьтесь...</P>
```

Здесь будет выводиться приветственный текст.

```
</TD>
</TR>
<TR ALIGN="center" VALIGN="middle">
<TD>
<INPUT TYPE="button" ID="cmdEnterName" VALUE="Введите свое имя"
↳onclick="showNameDialog();">
```

При нажатии на эту кнопку на экран выводится модальное Web-окно.

```
</TD>
</TR>
</BODY>
</HTML>
```

Вот и все. Сохраните этот код в файле 5.5.htm, но пока не открывайте, т. к. у нас еще нет файла 5.6.htm.

А теперь напишем код Web-страницы, которая будет отображена в модальном окне:

```
<HTML>
<HEAD>
<TITLE>Но сначала представьтесь...</TITLE>
<SCRIPT>
function setupData() {
  txtName1.value = window.dialogArguments[0];
  txtName2.value = window.dialogArguments[1]; }
```

Эта функция выполняется при загрузке Web-страницы. Она помещает имя и фамилию пользователя, сохраненные главной страницей, в соответствующие поля ввода.

```
function sendData () {
```

Эта функция выполняется при нажатии на кнопку **OK** и отправляет данные главному окну.

```
var name1, name2, dataRight = true, message;
name1 = txtName1.value;
name2 = txtName2.value;
if (name1.length == 0) {
  dataRight = false;
  message = "Введите имя."; }
else {
  if (name2.length == 0) {
    dataRight = false;
    message = "Введите фамилию."; } }
```

Проверяем, ввел ли пользователь нужные данные. Этот код практически целиком взят из примера страницы с проверкой ввода (файл 5.3.htm).

```
if (dataRight) {
  window.returnValue = [name1, name2];
  window.close(); }
```

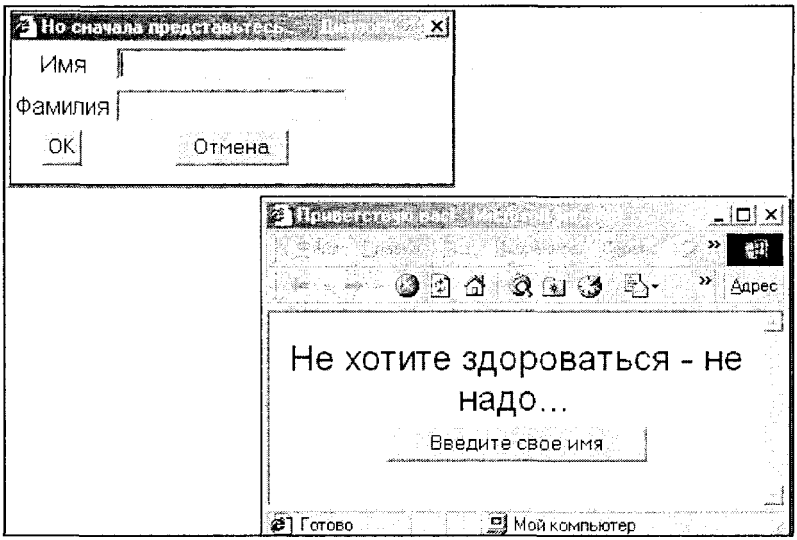


Рис. 5.6. "Приветствующая" Web-страница

Если данные введены правильно, помещаем их в виде массива в свойство `returnValue` и закрываем окно.

```
else
  window.alert(message); }
```

Если пользователь что-то забыл ввести, выводим напоминание.

```
function cancelData() {
  window.returnValue = null;
  window.close(); }
```

Эта функция выполняется при нажатии на кнопку **Отмена**. Она помещает в свойство `returnValue` `null` и закрывает окно.

```
</SCRIPT>
</HEAD>
<BODY onload="setupData();">
<TABLE>
<TR ALIGN="center" VALIGN="middle">
<TD><LABEL FOR="txtName1">Имя</LABEL></TD>
<TD><INPUT TYPE="text" ID="txtName1"></TD>
</TR>
<TR ALIGN="center" VALIGN="middle">
<TD><LABEL FOR="txtName2">Фамилия</LABEL></TD>
<TD><INPUT TYPE="text" ID="txtName2"></TD>
</TR>
<TR ALIGN="center" VALIGN="middle">
```

```
<TD><INPUT TYPE="button" ID="cmdOK" VALUE="OK"
↳onclick="sendData();"></TD>
<TD><INPUT TYPE="button" ID="cmdCancel" VALUE="Отмена"
↳onclick="cancelData();"></TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

Остальной код взят из примера страницы с проверкой ввода (файл 5.3.htm) с минимальными изменениями. Сохраните его в файле 5.6.htm.

Теперь откройте в Internet Explorer файл 5.5.htm. Нажмите кнопку **Введите свое имя**, заполните поля появившегося на экране модального окна и нажмите кнопку **OK**. У вас получится нечто, подобное показанному на рис. 5.6.

Немодальные Web-окна

Немодальными называются диалоговые окна, позволяющие пользователю переключаться в другие окна приложения. Они используются не так часто, как модальные, в основном, для того, чтобы позволить пользователю вводить и изменять какие-либо параметры приложения, не являющиеся критически важными. Как правило, немодальное диалоговое окно "плавает" поверх основного окна; при любом изменении данных пользователем информация в главном окне тотчас обновляется.

Работа с немодальными Web-окнами

Для работы с немодальными Web-окнами используется тот же набор свойств, что и для создания модальных окон. Однако для создания немодальных окон предназначен метод `window.showModelessDialog`, имеющий такой же формат, что и его коллега `window.showModalDialog`:

```
window.showModelessDialog({Адрес страницы формы}[[,{Аргументы}],
↳{Свойства окна}]);
```

Однако здесь есть одно существеннейшее "но". Если модальное окно создается на конечное время и после закрытия передает в вызвавший его скрипт главного окна какие-либо данные, то немодальное окно поступить так не может. В самом деле, немодальный диалог может быть открыт неопределенно долго, стало быть, он не сможет передать вызвавшему скрипту необходимые данные после закрытия. Поэтому в случае немодальных диалогов используется другой подход.

Давайте еще раз приведем пример вызова модального окна:

```
window.showModalDialog("dialog.htm", ["param1", "param2], "scroll=no");
```

Здесь мы передали создаваемому окну массив параметров. Чтобы передать те же параметры немодальному окну, делаем так. Создадим две глобальные переменные, где будут размещаться значения этих параметров:

```
var param1, param2;
```

Вызовем на экран наше немодальное окно и передадим ему в качестве параметра указатель на объект главного (родительского) окна:

```
window.showModelessDialog("dialog.htm", window, "scroll=no");
```

В немодальном окне извлекаем этот указатель:

```
var parentWindow;
parentWindow = window.dialogArguments;
```

И получаем доступ к переменным, содержащим нужные параметры, точно так же, как и к функциям-методам и свойствам:

```
tempParam1 = parentWindow.param1;
tempParam2 = parentWindow.param2;
```

Можете считать, что созданные в главном окне глобальные переменные — те же самые свойства, только предназначенные для особых нужд. Точно так же вы можете получить доступ к функциям, определенным в главном окне (считайте их методами этого окна):

```
parentWindow.refreshDate(d1, d2, f + 1);
```

Ну, а вернуть данные в главное окно теперь совсем просто:

```
parentWindow.datum1 = d1;
parentWindow.datum2 = d2;
parentWindow.datum3 = d3;
parentWindow.acceptData;
```

Пример немодального диалогового окна

А теперь перепишем предыдущий пример таким образом, чтобы использовать немодальное диалоговое окно для ввода параметров пользователя. Так как HTML-код не претерпит никаких изменений, приведем только код JavaScript-сценариев для новой версии окон.

Сначала приведем JavaScript-код главной страницы:

```
<SCRIPT>
var name1 = "", name2 = "";
```

Это наши глобальные переменные — новые свойства нашего главного окна.

```
function showNameDialog() {
  void window.showModelessDialog("5.8.htm", window,
  ↳"dialogHeight:120px; dialogWidth:300px; help=no; status=no"); }
```

Эта функция теперь просто показывает немодальное диалоговое окно, передавая ему в качестве параметра ссылку на главное окно.

```
function acceptData() { hello.innerText = "Приветствую вас, " + name1 +
↳" " + name2 + "!"; }
```

А эта функция занимается тем, что обновляет приветственный текст, беря имя и фамилию пользователя из переменных `name1` и `name2`. Заметьте также, что эта функция вызывается только из немодального диалогового окна, т. е. извне.

```
</SCRIPT>
```

Замените этим фрагментом код скриптов из файла 5.5.htm и сохраните его под именем 5.7.htm. Пока не открывайте его — у нас еще нет файла 5.8.htm.

А это — новые скрипты для немодального диалогового окна.

```
<SCRIPT>
var parentWindow;
```

Глобальная переменная, хранящая ссылку на главное окно.

```
function setupData() {
  parentWindow = window.dialogArguments;
  txtName1.value = parentWindow.name1;
  txtName2.value = parentWindow.name2; }
```

Эта функция вызывается при загрузке страницы. Она сохраняет в глобальной переменной ссылку на главное окно, переданную в качестве параметра. Кроме этого, пользуясь сохраненной ссылкой, она извлекает из свойств `name1` и `name2` главного окна имя и фамилию пользователя и помещает их в поля ввода.

```
function sendData () {
  var n1, n2, dataRight = true, message;
  n1 = txtName1.value;
  n2 = txtName2.value;
  if (n1.length == 0) {
    dataRight = false;
    message = "Введите имя."; }
  else {
    if (n2.length == 0) {
      dataRight = false;
      message = "Введите фамилию."; } }
  if (dataRight) {
    parentWindow.name1 = n1;
    parentWindow.name2 = n2;
    parentWindow.acceptData(); }
```

```
  else
    window.alert(message); }
```

Эта функция проверяет данные, введенные пользователем, и в случае правильности данных пересылает их в свойства name1 и name2 главного окна и вызывает его же метод acceptData. В результате всех этих действий приветствие в главном окне обновляется.

```
function cancelData() { window.close(); }
```

А эта функция, вызываемая при нажатии кнопки **Отмена**, просто закрывает диалоговое окно. Впоследствии вы сможете снова открыть его, нажав кнопку **Введите свое имя** главного окна.

```
</SCRIPT>
```

Замените этим фрагментом код скриптов из файла 5.6.htm и сохраните его под именем 5.8.htm.

После этого откройте файл 5.7.htm в Internet Explorer. То, что вы увидите, будет выглядеть и работать почти так же, как предыдущий пример за тем лишь исключением, что окно, где вводятся данные пользователя, не будет закрываться при нажатии кнопки **OK**. Попробуйте закрыть и снова открыть диалоговое окно, и вы увидите, что в полях ввода будут подставлены правильные значения имени и фамилии, введенные вами ранее.

Текстовые фрагменты

Internet Explorer предоставляет Web-программистам еще одну интересную и полезную возможность: работу с фрагментами текста, являющегося содержимым Web-страницы. Объекты фрагментов текста, доступные в Internet Explorer, бывают двух видов. Во-первых, это объект selection, предоставляющий доступ к тексту, выделенному в окне Web-обозревателя пользователем. И, во-вторых, это объекты класса TextRange, открывающий доступ к произвольным фрагментам текста.

Сначала мы изучим класс TextRange и предоставляемые им возможности управления текстом страницы.

Класс *TextRange*

Класс TextRange открывает доступ к некоторому фрагменту текста документа или элемента управления. Вы можете создать объект класса TextRange и манипулировать его содержимым как угодно.

Объект класса TextRange создается с помощью метода createTextRange. Этот метод поддерживается объектами тега документа body и элементами управления, имеющими внутри себя текст (поля ввода, область редактирования,

кнопки, скрытое поле). При этом содержимым полученного объекта становится все содержимое документа или элемента управления:

```
allText = document.body.createTextRange();
allAddress = txtAddress.createTextRange();
```

Получить доступ к содержимому созданного объекта вы можете, использовав свойства `text` или `htmlText`. Значение первого свойства представляет текст, содержащийся внутри объекта `TextRange`, значение второго — исходный HTML-код его содержимого. Свойство `text` доступно как для чтения, так и для записи, в то время как `htmlText` — только для чтения. Для изменения HTML-кода можно использовать метод `pasteHTML`:

```
address = allAddress.text;
htmlSource = allText.htmlText;
allAddress.text = newAddress;
allText.pasteHTML(newHTMLSource);
```

Класс `TextRange` поддерживает набор свойств, определяющий его характеристики. Эти свойства перечислены в табл. 5.12.

Таблица 5.12. *Свойства класса* `TextRange`

Свойство	Описание
`boundingHeight`	Возвращает высоту воображаемого прямоугольника, охватывающего текст-содержимое объекта `TextRange`
`boundingLeft`	Возвращает горизонтальную координату левого верхнего угла воображаемого прямоугольника, охватывающего текст-содержимое объекта `TextRange`. Координата отсчитывается относительно объекта, содержащего этот `TextRange`
`boundingTop`	Возвращает вертикальную координату левого верхнего угла воображаемого прямоугольника, охватывающего текст-содержимое объекта `TextRange`. Координата отсчитывается относительно объекта, содержащего этот `TextRange`
`boundingWidth`	Возвращает ширину воображаемого прямоугольника, охватывающего текст-содержимое объекта `TextRange`
`htmlText`	Возвращает "сырой" HTML-код содержимого объекта `TextRange`
`text`	Текстовое представление содержимого объекта `TextRange`

Также класс `TextRange` поддерживает набор методов для изменения размеров и границ и манипуляции его содержимым. Эти методы перечислены в табл. 5.13.

Таблица 5.13. *Методы класса* `TextRange`

Метод	Описание
`collapse(true|false)`	Сжимает объект `TextRange` в точку и в таком виде помещает его в начале (если в качестве параметра передано `true`; значение по умолчанию) или в конце (`false`) исходного объекта
`compareEndPoints({Объект}, {Диапазон})`	Сравнивает текущий объект `TextRange` с переданным в качестве первого параметра. Вторым параметром передается указание на то, как сравнивать объекты. Доступны четыре значения: `"StartToEnd"` сравнивает начальную точку текущего объекта с конечной точкой объекта, переданного первым параметром, `"StartToStart"` — начальную с начальной, `"EndToStart"` — конечную с начальной, `"EndToEnd"` — конечную с конечной. Возвращает `-1`, если сравниваемая точка текущего объекта находится левее (выше) сравниваемой точки объекта, переданного первым параметром; `0`, если они равны; `1`, если точка текущего объекта правее (ниже)
`duplicate()`	Возвращает новый объект `TextRange`, являющийся копией текущего
`expand({Элемент текста})`	Расширяет текущий объект так, чтобы любой элемент текста, частично включенный в область текста и соответствующий переданному в качестве параметра типу, был включен в него полностью. Доступны четыре значения параметра: `"character"` расширяет область текста на символ, `"word"` — на слово, `"sentence"` — на предложение, а `"textedit"` — на весь документ. Возвращает `true`, если объект был успешно расширен, и `false` — в противном случае
`findText({Что искать}[,{Размер области поиска}[,{Флаг}]])`	Ищет текст в текущем объекте и, при удачном поиске, "сжимает" объект, чтобы захватить найденный текст. Возвращает `true`, если поиск был удачным, и `false` — в противном случае. Вторым параметром может быть передан размер области текста, в которой будет производиться поиск, в виде количества символов, начиная с начала области текста. Третьим параметром передается дополнительный флаг поиска: `2` задает поиск только целых слов, `4` — поиск с совпадением регистра символов, а `6` объединяет эти флаги в один

Таблица 5.13 *(продолжение)*

Метод	Описание
`getBookmark()`	Возвращает "закладку" — уникальную текстовую строку, позволяющую в дальнейшем идентифицировать данную область текста
`inRange({Объект TextRange})`	Возвращает `true`, если объект, переданный в качестве параметра, содержится внутри текущего или равен ему
`isEqual({Объект TextRange})`	Возвращает `true`, если объект, переданный в качестве параметра, равен текущему
`move({Элемент текста}[,{Количество}])`	Сжимает текущий объект в точку и перемещает его на заданное количество элементов текста. Доступны четыре значения первого параметра, задающего элемент текста: `"character"` сдвигает область текста на символ, `"word"` — на слово, `"sentence"` — на предложение, а `"textedit"` перемещает ее в начало или конец области текста, находившейся в этом объекте ранее. Количество элементов может быть положительным или отрицательным; если оно отсутствует, выполняется сдвиг на один элемент. Возвращает количество элементов текста, на которое фактически был сдвинут объект
`moveEnd({Элемент текста}[,{Количество}])`	Перемещает конечную точку области текста. Параметры те же, что и у метода `move`.
`moveStart({Элемент текста}[,{Количество}])`	Перемещает начальную точку области текста. Параметры те же, что и у метода `move`
`moveToBookmark({Закладка})`	Перемещает текущий объект так, чтобы он охватил фрагмент текста, соответствующий ранее сохраненной "закладке", переданной в качестве параметра. Возвращает `true`, если перемещение прошло успешно, и `false` — в противном случае
`moveToElementText({Элемент страницы})`	Перемещает текущий объект так, чтобы он охватил текст элемента страницы, переданного в качестве параметра
`moveToPoint({X}, {Y})`	Сжимает текущий объект в точку и перемещает его в точку с заданными координатами. Координаты отсчитываются относительно окна
`parentElement()`	Возвращает ссылку на элемент страницы, полностью содержащий текущий объект `TextRange`

Таблица 5.13 *(окончание)*

Метод	Описание
`pasteHTML({Текст})`	Помещает переданный в качестве параметра HTML-код вместо прежнего содержимого текущего объекта `TextRange`
`scrollIntoView (true|false)`	Вызывает прокрутку страницы в окне Web-обозревателя так, что текст, входящий в текущий объект `TextRange`, появляется в окне. Если в качестве параметра передано `true`, то он окажется у верхнего края окна, если `false` — у нижнего
`select()`	Выделяет текст, входящий в текущий объект `TextRange`
`setEndPoints({Объект}, {Диапазон})`	Если вторым параметром передано `"StartToEnd"`, то перемещает начальную точку текущего объекта в конечную точку объекта, переданного первым параметром, `"StartToStart"` — начальную в начальную, `"EndToStart"` — конечную в начальную, `"EndToEnd"` — конечную в конечную

Класс `TextRange` событий не поддерживает, т. к. пользователь непосредственно с ним не общается.

Для каких целей можно применять объекты класса `TextRange`? Например, для того, чтобы позволить пользователю изменять какой-либо текст на Web-странице. Для этого можно воспользоваться свойствами `text` и `htmlText`, а также методом `pasteHTML`. Также объект класса `TextRange` полезен при поиске текста на Web-странице. Для этого можно написать такой код:

```
var rng;
rng = document.body.createTextRange();
if (rng.findText("Некие текст"))
  rgn.scrollIntoView(true);
```

Здесь мы ищем некую строку в полном тексте документа и прокручиваем содержимое страницы в окне так, чтобы сделать найденный текст видимым.

Объект *selection*

Объект `selection` предоставляет доступ к тексту, выделенному в окне Web-обозревателя. Этот объект поддерживается только Internet Explorer и доступен через свойство `selection` объекта `document`:

```
document.selection
```

Navigator же для доступа к выделенному тексту предоставляет метод `getSelection()` объекта `document`. Этот метод возвращает текстовую строку, представляющую выделенный текст:

```
strSelected = document.getSelection();
```

Конечно, вы можете использовать это значение для каких-либо действий. В Navigator это сделать проще, чем в Internet Explorer. Зато последний предоставляет несравнимо больше возможностей.

Пользователь может выделить текст на странице вручную. Можно сделать это и из скрипта, вызвав метод `select`, поддерживаемый объектами полей ввода и `textRange`:

```
someTextRange.select();
txtAddress.select();
```

Объект `selection` имеет свойство `type`, позволяющее узнать тип выделения, сделанного пользователем. Оно возвращает `"text"`, если на странице что-то выделено, и `"none"` — в противном случае. Кроме того, `selection` поддерживает ряд методов, перечисленных в табл. 5.14.

Таблица 5.14. *Методы класса* `selection`

Метод	Описание
`clear()`	Стирает выделенный текст
`createRange()`	Возвращает объект `TextRange`, включающий в себя выделенный текст
`empty()`	Убирает выделение с текста

Объект `selection` событий не поддерживает.

Пример страницы с изменяемым текстом

А теперь создадим небольшую Web-страницу, текст которой пользователь может изменять, просто выделив его и нажав кнопку **Изменить**. Также предусмотрим поле ввода и кнопку **Найти** для поиска текста на странице.

```
<HTML>
<HEAD>
<TITLE>Работа с текстом</TITLE>
<SCRIPT>
var rgn = null;
```

Эта глобальная переменная будет хранить ссылку на объект `TextRange`. Дело в том, что мы дали возможность пользователю искать текст в документе, и

при поиске будет создан объект TextRange, с помощью которого возможно изменение найденного фрагмента. Вот этот объект TextRange и будет здесь храниться.

```
function editText() {
```

Эта функция вызывается при нажатии кнопки **Изменить**.

```
var str1, str2;
if (rgn == null) {
```

Если объект TextRange еще не создан

```
if (document.selection.type == "none") {
```

и если пользователь ничего не выделил

```
window.alert("Сначала выделите какой-нибудь текст.");
return; }
```

выводим на экран предупреждение и выходим из функции.

```
else
  rgn = document.selection.createRange(); }
```

Иначе создаем объект TextRange, вмещающий в себя выделенный фрагмент документа.

```
str1 = rgn.htmlText;
str2 = window.prompt("Введите новый текст...", str1);
```

Выводим на экран окно с полем ввода, где пользователь должен будет вести новый текст.

```
if (str2 != null) rgn.pasteHTML(str2);
document.selection.empty();
rgn = null; }
```

Если пользователь действительно ввел новый текст и нажал кнопку **OK**, вставляем его на место старого содержимого TextRange. И обязательно обнуляем переменную rgn.

```
function findIt() {
```

Эта функция вызывается при нажатии кнопки **Найти**.

```
var s;
s = txtFind.value
if (s.length == 0)
  window.alert("Введите текст для поиска.")
```

Если пользователь забыл ввести в поле ввода искомый текст, напоминаем ему об этом.

```
else {
  rgn = document.body.createTextRange();
```

Иначе создаем объект `TextRange`, вмещающий в себя полный текст документа. Внутри этого объекта мы и будем искать введенный пользователем текст.

```
if (rgn.findText(txtFind.value)) {
  rgn.select();
  rgn.scrollIntoView(true); }
```

Если текст найден, выделяем его и прокручиваем содержимое окна так, чтобы он был виден. Если теперь нажать кнопку **Изменить**, будет использовано это же самое выделение, а не вновь созданное.

```
else
  rgn = null; } }
```

Иначе обнуляем переменную `rgn`.

```
</SCRIPT>
</HEAD>
<BODY>
<TABLE WIDTH="100%">
<TR ALIGN="center" VALIGN="middle">
<TD WIDTH="30%">
<INPUT TYPE="button" ID="cmdEdit" VALUE="Изменить..."
↳onclick="editText();">
</TD>
<TD>
<LABEL FOR="txtFind">Искать текст</LABEL>
<INPUT TYPE="text" ID="txtFind">
<INPUT TYPE="button" ID="cmdFind" VALUE="Найти" onclick="findIt();">
</TD>
</TR>
</TABLE>
<P>Вставьте сюда любой достаточно большой текст.</P>
</BODY>
</HTML>
```

Остальной HTML-код не таит никаких сюрпризов. Сохраните его в файле 5.9.htm и откройте в Internet Explorer. Попробуйте выделить какой-либо фрагмент текста и изменить его, потом испытайте поиск. В конце концов, у вас может получиться что-либо подобное рис. 5.7.

Вот мы и рассмотрели работу с текстовыми фрагментами в Internet Explorer. Конечно, реально эти знания могут быть полезными далеко не всегда: какой

же Web-дизайнер позволит пользователю изменять текст на своей странице! Однако если вы, скажем, выкладываете в Сеть большие текстовые документы, то можете предусмотреть на странице поиск по ключевому слову.

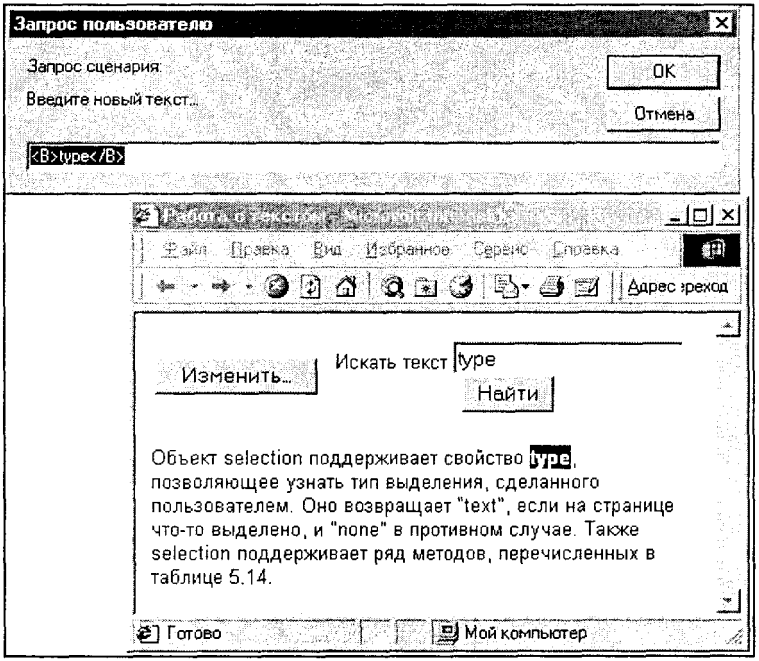


Рис. 5.7. Web-страница с возможностью поиска и изменения текста

Drag-n-drop и Буфер обмена Windows

Internet Explorer 5.0 и более новые его версии, кроме всего прочего, предлагает Web-программистам возможности работы со стандартным Буфером обмена Windows и организации drag-n-drop (перетаскивания) на создаваемых ими Web-страницах. При этом пользователь может перемещать выделенный текст из одного элемента страницы в другой. Также программист получает возможность управлять тем, что пользователь помещает в Буфер обмена. Это может быть применено, например, для помещения туда каких-либо дополнительных данных или, наоборот, для предотвращения этого; таким образом программист может запретить пользователю копировать со страницы текст, защищенный авторскими правами.

Здесь мы рассмотрим поочередно работу с Буфером обмена и поддержку drag-n-drop. В конце главы будет рассмотрено два примера.

Буфер обмена Windows

Здесь мы рассмотрим работу со стандартным Буфером обмена Windows.

Объект *clipboardData*

Объект `clipboardData` служит для доступа к данным, находящимся в Буфере обмена. Получить к нему доступ можно, обратившись к свойству `clipboardData` объекта `window`:

```
window.clipboardData
```

Для программиста доступны операции помещения, извлечения и удаления данных, имеющих один из пяти предопределенных форматов. Для этого объект `clipboardData` поддерживает три метода, перечисленных в табл. 5.15.

Таблица 5.15. *Методы объекта* `clipboardData`

Метод	Описание
`clearData([{Формат данных}])`	Удаляет из Буфера обмена данные, находящиеся в перечисленных в первом параметре форматах. Доступны пять форматов: `"Text"` — текстовый, `"URL"` — интернет-адрес, `"File"` — файл, `"HTML"` — HTML-код и `"Image"` — графическое изображение. Допустимо задавать несколько форматов, перечислив их через запятую. Если параметр отсутствует, из Буфера обмена будут удалены все данные
`getData({Формат данных})`	Возвращает данные из Буфера обмена в заданном формате. Доступно два формата: `"Text"` — текстовый и `"URL"` — интернет-адрес; одновременно можно задавать только один формат
`setData({Формат данных},{Данные})`	Помещает в Буфер обмена данные в заданном формате. Доступно два формата: `"Text"` — текстовый и `"URL"` — интернет-адрес; одновременно можно задавать только один формат. Возвращает `true`, если данные помещены в Буфер обмена

Поддержка Буфера обмена элементами страниц

Для обработки взаимодействия с Буфером обмена все элементы страниц поддерживают ряд событий. Эти события перечислены в табл. 5.16.

Таблица 5.16. События взаимодействия с Буфером обмена

Событие	Описание
onbeforecopy	Наступает перед копированием выделенного содержимого элемента страницы в Буфер обмена
onbeforecut	Наступает перед вырезанием выделенного содержимого элемента страницы в Буфер обмена
onbeforepaste	Наступает перед вставкой содержимого Буфера обмена в элемент страницы
oncopy	Наступает при копировании выделенного содержимого элемента страницы в Буфер обмена
oncut	Наступает при вырезании выделенного содержимого элемента страницы в Буфер обмена
onpaste	Наступает при вставке содержимого Буфера обмена в элемент страницы

Работа с Буфером обмена

События `onbeforecopy`, `onbeforecut` и `onbeforepaste` можно использовать, чтобы разрешить или запретить пункты контекстного меню **Скопировать**, **Вырезать** и **Вставить** соответственно (и, разумеется, соответствующие клавишные комбинации). Для этого достаточно присвоить свойству `returnValue` объекта `event` значение `false` (чтобы разрешить; в этом случае мы отменяем поведение меню по умолчанию) или `true` (чтобы запретить). (По умолчанию пункт **Скопировать** разрешен всегда, тогда как пункты **Вырезать** и **Вставить** всегда запрещены, т. к. пользователь не имеет права изменять содержимое Web-страницы.)

```
<P ID="pSource" onbeforecut="pBeforeCut();">Вырежите этот текст</P>
function pBeforeCut() { window.event.returnValue = false; }
```

Сейчас мы отменили поведение Web-обозревателя по умолчанию и разрешили пункт меню **Вырезать**.

Заметьте также, что если вы разрешили пользователю вырезать и вставлять текст на своей странице, то не придется писать для этого дополнительный код. Web-обозреватель выполняет заданные операции сам.

Если вы захотите запретить пункт меню **Скопировать**, выполните аналогичный код:

```
<P ID="pSource" onbeforecopy="pBeforeCopy();">А этот текст не
↳копируется</P>
function pBeforeCopy() { window.event.returnValue = false; }
```

Так вы сможете защитить фрагмент страницы от несанкционированного копирования. Чтобы сделать это для всей страницы, напишите обработчик события onbeforecopy для тела документа.

```
<BODY onbeforecopy="window.event.returnValue = false;">
```

Код разрешения операции вставки выглядит аналогично:

```
<P ID="pTarget" onbeforepaste="window.event.returnValue = false;">
↳Вставьте текст сюда</P>
```

Опять же, никакого дополнительного кода для реализации вставки текста не требуется. Web-обозреватель выполнит все сам. Единственное "но": для того чтобы вставить текст в элемент, не предназначенный для редактирования текста (все, кроме полей ввода), нам придется выделить небольшой фрагмент текста этого элемента. Неудобно, но что поделать — каждому элементу страницы должна отводиться своя роль. Что касается полей ввода, текст в них вставляется без всяких трудностей.

События oncopy, oncut и onpaste можно использовать для выполнения операций копирования, вырезания и вставки текста, если данные, помещаемые в Буфер обмена или извлекаемые из него, нужно дополнительно обработать. Но перед этим необходимо отменить выполнение этих операций по умолчанию, присвоив false свойству returnValue объекта event:

```
<P ID="pSource" onbeforecopy="window.event.returnValue = false;"
↳oncopy=sourceCopy();">Скопируйте этот текст</P>
function sourceCopy() {
  window.event.returnValue = false;
  window.clipboardData.setData("Text", pSource.innerText); }
```

Здесь мы реализовали специальную операцию копирования текста, содержимое элемента pSource копируется в Буфер обмена целиком. Вы можете проверить этот код на правильность, написав небольшую страничку и загрузив ее в Internet Explorer. Если вы выделите небольшой фрагмент текста в элементе pSource и скопируете его в Буфер обмена, то впоследствии обнаружите (скажем, вставив его в Блокнот или воспользовавшись утилитой Просмотр Буфера обмена, поставляемой в комплекте с Windows), что на самом деле был скопирован весь текст.

Точно так же можно реализовать специальную операцию для вырезания и вставки текста. Опять же, по умолчанию Web-обозреватель вырезает только выделенный текст и вставляет то, что находится в Буфере обмена (естественно, только текстовые данные). Чтобы запретить поведение по умолчанию и реализовать эти операции по-своему, присвойте false свойству returnValue объекта event и потом делайте с текстом, что хотите.

Зачем все это может понадобиться? Например, в электронном магазине, когда пользователь щелкает правой кнопкой мыши по фотографии товара и копирует в Буфер обмена не рисунок, а полное его описание в текстовом виде. Или опять-таки для защиты выложенной в Сеть информации от несанкционированного копирования.

Drag-n-drop

А здесь мы рассмотрим реализацию операции drag-n-drop средствами JavaScript и DOM.

Скажу сразу, что здесь имеется в виду не перетаскивание элементов страницы, рассмотренное и реализованное нами в предыдущей главе, а перенос методом drag-n-drop данных. Таким образом можно переносить данные из одного элемента страницы в другой, почти так же, как и с использованием Буфера обмена. Но если Буфер обмена в системе один на все приложения, drag-n-drop использует свое хранилище информации, не зависимое от системы.

Но, прежде всего, условимся о терминологии. Элемент, который пользователь перемещает по странице с целью "бросить" куда-нибудь, назовем *источником*. Элемент, куда пользователь "бросит" источник, назовем *приемником* или *целью*.

Объект *dataTransfer*

Объект `dataTransfer` служит для временного хранения информации во время операции drag-n-drop. Он чем-то подобен объекту Буфера обмена `clipboardData`, за тем исключением, что принадлежит приложению, а не системе. Доступ к этому объекту можно получить через свойство `dataTransfer` объекта `event`.

```
window.event.dataTransfer
```

Объект `dataTransfer` поддерживает те же методы, что и `clipboardData`: `clearData`, `getData` и `setData`. Кроме этого, он поддерживает два дополнительных свойства.

Свойство `dropEffect` задает тип операции drag-n-drop для элемента-цели. Доступны четыре значения: `"copy"` — операция копирования, `"link"` — создания ссылки, `"move"` — переноса и `"none"` — операция drag-n-drop не разрешена.

Свойство `effectAllowed` задает тип операции drag-n-drop для элемента-источника. Возможных значений типа здесь больше; все они перечислены в табл. 5.17. Согласно заданному типу операции будет отображен соответствующий курсор мыши.

Таблица 5.17. *Типы операции drag-n-drop для элемента-источника*

Тип	Описание
`copy`	Копирование данных
`link`	Создание ссылки на данные
`move`	Перемещение данных
`copyLink`	Копирование данных или создание ссылки на данные в зависимости от элемента-цели
`copyMove`	Копирование или перемещение данных в зависимости от элемента-цели
`linkMove`	Перемещение данных или создание ссылки на данные в зависимости от элемента-цели
`all`	Все операции
`none`	Операция drag-n-drop не разрешена
`uninitialized`	Значение по умолчанию. Ни одно из вышеперечисленных значений не было присвоено

Поддержка drag-n-drop элементами страниц

Для выполнения операций drag-n-drop все элементы страниц поддерживают ряд событий, перечисленных в табл. 5.18.

Таблица 5.18. *События поддержки drag-n-drop*

Событие	Описание
`ondrag`	Наступает, когда пользователь перемещает элемент-источник по странице. Всегда передается элементу-источнику
`ondragend`	Наступает, когда пользователь отпускает элемент-источник, тем самым, заканчивая операцию перетаскивания. Всегда передается элементу-источнику
`ondragenter`	Наступает, когда пользователь помещает элемент-источник над элементом-целью. Всегда передается элементу-цели
`ondragleave`	Наступает, когда пользователь убирает элемент-источник с элемента-цели. Всегда передается элементу-цели
`ondragover`	Наступает, когда пользователь перемещает элемент-источник над элементом-целью. Всегда передается элементу-цели
`ondragstart`	Наступает, когда пользователь начинает перетаскивать мышью элемент страницы. Всегда направляется элементу-источнику

Таблица 5.18 (окончание)

Событие	Описание
ondrop	Наступает, когда пользователь отпускает элемент-источник над элементом-целью, тем самым, заканчивая операцию перетаскивания. Всегда передается элементу-цели

Порядок наступления этих событий показан в табл. 5.19.

Таблица 5.19. Порядок наступления событий при перетаскивании элемента

Действие	События, направляемые источнику	События, направляемые цели
1. Пользователь щелкает мышью по элементу и начинает перетаскивание	ondragstart	
2. Пользователь перемещает перетаскиваемый элемент (источник) по странице	ondrag	
3. Пользователь помещает элемент-источник на другой элемент (цель)	ondrag	ondragenter
4. Пользователь перемещает элемент-источник над элементом-целью	ondrag	ondragover
5. Пользователь убирает элемент-источник с элемента-цели	ondrag	ondragleave
6. Пользователь отпускает кнопку мыши над элементом-целью, завершая тем самым операцию перетаскивания	ondragend	ondrop

Реализация drag-n-drop

Реализация операции drag-n-drop на Web-странице состоит, в основном, из трех этапов:

- *начало операции*, когда пользователь приступает к перемещению элемента-источника. Здесь свойству `effectAllowed` присваивается значение допустимого типа операции, и в объект `dataTransfer` помещаются нужные данные;
- *опознание цели*. При перемещении над элементом страницы, могущим быть целью, свойству `dropEffect` присваивается значение допустимого типа операции. Если оно совпадает с аналогичным значением элемента-источника, изменяется внешний вид курсора мыши;

- *завершение операции* — пользователь "бросает" элемент-источник на элемент-цель. Данные, сохраненные ранее в объекте `dataTransfer`, извлекаются и используются тем или иным способом.

Теперь рассмотрим эти этапы подробнее.

Первый этап — начало операции. Рассмотрим подробнее, как он реализуется.

Web-обозреватель сам умеет выполнять drag-n-drop с выделенным текстом. Как правило, при этом текст будет копироваться (т. е. свойство `effectAllowed` будет установлено в `"copy"`). Вы можете написать свой обработчик события `ondragstart` для источника и задать в нем иной тип операции, а также поместить в объект `dataTransfer` другие данные (не выделенный текст, а что-то иное):

```
function sourceDragStart() {
  window.event.dataTransfer.effectAllowed = "move";
  window.event.dataTransfer.setData("Text",
  ↳"Этот текст мы и будем таскать."); }
```

Здесь мы реализовали операцию переноса и поместили в `dataTransfer` посторонний текст.

Заметьте, что мы не отменяли обработку события по умолчанию. Если же вы сделаете это, присвоив `false` свойству `returnValue` объекта `event`, операция drag-n-drop не выполнится.

Теперь поговорим о втором этапе. Некоторые элементы страницы (например, поля ввода) выполняют его сами; других же нужно специально об этом просить. "Просьба" реализуется в обработчике события `ondragenter` цели. Если допустимые типы операции для источника и цели совпадут, Web-обозреватель изменит вид курсора, говоря о том, что пользователь может "бросить" источник, тем самым, прекратив операцию drag-n-drop.

Обработчик события `ondragenter` цели может иметь, например, такой вид:

```
function targetDragEnter() {
  window.event.returnValue = false;
  window.event.dataTransfer.dropEffect = "move"; }
```

Здесь мы также задаем допустимое действие для цели, отменив сначала обработку по умолчанию, иначе элемент-цель не примет данные.

Третий этап выполняется полями ввода также непосредственно, без всякого кодирования. Для других элементов страницы обработка реализуется в коде обработчика события `ondrop` цели. В нем мы, опять же, отменяем обработку по умолчанию, иначе элемент-цель не примет данные:

```
function targetDrop() {
  window.event.returnValue = false;
  pTarget.innerText = window.event.dataTransfer.getData("Text"); }
```

Разумеется, это простейший вариант реализации drag-n-drop. Вы можете создать более сложный механизм, реализующий разные типы операций. Скажем, для некоторых объектов-целей будет разрешена операция перемещения данных, а для других — только копирование. Также можно каким-то образом обрабатывать событие ondrag источника, например отображать в строке состояния или еще где-либо координаты курсора мыши или иные данные.

Пример страницы

Теперь давайте рассмотрим пример Web-страницы, в которой были реализованы работа с Буфером обмена и поддержка drag-n-drop. Эта страница будет содержать три элемента, текст из которых можно будет перетащить или перенести через Буфер обмена в четвертый.

```
<HTML>
<HEAD>
<TITLE>Буфер обмена и drag-n-drop</TITLE>
<STYLE>
#source1 { color: white; background-color: teal; left: 10; top: 10;
↳width: 80; height: 100; position: absolute }
#source2 { color: white; background-color: teal; left: 110; top: 10;
↳width: 80; height: 100; position: absolute }
#source3 { color: white; background-color: teal; left: 210; top: 10;
↳width: 80; height: 100; position: absolute }
#target { color: white; background-color: black; position: absolute;
↳left: 110; top: 150; width: 80; height: 100 }
</STYLE>
```

Координаты свободно позиционированных элементов задаем в таблице стилей.

```
<SCRIPT>
function sourceDragStart(s) {
  window.event.dataTransfer.effectAllowed = "copy";
  window.event.dataTransfer.setData("text", s.innerText); }
```

Эта функция вызывается, когда пользователь начинает перетаскивать элемент-источник. Она задает допустимую операцию для источника и помещает все содержимое элемента-источника в объект dataTransfer. В качестве первого параметра передается ссылка на источник; она нужна, чтобы функция смогла извлечь его содержимое.

```
function targetDragEnter() {
  window.event.returnValue = false;
  window.event.dataTransfer.dropEffect = "copy"; }
```

```
function targetDrop() {
  window.event.returnValue = false;
  target.innerText = window.event.dataTransfer.getData("text"); }
```

Этот код вам, конечно, уже знаком. Аналогичные функции мы уже рассмотрели ранее.

```
function sourceCopy(s) {
  window.event.returnValue = false;
  window.clipboardData.setData("Text", s.innerText); }
```

Эта функция вызывается при копировании текста из элемента-источника. Она помещает все его содержимое в Буфер обмена.

```
function targetPaste() {
  window.event.returnValue = false;
  target.innerText = window.clipboardData.getData("Text"); }
```

А эта функция вызывается при вставке теста в приемник. Она замещает все содержимое элемента-приемника содержимым Буфера обмена.

```
</SCRIPT>
</HEAD>
<BODY>
<DIV ID="source1" ondragstart="sourceDragStart(this);"
oncopy="sourceCopy(this);" CLASS="source">Элемент 1</DIV>
<DIV ID="source2" ondragstart="sourceDragStart(this);"
oncopy="sourceCopy(this);" CLASS="source">Элемент 2</DIV>
<DIV ID="source3" ondragstart="sourceDragStart(this);"
oncopy="sourceCopy(this);" CLASS="source">Элемент 3</DIV>
```

Это элементы-источники.

```
<DIV ID="target" ondragenter="targetDragEnter();" ondrop="targetDrop();"
ondragover="window.event.returnValue = false;"
onbeforepaste="window.event.returnValue = false;"
onpaste="targetPaste();">Цель</DIV>
```

А это элемент-приемник.

```
</BODY>
</HTML>
```

Вот и весь код. Сохраните его в файле 5.10.htm и откройте в Internet Explorer. Попробуйте перетащить текст из элемента-источника в элемент-приемник, затем испытайте операцию копирования и вставки. В конце концов, у вас получится что-то, похожее на рис. 5.8.

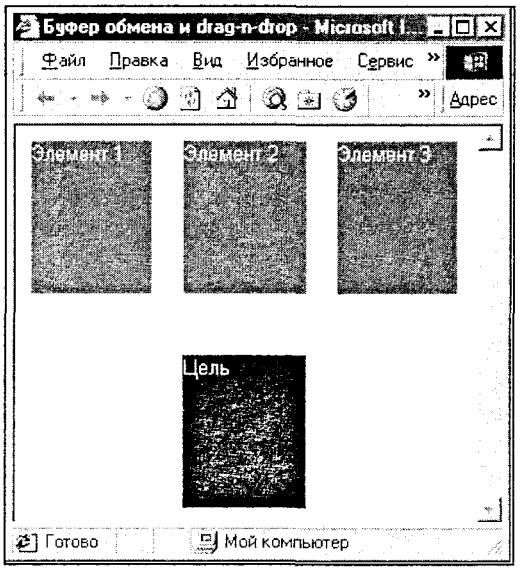


Рис. 5.8. Web-страница с поддержкой drag-n-drop и копирования-вставки текста

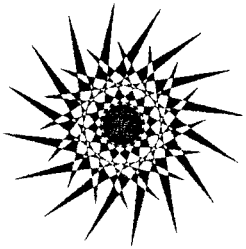
Хотите знать больше?

Вот мы и научились принимать от пользователя полезные данные, отправлять их серверной программе и обрабатывать на стороне клиента. Мы выяснили, что можно делать с Web-формами и элементами управления и узнали, как можно изменить текст Web-страницы и позволить сделать это пользователю. Что дальше?

Знаете ли вы, какие программы сейчас наиболее популярны? Программы, работающие с базами данных. Именно их сейчас пишут и поддерживают миллионы программистов по всему миру, с ними работают десятки миллионов пользователей. Эти бесчисленные бухгалтерские, складские, банковские, учетные приложения бьют все рекорды популярности. И нет ничего удивительного в том, что стали появляться Web-страницы, работающие с базами данных.

О них мы и будем говорить в дальнейшем.

ГЛАВА 6



Работа с базами данных

Здесь мы расскажем, как создаются Web-страницы, работающие с базами данных.

Для чего может понадобиться создавать такие страницы? Предположим, что некая фирма имеет Web-сайт с прайс-листами своей продукции. Допустим, что вся деятельность этой фирмы автоматизирована с использованием программных средств, не важно, серийных пакетов или собственных разработок. Очень часто в этом случае будет возникать задача обновления информации, содержащейся в прайс-листах, из баз данных складской программы. Если выполнять обновление вручную, на это уйдет очень много времени. При частом обновлении Web-дизайнер только и будет заниматься переносом данных из складской программы на Web-страницы. Что делать?

Единственный выход в этой ситуации — написать Web-страницы прайс-листов, которые будут брать информацию прямо из баз данных складской программы.

Далее. Вы, наверняка, видели на многих Web-страницах формы регистрации в различных списках. Это могут быть списки пользователей какого-либо сайта, списки получателей рассылок и т. п. Формы регистрации — классический пример страниц, работающих с базами данных.

Но прежде, чем перейти непосредственно к вопросам программирования таких страниц, выясним, что такое база данных, определимся с терминологией и рассмотрим два принципиально разных подхода к работе с базами данных.

Базовые понятия

Прежде чем начать реализовывать на Web-страницах доступ к базам данных, давайте решим для себя, что же это такое, и дадим несколько определений, без которых не обойтись в дальнейшем.

Что такое база данных

Итак, *база данных* — это хранилище специально организованных данных, записанных в особом формате. Подавляющее большинство баз данных, эксплуатируемых в настоящее время, имеют так называемый *реляционный* формат, т. е. организованы в виде таблиц. Каждая такая база данных может включать в себя одну или большее количество таблиц; сложные базы данных, как правило, имеют много таблиц, связанных между собой. Таблица в свою очередь состоит из набора строк, разделенных на ячейки, причем в каждой ячейке содержатся данные определенного типа: текст, числа, логические величины, даты и т. д. Строки таблицы баз данных называются *записями*, а ячейки, на которые делится запись, — *полями*. Как я уже говорил, поле имеет определенный тип и обязательно снабжено *заголовком*, по которому программа, работающая с данными, и осуществляет доступ.

Доступ к отдельным записям производится несколько по-другому. В один конкретный момент времени программа может работать только с одной записью, называемой текущей. Разумеется, программа должна иметь возможность перемещаться от одной записи к другой для того, чтобы получить все данные базы. Программа может добавлять, изменять и удалять записи базы данных, если сама база такое позволяет.

Доступ к базам данных производится при помощи *процессора данных*, либо встроенного в саму программу, либо представляющего собой отдельную DLL-библиотеку. В составе Microsoft Windows поставляется встроенный процессор данных *ODBC* (Open DataBase Connectivity — открытый доступ к базам данных). Известен также процессор данных фирмы Borland — *BDE* (Borland Database Engine — процессор баз данных Borland).

В последнее время большую популярность приобрели *серверы баз данных* — особые серверные программы, реализующие сложную обработку данных по заказам клиентских программ. Клиентская программа в этом случае получает доступ к серверу баз данных по сети (в том числе по Интернету) и запрашивает у него данные для работы; все команды на изменение и удаление записей также пересылаются серверу. Для составления сложных запросов к серверам баз данных был разработан язык описания запросов *SQL* (Structured Query Language — язык структурированных запросов). Некоторые обычные, клиентские процессоры данных также предоставляют возможность использования SQL.

Два подхода к работе с базами данных

Как можно работать с базами данных, используя Web-страницы? По-разному. В настоящее время существуют два подхода: серверный и клиентский.

При *серверном подходе* с базой данных работает серверная программа. (Очень часто и Web-сервер, и сервер баз данных располагаются на одном компьютере.) Чтобы получить какие-либо данные, пользователь посылает

запрос серверной программе, которая извлекает данные из базы и формирует на их основе Web-страницу. Возможно, пользователь также задает какие-то параметры, например фильтр для выборки данных, и серверная программа учитывает их. После этого пользователь получает в окне Web-обозревателя обычную Web-страницу, содержащую выбранные данные.

```
http://www.supershop.ru/catalogs/search.exe?good=tapochkifortarakans
```

Это типичный интернет-адрес серверной программы, извлекающей данные из базы. Как видите, он содержит параметры, которые будут переданы серверной программе методом `get`. Этот пример мы рассмотрели в предыдущей главе, посвященной Web-формам.

Каковы преимущества и недостатки серверного подхода? Разумеется, вся обработка данных переносится с клиента на сервер, что позволяет работать с данными даже на сравнительно маломощных компьютерах. Кроме того, серверная программа возвращает обычную Web-страницу, которую можно просмотреть в любом Web-обозревателе. А это тоже большое достоинство. Однако на серверный компьютер при этом ложится огромная нагрузка: помимо Web-сервера он является еще и сервером базы данных, который очень требователен к ресурсам. При одновременном поступлении большого количества запросов серверный компьютер может с ними не справиться. Конечно, можно разместить Web-сервер и сервер данных на разных компьютерах, но такое могут позволить себе только достаточно большие организации.

Этих недостатков лишен *клиентский подход*, когда вся обработка данных происходит на клиентском компьютере, т. е. на компьютере пользователя. При этом Web-обозреватель сам запрашивает данные у сервера, а сервер их передает. После этого Web-обозреватель, сообразуясь со специальными атрибутами обычных тегов HTML, форматирует данные в таблицу или помещает их в формы для правки.

Однако этот подход тоже имеет свои недостатки. Во-первых, для него нужен достаточно мощный компьютер. Во-вторых, обработку данных на стороне клиента поддерживает только Internet Explorer версии 4.0 или более поздней. А в-третьих, возрастает объем данных, посылаемых по Сети, что может перегрузить медленный канал связи.

Поскольку написание серверных программ выходит за рамки этой книги, мы рассмотрим клиентский подход. Но сначала поговорим об одном HTML-теге, о котором мимоходом было упомянуто еще в главе 1. Это тег внедрения на страницу элементов ActiveX. Нужен он нам потому, что работа с базами данных в Internet Explorer реализована именно через элементы ActiveX. Ранее я говорил только о *видимых* элементах ActiveX, т. е. тех, которые визуально присутствуют на странице и показывают какую-либо инфор-

мацию. Но элементы ActiveX бывают также и *невидимыми*; они присутствуют на странице и выполняют те или иные действия, но визуально себя не проявляют.

Тег <OBJECT>...</OBJECT>

Парный тег <OBJECT> служит для помещения на страницу элемента ActiveX. Его формат приведен ниже:

```
<OBJECT CLASSID="clsid:{Идентификатор класса}" [TYPE="{Тип данных MIME}"]
[CODE="{Интернет-адрес файла дистрибутива}"]
[DATA="{Интернет-адрес файла данных}"]>
. . . Теги параметров
</OBJECT>
```

Важнейшим атрибутом этого тега является CLASSID. Он задает *уникальный идентификатор класса*, под которым элемент ActiveX регистрируется в Реестре Windows. Именно по этому идентификатору Windows определяет, что помещать на страницу и в каком файле это находится. Уникальный идентификатор класса представляет собой 128-разрядное двоичное число, записываемое в шестнадцатеричной форме, так что постарайтесь не ошибиться, когда будете его набирать.

В дальнейшем я приведу уникальные идентификаторы класса для элементов ActiveX, которые нам понадобятся. А сейчас продолжим рассмотрение тега <OBJECT>.

Атрибут TYPE не обязателен и задает тип данных MIME. Для элемента ActiveX он всегда равен "application/x-ole-object".

Атрибут CODE задает интернет-адрес файла дистрибутива элемента ActiveX. Если нужный элемент не присутствует в системе, Web-обозреватель сам запросит его с указанного в атрибуте CODE адреса и установит.

Некоторые элементы ActiveX требуют дополнительных данных для работы. Интернет-адрес файла с такими данными указывается в атрибуте DATA.

Кроме того, тег <OBJECT> поддерживает атрибуты ALIGN, CLASS, HEIGHT, HSPACE, ID, NAME, STYLE, TABINDEX, VSPACE и WIDTH. Некоторые из них применимы только для видимых элементов ActiveX.

Каждый элемент ActiveX может принимать набор параметров. Такими параметрами могут быть, скажем, интернет-адрес файла (или файлов) данных (если, помимо заданного в атрибуте DATA тега <OBJECT> требуются какие-то другие данные), цвет фона, размер шрифта надписей и т. п. Для того чтобы описать все параметры и их значения, используется одинарный тег <PARAM>. Список определений параметров, сделанных с помощью этого тега, располагается внутри тега <OBJECT>.

Тег <PARAM> имеет очень простой формат.

```
<PARAM NAME="{Имя параметра}" VALUE="{Значение параметра}">
```

Я даже не буду описывать подробно каждый атрибут — его назначение понятно из приведенного выше краткого описания.

Разумеется, тег <OBJECT> поддерживает большое количество свойств, методов и событий, но я не буду их здесь описывать. Полное описание этого тега приведено в приложениях 1 и 3. Отмечу только одно свойство, которое нам впоследствии очень понадобится, — object. Оно позволяет получить доступ к элементу ActiveX и воспользоваться его свойствами, методами и событиями.

Теперь самое время привести небольшой пример внедрения на Web-страницу элемента ActiveX. Сразу предупрежу, что это не какой-либо конкретный элемент — его уникальный идентификатор класса просто выдуман, как и другие параметры.

```
<OBJECT CLASSID="clsid:99B42120-6EC7-11CF-A6C7-00AA00A47DD2"
ID="objSome" WIDTH="200" HEIGHT="200"
CODE="http://www.somecite.net/downloads/someactivex.cab">
<PARAM NAME="backgroundcolor" VALUE="teal">
<PARAM NAME="foregroundcolor" VALUE="black">
<PARAM NAME="fontsize" VALUE="10">
<PARAM NAME="headerfontsize" VALUE="24">
</OBJECT>
```

Вот и все об элементах ActiveX. Теперь пора переходить к основной теме этой главы — доступу к базам данных.

Доступ к текстовым таблицам

Текстовые таблицы — простейший вид баз данных. Это обычные текстовые файлы, где записи представлены в виде длинных строк, состоящих из полей фиксированной длины либо разделенных каким-нибудь символом (обычно, запятой или знаком табуляции). Доступ к таким данных Web-обозреватель реализует сам, без внешнего процессора данных. Единственный недостаток текстовых таблиц: данные, содержащиеся в них, не могут быть изменены пользователем.

Элемент *TDC*

Для доступа к текстовым таблицам используется специальный элемент ActiveX, называемый *TDC* (Tabular Data Control — элемент управления табличными данными). Его включение в HTML-страницу реализуется нижеуказанным кодом.

```
<OBJECT CLASSID="clsid:333C7BC4-460F-11D0-BC04-0080C7055A83"
↳ID="{Имя объекта}" WIDTH="0" HEIGHT="0">
<PARAM NAME="DataURL" VALUE="{Имя файла данных}">
<PARAM NAME="UseHeader" VALUE="true|false">
</OBJECT>
```

Здесь мы установили атрибуты `WIDTH` и `HEIGHT` равными нулю, чтобы внедренный в страницу элемент ActiveX не был на ней виден. Параметр `UseHeader` элемента TDC задает, будет ли первая запись текстовой таблицы рассматриваться как набор заголовков для ее полей.

Вот и весь код. Описание других полезных параметров элемента TDC приведено в табл. 6.1. Более подробное и полное описание элемента TDC содержится в приложении 3.

Таблица 6.1. *Параметры элемента TDC*

Параметр	Описание
`CaseSensitive`	Если `true` (значение по умолчанию), при отображении, фильтрации и сортировке данных будет учитываться регистр символов. Если `false`, регистр учитываться не будет
`CharSet`	Задает набор символов для данных текстовой таблицы. По умолчанию `"windows-1525"` (набор символов западноевропейских языков). Чтобы нормально отображались символы кириллицы, установите набор символов `"windows-1251"` (кириллица). Полный список поддерживаемых языков приведен в приложении 1
`DataURL`	Интернет-адрес файла текстовой таблицы
`EscapeChar`	Символ, служащий для обозначения специальных символов. Например, в файлах, где значения полей отделены друг от друга запятыми, запятые нельзя использовать внутри значений полей. Но если установить значение этого параметра, скажем, в `"\"`, можно обозначать запятые комбинацией символов `"\,"`
`FieldDelim`	Символ, используемый для отделения значений полей друг от друга. По умолчанию `","` (запятая)
`Filter`	Строка условия фильтрации записей для набора данных. Подробное описание см. ниже
`RowDelim`	Символ, используемый для отделения записей друг от друга. По умолчанию "новая строка"
`Sort`	Задает условия сортировки записей. По умолчанию `""` (пустая строка), т. е. сортировка отсутствует. Подробное описание см. ниже
`TextQualifier`	Символ, в который заключается значение поля. Этот символ может быть полезен, если значение поля содержит пробелы, запятые и подобные им символы, используемые также в качестве служебных. По умолчанию двойная кавычка

Таблица 6.1 *(окончание)*

Параметр	Описание
`UseHeader`	Если `true` (значение по умолчанию), то первая строка таблицы считается списком имен и типов данных полей. Если `false`, первая строка таблицы считается данными, а поля получают названия по умолчанию: "Column1", "Column2" и т. д.

Вместо того чтобы задавать все эти параметры в тегах `<PARAM>`, вы можете получить к ним доступ, воспользовавшись одноименными свойствами элемента TDC:

```
datProducts.object.Filter = "ProductNumber > 3";
```

Не забываем в этом случае указать свойство `object`, т. к. мы ссылаемся не на свойство тега `<OBJECT>`, а на свойство самого элемента ActiveX.

Немного поговорим о параметре `UseHeader`. Я рекомендую всегда перечислять имена полей и их типы в первой строке таблицы. Она может иметь примерно такой вид:

```
ProductNumber:int,ProductName:text,ProductPrice:float
```

Здесь мы задали три поля: `ProductNumber` целочисленного типа, `ProductName` текстового и `ProductPrice` — число с плавающей точкой. Полный список типов полей таков:

- текстовый (`text`) — это тип по умолчанию;
- дата (`date`);
- логический (`boolean`) — yes или no, 1 или 0;
- целое число (`int`);
- число с плавающей точкой (`float`).

Хорошо. Мы поместили на страницу элемент TDC. А что дальше? Как отобразить на странице значение какого-либо поля или, другими словами, "привязать" элемент страницы к полю базы данных? Для этого почти все элементы страницы поддерживают несколько свойств, методов и событий, о которых мы сейчас будем говорить.

Привязка элементов страниц к данным

Для привязки к какому-либо полю базы данных большинство элементов страницы поддерживает набор атрибутов и свойств, перечисленных в табл. 6.2.

Таблица 6.2. Свойства и атрибуты привязки к данным

Атрибут	Свойство	Описание
DATAFLD	dataFld	Название поля базы данных, к которому привязан элемент страницы
DATAFORMATAS	dataFormatAs	Формат представления данных. Если задано значение "text", то данные будут отображаться "как есть", а если "HTML" — то с учетом HTML-форматирования (если оно есть). Доступно также значение "localized-text", задающее отображение с учетом национальных установок системы
DATAPAGESIZE	dataPageSize	Применяется только для таблиц. Задает, сколько записей будет одновременно отображаться в таблице
DATASRC	dataSrc	Задает элемент TDC-источник данных

Итак, для того чтобы привязать элемент страницы к данным, нужно задать соответствующие значения атрибутов DATASRC и DATAFLD (или свойств dataSrc и dataFld в скрипте).

```
<P ID="pProductName" DATASRC="#objData" DATAFLD="productname"></P>
```

Здесь мы привязали к полю базы данных productname элемент pProductName.

```
<INPUT TYPE="text" ID="txtProductName" DATASRC="#objData"
↪DATAFLD="productname">
```

А теперь к базе данных привязано уже текстовое поле. Правда, пользователь не сможет редактировать содержимое поля — это ограничение самого текстового формата данных.

Если вы хотите представить какую-то часть базы данных в виде таблицы, поместите в каждую ячейку элемент <DIV>, привяжите его к нужному полю базы данных, а привязку к элементу TDC задайте в теге <TABLE>. И при этом не забудьте поместить ячейки, где будут отображаться данные из базы, внутрь тега <TBODY>.

```
<TABLE DATASRC="#datProducts" DATAPAGESIZE="20">
<TBODY>
<TR><TD><DIV DATAFLD="divProductNumber"></DIV></TD></TR>
<TR><TD><DIV DATAFLD="divProductName"></DIV></TD></TR>
<TR><TD><DIV DATAFLD="divProductPrice"></DIV></TD></TR>
</TBODY>
</TABLE>
```

Так, например, может выглядеть прайс-лист, о котором мы говорили в начале этой главы.

Объект *recordset* и управление данными

Все элементы TDC имеют внутренний объект `recordset`, с помощью которого можно управлять данными: переходить от одной записи к другой. Получить к нему доступ можно, обратившись к свойству `recordset` объекта TDC.

```
datProducts.recordset
```

Объект `recordset` поддерживает ряд методов для перемещения по записям базы данных. Это методы `MovePrevious` и `MoveNext`, делающие текущей соответственно предыдущую и последующую запись, и `MoveFirst` и `MoveLast`, активизирующие самую первую и самую последнюю запись.

Свойство `AbsolutePosition` позволит вам выяснить, сколько записей содержится в базе данных.

Для доступа к значениям отдельных полей из кода скрипта вы можете использовать коллекцию `Fields`, содержащую все объекты полей `Field`, входящих в таблицу. Таким образом, вы можете извлечь значение из нужного поля, написав такой код:

```
price = datProducts.recordset.Fields("ProductPrice").Value;
```

Также объект `recordset` поддерживает методы для добавления и удаления записей, но мы рассмотрим их позже.

Пример страницы, привязанной к данным

Теперь рассмотрим небольшой пример страницы, привязанной к данным. Она будет отображать содержимое одной записи текстовой таблицы.

Сначала создадим саму текстовую таблицу. Пусть это будет прайс-лист; содержать он будет три поля: номер товара, его наименование и цена. Первая строка таблицы будет содержать имена полей. Наберите таблицу в Блокноте, разделяя значение полей запятыми, и сохраните под именем 6.1.csv. У вас должно получиться что-то, похожее на это:

```
ProductNumber,ProductName,ProductPrice
1,Веник,10
2,Супервеник,20
3,Метла,50
4,Совок,5
5,Очень хороший совок,25
```

Слишком большой и сложной таблицу делать не надо. Это просто пример, и он должен быть небольшим и понятным.

Теперь напишем код нашей страницы.

```
<HTML>
<HEAD>
<TITLE>Web + данные 1</TITLE>
</HEAD>
<BODY>
<OBJECT CLASSID="clsid:333C7BC4-460F-11D0-BC04-0080C7055A83"
ID="datProducts" WIDTH="0" HEIGHT="0">
<PARAM NAME="DataURL" VALUE="6.1.csv">
<PARAM NAME="UseHeader" VALUE="true">
</OBJECT>
```

Первым делом помещаем на страницу элемент TDC.

```
<TABLE WIDTH="100%">
```

Используем таблицу для оформления — не более.

```
<TR>
<TD><P>№№</P></TD>
<TD><P>Товар</P></TD>
<TD><P>Цена</P></TD>
</TR>
<TR>
<TD><DIV DATASRC="#datProducts" DATAFLD="ProductNumber"></DIV></TD>
<TD><DIV DATASRC="#datProducts" DATAFLD="ProductName"></DIV></TD>
<TD><DIV DATASRC="#datProducts" DATAFLD="ProductPrice"></DIV></TD>
```

Размещаем элементы, привязанные к данным. Повторю, что наша страница отображает содержимое только одной записи.

```
</TR>
</TABLE>
<TABLE WIDTH="100%">
<TR>
<TD><INPUT TYPE="button" ID="cmdFirst" VALUE="&lt;&lt;"
onclick="datProducts.recordset.MoveFirst();"></TD>
<TD><INPUT TYPE="button" ID="cmdPrevious" VALUE="&lt;"
onclick="datProducts.recordset.MovePrevious();"></TD>
<TD><INPUT TYPE="button" ID="cmdNext" VALUE="&gt;"
onclick="datProducts.recordset.MoveNext();"></TD>
<TD><INPUT TYPE="button" ID="cmdLast" VALUE="&gt;&gt;"
onclick="datProducts.recordset.MoveLast();"></TD>
```

А эти кнопки предназначены для перемещения по базе данных. Мы разместили их в другой таблице.

```
</TR>
</TABLE>
</BODY>
</HTML>
```

Сохраните этот код в файле под именем 6.1.htm и откройте его и Internet Explorer. Вы увидите нечто, похожее на рис. 6.1. Пощелкайте по кнопкам и посмотрите, что произойдет.

Рис. 6.1. Страница, отображающая одну запись текстовой таблицы

Это была простейшая страница, работающая с данными. Разумеется, практической пользы от нее никакой. Ну кому, скажите на милость, нужен прайс-лист, отображающий информацию только по одному товару!

Дополнительная поддержка привязки к данным

Как мы уже знаем, тег таблицы `<TABLE>` может иметь атрибут `DATAPAGESIZE`, задающий количество строк, одновременно отображающихся в таблице, иными словами, размер страницы таблицы. Объект таблицы поддерживает набор методов, предназначенных для постраничного "листания" таблицы. Это методы `previousPage` и `nextPage`, выводящие соответственно предыдущую и последующую страницу, и `firstPage` и `lastPage`, выводящие самую первую и самую последнюю страницу таблицы.

Кроме того, тег `<OBJECT>` поддерживает набор событий, специально предназначенный для работы с данными. Все они перечислены в табл. 6.3.

Таблица 6.3. *События тега* `<OBJECT>` *для поддержки работы с базами данных*

Событие	Описание
`ondataavailable`	Наступает всякий раз, когда набор данных готов предоставить новые данные

***Таблица 6.3** (окончание)*

Событие	Описание
ondatasetchanged	Наступает, когда набор данных изменяется, например, после задания условий фильтрации записей
ondatasetcomplete	Наступает после того, как набор данных передаст все доступные данные
onrowenter	Наступает, когда набор данных переходит на новую запись
onrowexit	Наступает перед тем, как набор данных "уйдет" с текущей записи на другую

Эти события можно использовать, скажем, для отображения на странице каких-либо данных, не хранящихся в таблице, а вычисляемых на основании других данных, которые хранятся в таблице. Например, мы можем рассчитать для нашего прайс-листа цены с учетом 10-процентной скидки, написав такой код.

```
<OBJECT CLASSID="clsid:333C7BC4-460F-11D0-BC04-0080C7055A83"
ID="datProducts" WIDTH="0" HEIGHT="0"
onrowenter="newPrice.innerText =
datProducts.recordset.Fields('ProductPrice').Value * 0.9;">
<PARAM NAME="DataURL" VALUE="6.1.csv">
<PARAM NAME="UseHeader" VALUE="true">
</OBJECT>
<P ID="newPrice"></P>
```

Фильтрация и сортировка данных

Элемент TDC предоставляет Web-программисту возможность фильтрации и сортировки данных, выводимых на Web-странице. При этом само содержимое текстовой таблицы не затрагивается — изменяется только ее представление.

Итак, поговорим о фильтрации. *Фильтрация* — это отбор записей базы данных по определенным критериям. Такими критериями могут быть равенство или неравенство поля (полей) какому-либо значению. Для записи подобных критериев используется тот же синтаксис, что и для условных выражений языка JavaScript. Единственное исключение составляют операторы сравнения — они перечислены в табл. 6.4.

Как вы знаете, элемент TDC предусматривает параметр `Filter`. Вы можете задать его при определении TDC в теге `<PARAM>` или получить к нему доступ через свойство `Filter`.

```
datProducts.object.Filter = "ProductPrice > 10 & ProductPrice <= 100";
```

Таблица 6.4. *Операторы сравнения, используемые в выражениях фильтрации*

Оператор	Описание	Оператор	Описание
<	Меньше	>=	Больше или равно
>	Больше	<>	Не равно
=	Равно	&	Логическое И
<=	Меньше или равно	|	Логическое ИЛИ

Если нужно вообще отключить фильтрацию, присвойте свойству `Filter` пустую строку "".

Сортировка задается с помощью параметра `Sort`. Вы точно также можете или задать его значение в теге `<PARAM>`, или воспользоваться одноименным свойством.

Условие сортировки представляет собой список разделенных запятыми имен полей. Сначала сортировка выполняется по первому в списке полю, в случае, если значения первого поля всех записей будут одинаковыми — по второму полю и т. д. По умолчанию сортировка выполняется по возрастанию; если нужно отсортировать записи по убыванию, поместите перед именем нужного поля знак "-":

```
datProducts.object.Sort = "ProductName,-ProductPrice";
```

Здесь мы отсортировали записи сначала по имени товара, а потом по цене товара по убыванию.

Однако возникает небольшая проблема: результаты сортировки и фильтрации по заданным нами критериям мы увидим не сразу. Точнее, мы их совсем не увидим, пока набор данных не будет перезагружен. А выполняется это методом `Reset` элемента TDC.

```
datProducts.object.Filter = "ProductPrice > 10 & ProductPrice <= 100";
datProducts.object.Sort = "ProductName,-ProductPrice";
datProducts.object.Reset();
```

Пример прайс-листа с возможностью фильтрации и сортировки

Теперь давайте создадим настоящий прайс-лист, предусматривающий возможность фильтрации и сортировки записей. Он будет иметь вид таблицы, отображающей данные о товарах. Для этого нам нужно будет существенно увеличить размер нашей текстовой базы данных, хотя бы до тридцати запи-

сей, чтобы заданная нами фильтрация была, что называется, видна невооруженным глазом. Вы можете просто несколько раз скопировать одни и те же строчки под другими номерами и с другими ценами, как это сделал я. Сохраните новую базу данных под именем 6.2.csv.

Теперь напишем код нашей страницы.

```
<HTML>
<HEAD>
<TITLE>Web + данные 2</TITLE>
<SCRIPT>
function setFilter() {
  var s;
  s = tabProducts.dataSrc;
  tabProducts.dataSrc = "";
  datProducts.object.Filter = txtFilter.value;
  datProducts.object.Reset();
  tabProducts.dataSrc = s; }
```

Эта функция выполняется при нажатии на кнопку смены фильтра. Как видите, сначала свойство `dataSrc` таблицы сбрасывается, т. е. таблицы открепляется от набора данных. Это сделано потому, что таблица не сможет динамически перерисоваться после того, как мы изменим фильтр набора данных. Разумеется, после изменения фильтра и обновления набора данных мы восстанавливаем значение свойства `dataSrc` таблицы, чтобы в ней отобразился результат фильтрации.

```
function setSort() {
  var s;
  s = tabProducts.dataSrc;
  tabProducts.dataSrc = "";
  datProducts.object.Sort = txtSort.value;
  datProducts.object.Reset();
  tabProducts.dataSrc = s; }
```

Эта функция вызывается при нажатии на кнопку смены сортировки. Работает она точно так же, как и предыдущая функция.

```
</SCRIPT>
</HEAD>
<BODY>
<OBJECT CLASSID="clsid:333C7BC4-460F-11D0-BC04-0080C7055A83"
↳ID="datProducts" WIDTH="0" HEIGHT="0">
<PARAM NAME="DataURL" VALUE="6.2.csv">
<PARAM NAME="UseHeader" VALUE="true">
<PARAM NAME="Sort" VALUE="ProductName">
</OBJECT>
```

Это определение объекта набора данных. Заметьте, что здесь мы задали начальное значение сортировки.

```
<H1>Прайс-лист фирмы &quot;Веник &amp; совок&quot;</H1>
<TABLE WIDTH="100%" DATASRC="#datProducts" ID="tabProducts"
↳DATAPAGESIZE="10">
```

Таблица, привязанная к данным. Данные будут выводиться страницами по десять записей.

```
<THEAD>
<TR>
<TD WIDTH="10%"><DIV>№№</DIV></TD>
<TD WIDTH="70%"><DIV>Наименование</DIV></TD>
<TD WIDTH="20%"><DIV>Цена</DIV></TD>
</TR>
</THEAD>
```

Секция заголовка нашей таблицы.

```
<TBODY>
<TR>
<TD><DIV DATAFLD="ProductNumber"></DIV></TD>
<TD><DIV DATAFLD="ProductName"></DIV></TD>
<TD><DIV DATAFLD="ProductPrice"></DIV></TD>
</TR>
</TBODY>
```

Секция тела таблицы, привязанной к данным. Не забывайте, что строки, в которых отображаются данные, должны помещаться именно здесь — внутри тега <TBODY>.

```
</TABLE>
```

Далее идут элементы управления, с помощью которых задаются критерии фильтрации и сортировки и осуществляется "листание" таблицы. Они помещены в таблицу.

```
<TABLE WIDTH="100%" VALIGN="middle">
<TR>
<TD>
<LEGEND FOR="txtFilter">Критерий фильтрации</LEGEND>
<INPUT TYPE="text" ID="txtFilter">
<INPUT TYPE="button" ID="cmdFilter" VALUE="Применить"
↳onclick="setFilter();">
</TD>
<TD>
<LEGEND FOR="txtSort">Критерий сортировки</LEGEND>
<INPUT TYPE="text" ID="txtSort">
```

```
<INPUT TYPE="button" ID="cmdSort" VALUE="Применить" onclick="setSort();">
</TD>
<TD>
<INPUT TYPE="button" ID="cmdPageUp" VALUE="&lt;"
onclick="tabProducts.previousPage();">
</TD>
<TD>
<INPUT TYPE="button" ID="cmdPageDown" VALUE="&gt;"
onclick="tabProducts.nextPage();">
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

Вот и все. Сохраните код в файле 6.2.htm и откройте его в Internet Explorer. Отметьте, что данные, загруженные в таблицу, уже отсортированы по имени товара. Попытайтесь изменить порядок сортировки и критерии фильтрации. Посмотрите на рис. 6.2 и попробуйте достичь таких же результатов.

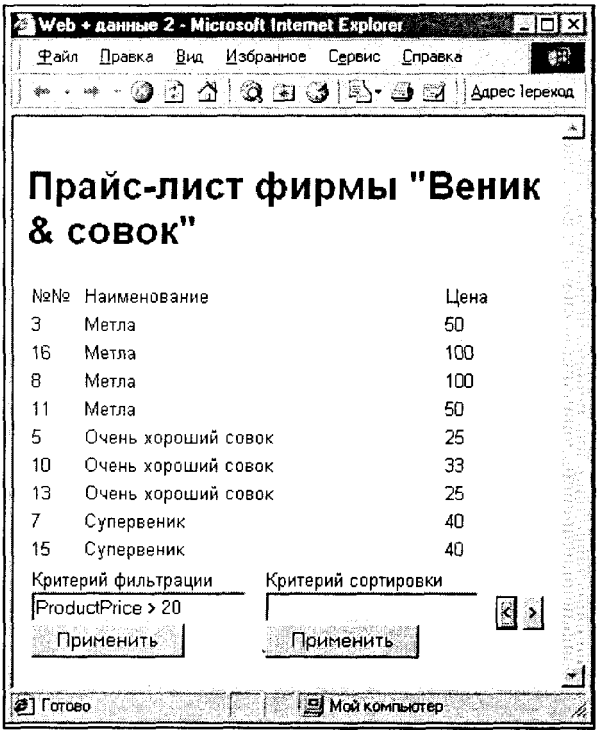


Рис. 6.2. Прайс-лист

Текстовые таблицы, рассмотренные нами, идеальны для отображения небольших списков, содержимое которых не должно изменяться. Если же вы

хотите дать возможность пользователю не только просматривать, но и изменять данные (добавлять, редактировать и удалять записи), следует воспользоваться элементом *RDS* (Remote Data Service — удаленный сервис данных). Элемент RDS предоставляет также возможность использовать для составления запросов к данным язык SQL. Важнейший его недостаток: он может работать только с Web-сервером, отчего и получил свое название — Remote Data Service.

Удаленный доступ к базам данных

Здесь мы рассмотрим использование элемента RDS для полноценного доступа к базам данных. Но сначала поговорим о процессоре данных, поставляемом в составе операционной системы Windows — ODBC.

Как работать с ODBC

ODBC — это стандартный процессор данных, встроенный в систему. Он используется многими программными средствами для доступа к данным, хранимым в файлах на жестком диске клиентского компьютера или на сервере баз данных. ODBC поддерживает множество форматов баз данных за счет того, что разделен на *ядро*, обеспечивающее базовую функциональность, и набор *драйверов*, служащих для непосредственного доступа к файлам или серверам баз данных. В частности, ODBC поддерживает файлы Microsoft Access, Microsoft Excel, Borland Paradox, Borland dBase и текстовые файлы.

Работать с ODBC довольно просто; самое сложное — правильно настроить соединение с базой данных на компьютере, где работает Web-сервер. Сейчас мы рассмотрим, как это делается.

Откройте Панель управления, выбрав в меню **Пуск** пункт **Настройка** и в появившемся на экране подменю — пункт **Панель управления**. Панель управления служит для задания системных настроек: там задаются параметры сети, видеоадаптера, принтеров, мультимедийных средств и т. п. Найдите в окне Панели управления значок **Источники данных ODBC (32)** и дважды щелкните по нему. На экране появится окно **Администратор источников данных ODBC**; переключитесь на вкладку **Системный DSN**. И давайте поговорим о том, что мы только что сделали.

Одно из отличий RDS от TDC состоит в том, что RDS требует для своей работы *источник данных ODBC* (DSN). Он содержит в себе сведения о местонахождении файлов базы данных и их формате; также имеет уникальное имя, по которому Web-сервер (и любая другая программа) может его идентифицировать. В параметрах элемента RDS указывается как раз это имя; зная его, RDS быстро доберется до данных. В нашем примере мы использу-

ем системный источник данных, который может быть применен любым пользователем и любой программой.

Элемент TDC грешил тем, что мог запрашивать только одну-единственную таблицу. RDS может полноценно работать с очень сложными многотабличными базами данных, используя язык описания запросов SQL. И, конечно, он может изменять данные, записанные в таблицах, чего не позволял делать TDC.

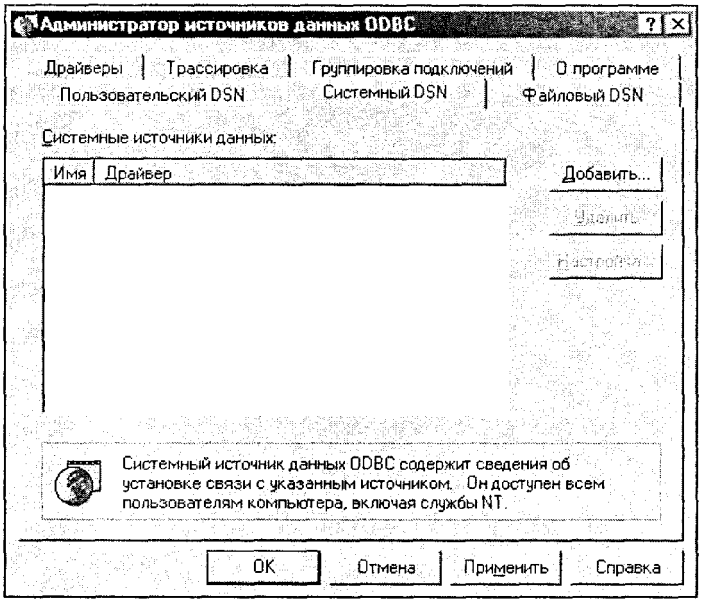


Рис. 6.3. Диалоговое окно **Администратор источников данных ODBC**

Итак, взгляните на рис. 6.3. На нем показано окно задания системных источников данных. Сейчас мы зададим новый системный DSN.

Для простоты и экономии времени мы не будем создавать новую базу данных, а используем ту же самую текстовую таблицу 6.2.csv, что мы создали ранее. ODBC включает в стандартной поставке драйвер текстовых таблиц. Но, т. к. ODBC всегда предполагает, что база данных состоит из множества таблиц, а наш файл содержит только одну, то в качестве имени базы данных мы зададим каталог, в котором находится файл 6.2.csv. Таким образом, каталог станет своего рода "базой данных", содержащей "таблицы", находящиеся в отдельных текстовых файлах.

Щелкните кнопку **Добавить**. На экране появится новое диалоговое окно **Создание нового источника данных**, показанное на рис. 6.4. Выберите в списке драйверов пункт **Microsoft Text Driver (*.txt, *.csv)** и нажмите кнопку **Готово**. На экране появится новое диалоговое окно **Установка текстового драйве-**

ра ODBC, показанное на рис. 6.5. Щелкните кнопку **Параметры>>**, чтобы получить доступ к скрытой части окна.

Введите в поле ввода **Имя источника данных** какое-либо имя, которое будет впоследствии использоваться для доступа к нему, например "6.2". В поле

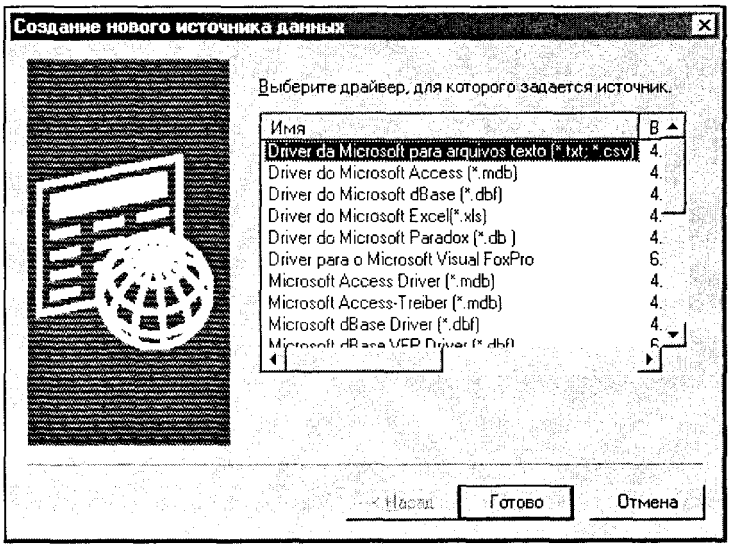


Рис. 6.4. Диалоговое окно **Создание нового источника данных**

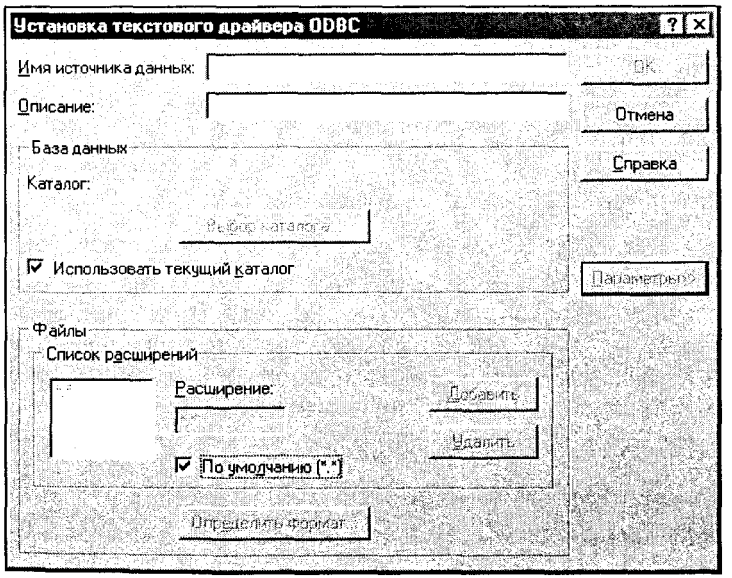


Рис. 6.5. Диалоговое окно **Установка текстового драйвера ODBC**

Описание вы можете ввести осмысленное описание создаваемого источника данных, но это не обязательно. Теперь отключите флажок **Использовать текущий каталог** и нажмите кнопку **Выбор каталога**. На экране появится стандартное диалоговое окно Windows **Выбор каталога**. Выберите в нем каталог, в котором находится созданная нами ранее текстовая таблица 6.2.csv, и нажмите **OK**. И еще два раза нажмите кнопку **OK**, чтобы закрыть окна **Установка текстового драйвера ODBC** и **Администратор источников данных ODBC**. Окно Панели управления также можно закрыть — оно нам больше не понадобится.

Таковы основные шаги создания нового источника данных ODBC. Если вы хотите получить доступ к другой, нетекстовой базе данных, например файлу Access или базе, хранимой на сервере Interbase, вам нужно будет воспользоваться соответствующим драйвером. Каждый драйвер баз данных, установленный в системе, будет иметь свои настройки; в случае затруднений обращайтесь к интерактивной справке, поставляемой с ним, или к печатной документации, если таковая имеется.

Теперь пора познакомиться поближе с элементом RDS.

Элемент RDS

Элемент RDS в основных чертах подобен своему более непритязательному собрату TDC. Скажем, код, вставляющий элемент RDS, выглядит очень знакомо. Судите сами:

```
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
↳ID="{Имя объекта}" HEIGHT="0" WIDTH="0">
<PARAM NAME="Server" VALUE="{Интернет-адрес сервера}">
<PARAM NAME="Connect" VALUE="{Строка параметров ODBC}">
<PARAM NAME="SQL" VALUE="{Выражение на языке SQL}">
</OBJECT>
```

Таков, в общих чертах, HTML-код элемента RDS. Кратко рассмотрим его параметры.

Прежде всего, нужно задать интернет-адрес Web-сервера, на котором находятся нужные данные и системный источник данных. Разумеется, на сервере должен быть установлен ODBC (если он работает под управлением Windows 95 или Windows NT 4.0 или более новых их версий, то ODBC там уже установлен). В противном случае, ODBC нужно установить отдельно.

Строка параметров ODBC имеет следующий формат:

```
dsn={Имя источника данных};uid={Имя пользователя};
↳pwd={Пароль пользователя}
```

Имя источника данных мы ввели при его создании, имя пользователя важно только в том случае, если производится обращение к серверу данных. В ос-

тальных случаях задайте имя "guest" (без кавычек). То же самое и с паролем: он нужен при обращении к серверу баз данных, в противном случае оставьте вместо него пустую строку.

Выражение на языке SQL служит для описания запроса данных из базы. В таком выражении можно задать поля, критерии фильтрации и параметры сортировки данных. Также язык SQL позволяет задавать так называемые вычисляемые поля, т. е. поля, значения которых не хранятся в таблице, а получаются в результате вычисления по некой формуле, использующей значения существующих полей таблиц. Используя вычисляемые поля, можно производить несложную обработку данных.

Описание языка SQL выходит за рамки этой книги.

Элемент RDS поддерживает и другие параметры, рассмотренные в табл. 6.5.

Таблица 6.5. *Параметры элемента RDS*

Параметр	Описание
Connect	Задает строку параметров ODBC
FilterColumn	Поле результирующей таблицы, по которому производится фильтрация записей
FilterCriterion	Критерий фильтрации записей результирующей таблицы. Представляет собой логическое выражение, состоящее из операторов сравнения, перечисленных в табл. 6.4, в текстовом виде
FilterValue	Значение, по которому будут фильтроваться записи результирующей таблицы
Server	Интернет-адрес Web-сервера, где определен источник данных
SortColumn	Поле результирующей таблицы, по которому производится сортировка записей
SortDirection	True, если сортировка производится по возрастанию, false — если по убыванию
SQL	Выражение на языке SQL, задающее, какие данные и в каком виде нужно представить пользователю. Позволяет задать имена таблиц, связи между ними, критерии фильтрации и сортировки записей

Как видите, возможности фильтрации и сортировки здесь значительно беднее, чем у элемента RDS. Но это вполне оправданно, ведь ODBC позволяет запрашивать данные с помощью выражений на языке SQL, которыми можно задать какие угодно сложные критерии фильтрации и сортировки записей.

Как я уже говорил, элемент RDS предоставляет возможность изменения данных. Давайте рассмотрим, какие же дополнительные возможности предлагают для этого элементы Web-страницы.

Код, вставляющий на Web-страницу элемент RDS, может быть таким:

```
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
ID="datProducts" HEIGHT="0" WIDTH="0">
<PARAM NAME="Server" VALUE="http://www.someserver.ru">
<PARAM NAME="Connect" VALUE="dsn=price-list;uid=guest;pwd=">
<PARAM NAME="SQL" VALUE="select number,name,price from pricelist when
price<100 order by name">
</OBJECT>
```

В этом примере мы задали SQL-выражение, извлекающее из записей таблицы `pricelist` базы данных поля `number`, `name` и `price`. При этом выбираются только товары с ценой (`price`), меньшей `100`; полученные данные сортируются по полю `name`. Как видите, недостаток встроенных средств фильтрации и сортировки элемента RDS с лихвой компенсируется мощью языка SQL.

Также обратите внимание, что мы задали сервер `http://www.someserver.ru`, на котором определен источник данных `price-list`. Для доступа к нему используется имя пользователя `guest` ("гостевой доступ" с минимальными правами) и пустой пароль.

Дополнительная поддержка изменения данных

Для поддержки изменения записей таблиц баз данных объект `recordset` поддерживает два новых метода: `AddNew` и `Delete`. Ниже мы их рассмотрим.

Первый метод добавляет к таблице новую пустую запись и делает ее текущей. После того как пользователь введет данные в поля этой записи и перейдет на другую запись (или добавит новую), производится сохранение введенных данных в таблице. Точно так же производится сохранение всех изменений, внесенных пользователем в уже имеющуюся запись; пока пользователь не перейдет на другую запись, все внесенные им изменения не сохраняются в таблице — имейте это в виду.

Второй метод позволяет удалить текущую запись.

Объект `<OBJECT>`, помимо перечисленных в табл. 6.3 событий, поддерживает еще два. Первое — `onrowsdelete` — наступает, когда запись или записи удаляются из набора данных. Второе — `onrowsinserted` — наступает сразу после того, как новые записи будут добавлены в набор данных. Эти события можно использовать для отображения данных о количестве записей в наборе данных; при удалении и добавлении записей это количество будет изменяться.

Элементы страницы, привязанные к данным, также поддерживают ряд событий, наступающих при изменении значений какого-либо поля (или полей). Они перечислены в табл. 6.6.

Таблица 6.6. *События элементов страниц, наступающие при изменении данных текущей записи*

Событие	Описание
`onafterupdate`	Наступает после того, как новые данные, введенные пользователем, помещаются в соответствующие поля таблицы
`onbeforeupdate`	Наступает перед тем, как новые данные, введенные пользователем, будут помещены в соответствующие поля таблицы
`onerrorupdate`	Наступает при любой ошибке сохранения введенных пользователем данных. По умолчанию на экран выводится предупреждение; отменив поведение по умолчанию, Web-программист сможет установить какие-либо специальные действия при ошибке сохранения данных.

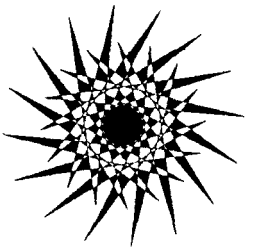
Имейте в виду, однако, следующее. Не каждый набор данных позволяет их изменять. Дело может быть в особенностях формата хранения данных, привилегиях, данных конкретному пользователю, или особенностях самого набора данных, возвращенного пользователю в результате выполнения SQL-запроса. Так, запросы, объединяющие данные из двух и более таблиц, не предоставляют такой возможности. Аналогично, пользователь не сможет удалять записи из текстовой таблицы, хотя сможет добавлять и изменять их.

К несчастью, для того чтобы проверить на практике работу элемента RDS, нужно иметь установленный на компьютере Web-сервер. Поэтому мы не будем рассматривать пример Web-страницы, использующей этот метод доступа к данным.

Хотите знать больше?

Здесь мы познакомились с возможностью доступа к базам данных из Web-страниц, предлагаемой Internet Explorer начиная с версии 4.0. Но его специфические возможности на этом не ограничиваются. Internet Explorer, кроме всего прочего, также позволяет разработчикам улучшить свои страницы с помощью так называемых фильтров и преобразований. Их описание будет приведено в последней главе этой книги.

ГЛАВА 7



Фильтры и преобразования

Помимо всех прочих "вкусностей", что припас Internet Explorer Web-программистам, есть еще так называемые фильтры и преобразования. Это мультимедийные эффекты, которые можно применить к любому элементу Web-страницы. *Фильтры* — это статические эффекты, т. е. они не изменяются с течением времени: затенение, освещение, изменение цветов, смазанность и т. п. *Преобразования* называются еще динамическими эффектами; к ним относятся всевозможные "всплывания" и "выплывания", "наползания" и "уползания" одного элемента на другой. Это действительно красиво, если, конечно, использовано умеренно и к месту.

Создание фильтров и преобразований

Фильтры и преобразования можно задавать в самом HTML-коде, а именно, в определении стиля элемента. Для этого используется атрибут `filter` и следующий формат:

```
filter: {Имя фильтра}({Свойства фильтра})
```

Элемент страницы должен иметь заданную ширину и высоту; это можно сделать с помощью атрибутов стиля `width` и `height`.

Например, давайте зададим для рисунка фильтр "размытость".

```
<IMG SRC="some.gif" ID="imgSome" STYLE="height: 100; width: 100;
↳filter: MotionBlur(strength=50)">
```

Этот синтаксис используется в Internet Explorer более ранних версий, чем 5.5. В Internet Explorer 5.5 предлагается альтернативный синтаксис:

```
filter: progid:DXImageTransform.Microsoft.{Имя фильтра}
↳({Свойства фильтра})
```

Стало быть, наш пример будет выглядеть так:

```
<IMG SRC="some.gif" ID="imgSome" STYLE="height: 100; width: 100; filter:
↳progid:DXImageTransform.Microsoft.MotionBlur(strength=50)">
```

Внимание!

Прежний синтаксис определения фильтра в Internet Explorer версии 5.5 и более поздних может не работать!

Для доступа к фильтрам из скриптов применяется коллекция `filters`. Чтобы получить к ней доступ, обратитесь к свойству `filters` элемента страницы.

```
iSome.filters.item(0).enabled = false;
```

Для создания преобразований используется точно такой же синтаксис. Давайте применим к нашему рисунку преобразование "плавное скрытие".

```
<IMG SRC="some.gif" ID="imgSome" STYLE="height: 100; width: 100;
↳filter: Fade(duration=4)">
```

Однако, для того чтобы запустить преобразование, нам нужно использовать скрипт. Например, можно сделать так:

```
<IMG SRC="some.gif" ID="imgSome" STYLE="height: 100; width: 100;
↳filter: Fade(duration=4)" onclick="startTrans();">
function startTrans() {
  imgSome.filters.item(0).Apply();
  imgSome.filters.item(0).visibility = false;
  imgSome.filters.item(0).Play(); }
```

Метод `Apply` "замораживает" преобразование; после этого вы можете творить с элементом страницы все, что захотите. В нашем примере мы просто скрыли его, присвоив значение `false` свойству `visibility`. Метод `Play` непосредственно запускает преобразование — наш рисунок будет скрыт при щелчке на нем.

Вы также можете использовать свойство `src` для того, чтобы поместить в рисунок новое изображение. В этом случае мы получим очень красивый "переход" из одного изображения в другое.

Все элементы страницы поддерживают событие `onfilterchange`, наступающее при изменении состояния или завершении работы фильтра или преобразования. Вы можете использовать это событие, например, для того, чтобы присвоить элементу страницы другой фильтр или иное преобразование.

Фильтры

Все доступные в Internet Explorer фильтры перечислены в табл. 7.1.

Таблица 7.1. *Фильтры*

Фильтр	Описание
`Alpha`	Делает элемент страницы прозрачным
`AlphaImageLoader`	Отображает графическое изображение внутри элемента страницы между его фоном и содержимым
`BasicImage`	Делает элемент страницы черно-белым, как бы просвеченным рентгеновскими лучами, вращает его. Отдельно можно задавать угол поворота, степень "просвеченности" и т. п.
`Blur`	Делает элемент страницы размытым, "туманным" (blur)
`Chroma`	Делает определенный цвет элемента прозрачным
`Compositor`	Объединяет цвета двух элементов страницы и выводит результат объединения
`DropShadow`	Заставляет элемент страницы "отбросить" тень, причем тень отображается отдельно от самого элемента
`Emboss`	Отображает элемент страницы выпуклым
`Engrave`	Отображает элемент страницы вдавленным
`Glow`	Отображает элемент страницы так, что он кажется "тлеющим"
`Gradient`	Градиентно закрашивает элемент страницы
`Light`	Отображает элемент страницы так, что он кажется освещенным
`MaskFilter`	Отображает прозрачный цвет элемента страницы цветом, указанным в свойстве `color`, а все непрозрачные цвета делает прозрачными
`Matrix`	Изменяет размеры элемента страницы, поворачивает или инвертирует его, используя матричные преобразования
`MotionBlur`	Делает элемент страницы размытым, словно быстро движущимся куда-то
`Pixelate`	Отображает элемент страницы отдельными пикселами
`Shadow`	Заставляет элемент страницы "отбросить" тень
`Wave`	Создает волнистое искажение элемента страницы

Преобразования

Все доступные в Internet Explorer преобразования перечислены в табл. 7.2.

Таблица 7.2. *Преобразования*

Преобразование	Описание
Barn	Создает эффект "открывающейся и закрывающейся двери"
BlendTrans	Простейшее преобразование, плавно заменяющее старое содержимое на новое. Сохранено для совместимости с Internet Explorer 4.0
Blinds	Создает эффект "открывающихся и закрывающихся жалюзи"
CheckerBoard	Создает эффект "шахматной доски"
Fade	Создает эффект наплыва (старое содержимое плавно пропадает, а новое одновременно так же плавно появляется)
GradientWipe	Новое содержимое элемента страницы "наползает" на старое, причем граница выглядит как градиентная цветная полоса
Inset	Новое содержимое элемента страницы диагонально "наползает" на старое
Iris	Создает эффект ирисовой диафрагмы
Pixelate	Старое содержимое элемента страницы "рассыпается" на отдельные пикселы и пропадает, а новое так же "собирается" из отдельных пикселов. Не путайте с фильтром Pixelate, описанным в табл. 7.1
RadialWipe	Новое содержимое элемента страницы радиально "наползает" на старое
RandomBars	Старое содержимое элемента страницы "рассыпается" на отдельные линии и пропадает, а новое так же "собирается" из отдельных линий
RandomDissolve	Новое содержимое элемента страницы поточечно "проявляется" на месте старого
RevealTrans	Простейшее преобразование, плавно заменяющее старое содержимое на новое, используя какой-либо эффект. Сохранено для совместимости с Internet Explorer 4.0
Slide	Старое содержимое элемента страницы "сдвигается" в сторону, открывая под собой новое
Spiral	Новое содержимое элемента страницы спирально закрашивает старое
Stretch	Новое содержимое элемента страницы "растягивается", заменяя собой старое

Таблица 7.2 (окончание)

Преобразование	Описание
Strips	Новое содержимое элемента страницы диагонально отдельными полосками "наползает" на старое
Wheel	Новое содержимое элемента страницы посекторно "наползает" на старое
Zigzag	Новое содержимое элемента страницы зигзагообразно отдельными полосками "наползает" на старое

Свойства и методы

Фильтры и преобразования поддерживают набор свойств и методов, предназначенных для управления их параметрами. Краткое описание всех этих свойств приведено в табл. 7.3, методов — в табл. 7.4; там также указаны фильтры и преобразования, поддерживающие то или иное свойство или метод. Для получения более полной информации обращайтесь к приложению 3.

Таблица 7.3. Свойства фильтров и преобразований

Свойство	Фильтр или преобразование	Описание
Add	MotionBlur, Wave	Если true, исходный элемент страницы перекрывает результат фильтрации. Если false (значение по умолчанию), результат фильтрации перекрывает исходный элемент страницы
Bands	Blinds, Slide	Количество полосок
Bias	Emboss, Engrave	Процентное значение, добавляемое к цвету элемента
Color	Chroma, DropShadow, Glow, MaskFilter, Shadow	Цвет
Direction	MotionBlur, Shadow	Направление. Задается в градусах и должно быть кратно 45
Direction	Blinds, CheckerBoard	Направление. Может быть одним из четырех значений: "up", "down", "right" и "left"

Таблица 7.3 (продолжение)

Свойство	Фильтр или преобразование	Описание
Duration	Все преобразования	Продолжительность преобразования в секундах
Dx, Dy	Matrix	Значения fDx и fDy матричных преобразований
Enabled	Все	Если true (значение по умолчанию), то фильтр или преобразование применяется к элементу страницы
EndColor, EndColorStr	Gradient	Конечный цвет градиентной закраски
FilterType	Matrix	Задает тип пикселов нового содержимого: "bilinear" (значение по умолчанию) или "nearest neighbor"
FinishOpacity	Alpha	Конечный уровень градиентной прозрачности; может быть от 0 (полная прозрачность; значение по умолчанию) до 100 (полная непрозрачность)
FinishX, FinishY	Alpha	Горизонтальная и вертикальная координаты позиции, в которой заканчивается область градиентной прозрачности
Freq	Wave	Количество "волн"
Function	Compositor	Функция преобразования. Подробнее см. приложение 3
GradientSize	GradientWipe	Часть площади элемента страницы, покрываемой градиентной полосой. Может быть от 0.0 до 1.0
GradientType	Gradient	Если 1 (значение по умолчанию), то градиентная заливка располагается по горизонтали, если 0 — по вертикали
GrayScale	BasicImage	Если 1, то элемент страницы отображается черно-белым, если 0 (значение по умолчанию) — цветным
GridSizeX, GridSizeY	Spiral, Zigzag	Количество полосок по горизонтали или вертикали. Может быть от 1 до 100
Invert	BasicImage	Если 1, то элемент страницы отображается с инвертированными цветами, если 0 (значение по умолчанию) — как обычно

Таблица 7.3 (продолжение)

Свойство	Фильтр или преобразование	Описание
IrisStyle	Iris	Форма "лепестков" ирисовой диафрагмы. Подробнее см. приложение 3
LightStrength	Wave	Постоянство окраски "волн". Может быть от 0 до 100
M11, M12, M21, M22	Matrix	Значения fM11, fM12, fM21 и fM22 матричных преобразований
MakeShadow	Blur	Если true, то элемент страницы будет отображаться как тень, если false (значение по умолчанию) — как обычно
Mask	BasicImage	Если 1, то прозрачный цвет элемента страницы будет заменен значением свойства MaskColor, если 0 (значение по умолчанию) — не будет
MaskColor	BasicImage	Цвет, на который будет заменен прозрачный цвет элемента страницы
MaxSquare	Pixelate	Максимальный размер пиксела
Mirror	BasicImage	Если 1, то элемент страницы будет отображен зеркально, если 0 (значение по умолчанию) — как обычно
Motion	Barn, Iris	Если "out" (значение по умолчанию), движение происходит из центра к границам, если "in" — от границ к центру
Motion	Strip	Угол, из которого появляется новое содержимое: "leftdown", "leftup", "rightdown" или "rightup"
Motion	GradientWipe	Если "forward" (значение по умолчанию), движение производится согласно значению свойства WipeStyle, если "reverse" — в обратном
OffX, OffY	DropShadow	Горизонтальное и вертикальное смещения тени
Opacity	Alpha	Начальный уровень градиентной прозрачности; может быть от 0 (полная прозрачность; значение по умолчанию) до 100 (полная непрозрачность)
Opacity	BasicImage	Уровень прозрачности элемента страницы. Может быть от 0.0 до 1.0

Таблица 7.3 (продолжение)

Свойство	Фильтр или преобразование	Описание
Orientation	Barn, RandomBars	Если "horizontal", преобразование происходит по горизонтали, если "vertical" — по вертикали
Overlap	Fade	Время, относительно общей продолжительности преобразования, когда и старое, и новое содержимое элемента страницы отображаются одновременно. Может быть от 0.0 до 1.0
Percent	Все преобразования	Процент выполнения преобразования. Может быть от 0 (преобразование еще не начиналось) до 100 (преобразование закончено)
Phase	Wave	Фаза "волн". Может быть от 0 до 100
PixelRadius	Blur	Размер области "размытия". Может быть от 1.0 до 100.0
Positive	DropShadow	Если true (значение по умолчанию), то тень создается из прозрачных пикселов элемента страницы, если false — из непрозрачных ("негативная" тень)
Rotation	BasicImage	Задает поворот элемента страницы. Доступны четыре значения: 0 (значение по умолчанию) — нет поворота, 1 — на 90°, 2 — на 180° и 3 — на 270°
ShadowOpacity	Blur	Прозрачность тени. Может быть от 0.0 (полностью прозрачная) до 1.0 (полностью непрозрачная)
SizingMethod	AlphaImageLoader	Способ размещения изображения в границах элемента страницы. Доступно три значения: "crop" — обрезание изображения, "image" (значение по умолчанию) — уменьшение или увеличение самого элемента страницы и "scale" — уменьшение или увеличение изображения
SizingMethod	Matrix	Способ размещения нового изображения в границах элемента страницы. Доступно два значения: "clip to original" (значение по умолчанию) — обрезание изображения и "auto expand" — уменьшение или увеличение изображения

Таблица 7.3 *(продолжение)*

Свойство	Фильтр или преобразование	Описание
SlideStyle	Slide	Способ замещения содержимого элемента страницы. Доступны три значения: "HIDE" (значение по умолчанию) — скрытие, "PUSH" — выталкивание и "SWAP" — замена
Spokes	Wheel	Количество секторов. Может быть от 2 до 20
SquaresX, SquaresY	CheckerBoard	Количество рядов по горизонтали и вертикали
src	AlphaImageLoader	Интернет-адрес файла изображения
StartColor, StartColorStr	Gradient	Начальный цвет градиентной закраски
StartX, StartY	Alpha	Горизонтальная и вертикальная координаты позиции, в которой начинается область градиентной прозрачности
status	Все преобразования	Возвращает состояние выполнения преобразования: 0, если оно было остановлено, 1, если оно было применено, 2, если оно выполняется
Strength	MotionBlur, Glow, Wave	Дистанция в пикселах, на которой выполняется преобразование
StretchStyle	Stretch	Способ замещения содержимого элемента страницы. Доступны три значения: "HIDE" — скрытие, "PUSH" — выталкивание и "SPIN" (значение по умолчанию) — замена
Style	Alpha	Параметры градиентной прозрачности: 0 (значение по умолчанию) — нет градиента, 1 — линейный градиент, 2 — круговой, 3 — прямоугольный
Transition	RevealTrans	Эффект преобразования. Подробнее см. приложение 3
WipeStyle	RadialWipe	Способ замещения содержимого элемента страницы. Доступны три значения: "CLOCK" (значение по умолчанию) — вращение вокруг центра элемента страницы по часовой стрелке, "WEDGE" — вращение сразу в обе стороны и "RADIAL" — радиальное вращение

Таблица 7.3 (окончание)

Свойство	Фильтр или преобразование	Описание
WipeStyle	GradientWipe	Если 0 (значение по умолчанию), движение происходит по горизонтали, если 1 — по вертикали
XRay	BasicImage	Если 1, то элемент страницы будет отображен "просвеченным рентгеновскими лучами", если 0 (значение по умолчанию) — как обычно

Таблица 7.4. Методы фильтров и преобразований

Метод	Фильтр или преобразование	Описание
AddAmbient({Красный}, {Зеленый},{Синий}, {Интенсивность})	Light	Добавляет источник рассеянного света с заданными цветовыми параметрами
AddCone({X1},{Y1},{Z1}, {X2},{Y2},{Красный}, {Зеленый},{Синий}, {Интенсивность},{Угол})	Light	Добавляет источник направленного света с заданными цветовыми параметрами. X1, Y1 и Z1 задают координаты источника, а X2 и Y2 — точки, куда падает свет
AddPoint({X},{Y},{Z}, {Красный},{Зеленый}, {Синий}, {Интенсивность})	Light	Добавляет источник направленного света с заданными цветовыми параметрами. X, Y и Z задают координаты источника
Apply()	Все преобразования	"Замораживает" элемент страницы, после чего вы можете делать с ним все, что хотите. Для того чтобы запустить преобразование, используйте метод Play
ChangeColor({№}, {Красный},{Зеленый}, {Синий}, 1|0)	Light	Изменяет цвет источника света с заданным номером. Последний параметр задает абсолютное (1 или любое ненулевое значение) или относительное (0) изменение цвета

Таблица 7.4 (окончание)

Метод	Фильтр или преобразование	Описание
`ChangeStrength({№}, {Интенсивность}, 1|0)`	`Light`	Изменяет интенсивность источника света с заданным номером. Последний параметр задает абсолютное (`1` или любое ненулевое значение) или относительное (`0`) изменение интенсивности
`Clear()`	`Light`	Удаляет все источники света
`MoveLight({№},{X},{Y}, {Z},1|0)`	`Light`	Перемещает источник света с заданным номером. Последний параметр задает абсолютное (`1` или любое ненулевое значение) или относительное (`0`) перемещение
`Play([{Продолжительность}])`	Все преобразования	Запускает преобразование
`Stop()`	Все преобразования	Немедленно останавливает преобразование

Пример Web-страницы, использующей фильтры и преобразования

Завершает этот раздел пример Web-страницы. Давайте создадим что-то, похожее на рекламную листовку — будем рекламировать фильтры и преобразования Internet Explorer. На этой странице мы поместим три текстовых элемента, к которым будут применены три разных фильтра. Четвертый элемент, содержимое которого будет меняться, мы подвергнем действию преобразования, причем тоже три, в зависимости от отображаемого в нем текста.

```
<HTML>
<HEAD>
<TITLE>Фильтры и преобразования</TITLE>
<SCRIPT>
var nFilter = 0;
```

Это номер преобразования.

```
function changeFilter() {
```

Эта функция будет применяться при завершении преобразования.

```
slideShow.filters.item(nFilter).Apply();
```

"Замораживаем" элемент страницы.

```
switch (nFilter) {
  case 0 :
    slideShow.innerText = "Все стихии станут вам покорны...";
    break;
  case 1 :
    slideShow.innerText = "...если вы будете использовать встроенные
    ↳в Internet Explorer...";
    break;
  case 2 :
    slideShow.innerText = "...визуальные фильтры и преобразования!";
    break; }
```

В зависимости от номера преобразования помещаем в четвертый элемент страницы тот или иной текст.

```
slideShow.filters.item(nFilter).Play();
```

Затем запускаем преобразование.

```
if (nFilter == 2)
  nFilter = 0
else
  nFilter += 1; }
```

И в заключение изменяем номер преобразования.

```
</SCRIPT>
</HEAD>
<BODY STYLE="text-align: center; position: absolute"
↳onload="changeFilter();">
```

Нам нужно будет запустить преобразование при загрузке страницы.

```
<DIV STYLE="font-size: 24pt; color: red; left: 10; top: 10; width: 100;
↳height: 30; filter: progid:DXImageTransform.Microsoft.Glow()">
Огонь
</DIV>
```

Это первый статический элемент страницы. Мы применим к нему фильтр "тление".

```
<DIV STYLE="font-size: 24pt; color: blue; left: 10; top: 50; width: 100;
↳height: 30; filter: progid:DXImageTransform.Microsoft.Wave(Freq=4)">
Вода
</DIV>
```

Это второй статический элемент страницы, к которому мы применим фильтр "волны".

```
<DIV STYLE="font-size: 24pt; color: lightskyblue; left: 10; width: 100;
↳height: 30; top: 100; filter:
progid:DXImageTransform.Microsoft.Blur()">
Воздух
</DIV>
```

А третий статический элемент довольствуется фильтром "туман".

```
<DIV ID="slideShow" STYLE="font-size: 12pt; background-color: oldlace;
↳left: 10; top: 100; width: 200; height: 100; filter:
↳progid:DXImageTransform.Microsoft.Fade(Duration=3)
↳progid:DXImageTransform.Microsoft.Iris(Duration=3)
↳progid:DXImageTransform.Microsoft.Barn(Duration=3)"
↳onfilterchange="changeFilter();">
Все стихии будут вам покорны...
</DIV>
```

Это четвертый, динамический элемент страницы. Заметьте, как мы применили к нему сразу три преобразования. Они будут запускаться поочередно в функции `changeFilter`, помещенной в заголовке HTML-документа.

```
</BODY>
</HTML>
```

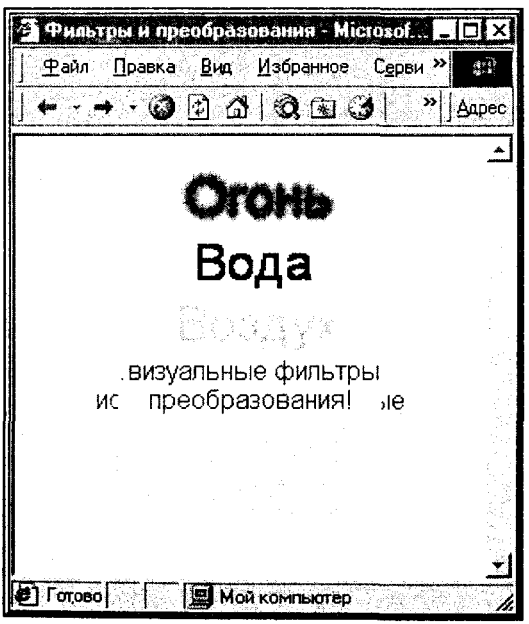


Рис. 7.1. Web-страница, созданная с использованием фильтров и преобразований

Сохраните этот код в файле под именем 7.1.htm и откройте его в Internet Explorer. На рис. 7.1 изображено то, что вы увидите. К несчастью, бумага не может передать движения...

И все это мы создали только с помощью HTML и JavaScript. Человек, не знакомый с фильтрами и преобразованиями Internet Explorer, может подумать, что использованы анимированные GIF-изображения, и ошибется. Одно из преимуществ фильтров и преобразований — красивый мультимедийный эффект, сделанный одной или двумя строчками кода.

Заключение

Вот и закончилась эта книга.

Я рассказал вам о многом. Конечно, это не полное описание всех возможностей, предлагаемых HTML, JavaScript и современными программами Web-обозревателей. Еще никому не удавалось объять необъятное. Однако вы научились создавать HTML-страницы, сначала простейшие, статические, а потом и динамические, с использованием объектной модели документа. Также вы узнали о взаимодействии с пользователем, работе с базами данных и мультимедийных эффектах, которые можно применять к элементам страницы. А это уже немало.

Современные Web-обозреватели — это сложнейшие и мощнейшие программные пакеты, состоящие из множества модулей. Полное их описание занимает десятки мегабайт и тысячи страниц печатной документации. Кроме того, фирмы-производители постоянно обновляют свои программы, в результате чего текущая документация непрерывно изменяется и дополняется.

К чему я это? А вот к чему.

Если вы хотите знать все о HTML, Web-дизайне, JavaScript, Web-программировании, объектной модели документа, даже пользовательских свойствах той или иной программы Web-обозревателя, вам нужно читать фирменную документацию и дополнительную литературу, написанную "по следам" такой документации. Проще всего добыть ее на сайте фирмы-производителя. Там же можно найти документацию в виде заархивированных файлов, доступных для скачивания. Они почти всегда бесплатны, т. е. вам не нужно будет платить за их получение или чтение.

Далее в таблице приведены несколько Web-адресов как сайтов фирм-производителей программ Web-обозревателей, так и других программистских сайтов, куда тоже стоит заглянуть. Сделайте себе закладки на эти сайты и заглядывайте туда почаще.

Адреса Web-сайтов, посвященных JavaScript и Web-программированию

Адрес	Описание
http://msdn.microsoft.com/ie/	"Дом" Internet Explorer. Исчерпывающее описание Web-обозревателя фирмы Microsoft, HTML, JavaScript и многого другого
http://developer.netscape.com/	"Дом" Netscape Navigator. Почти вся документация доступна для скачивания
http://www.w3c.org	Как сказал один мой хороший знакомый, "здесь есть все". А потому, что это сайт самого W3C
http://webreview.com/	Огромное количество документации по HTML, CSS, JavaScript, программам Web-обозревателей
http://docs.rinet.ru/	Огромное количество документации для программистов, и не только по HTML и JavaScript. Правда, по-английски
http://webims.virtualave.net/	Начала Web-дизайна и Web-программирования для начинающих. Все по-русски
http://www.free-graphics.ru/	Архив бесплатных графических файлов, которые вы можете использовать на своих Web-страницах
http://javascripts.boom.ru/	Архив бесплатных JavaScript-скриптов
http://www.sources.ru/	Архив бесплатных исходных текстов программ. Есть и JavaScript-скрипты

Вот и все. А я с вами прощаюсь. До свидания!

Владимир Дронов, vlad@vgumi.vlink.ru.

ПРИЛОЖЕНИЕ 1

HTML

В этом приложении описываются все теги и атрибуты HTML, поддерживаемые двумя популярнейшими программами Web-обозревателей: IE — Internet Explorer и N — Netscape Navigator.

Атрибуты

Атрибуты HTML приведены в алфавитном порядке. Поясняется значение атрибута, приводится описание его формата.

ABOVE

Задает слой, который будет перекрываться текущим слоем.

```
<LAYER ABOVE="{Имя слоя}">
```

Поддерживается N начиная с 4.0.

ACCESSKEY

Задает клавишу-ускоритель для быстрого доступа к элементу страницы. Например, можно присвоить клавишу-ускоритель полю ввода и при ее нажатии вместе с клавишей <Alt> поле ввода будет активизировано.

```
<INPUT ACCESSKEY="{Клавиша-ускоритель}">
```

Поддерживается IE начиная с 4.0 для элементов управления и внедренных объектов и начиная с 5.0 для всех остальных элементов.

ACTION

Задает интернет-адрес серверной программы, которой форма передаст введенные пользователем данные.

```
<FORM ACTION="{Интернет-адрес серверной программы}">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

ALIGN (<CAPTION> и <LEGEND>)

Задает выравнивание элемента заголовка таблицы (<CAPTION>) по отношению к таблице или пояснительной надписи (<LEGEND>) относительно группы полей.

```
<CAPTION ALIGN="bottom|center|left|right|top">
```

Для IE доступны пять значений:

- bottom — выравнивание по нижнему краю и по центру;
- center — выравнивание по центру;
- left — выравнивание по левому краю;
- right — выравнивание по правому краю;
- top — выравнивание по верхнему краю и по центру.

В N реализованы только значения top и bottom; значение по умолчанию top.

Поддерживается IE начиная с 3.02 для <CAPTION> и начиная с 4.0 для <LEGEND>. Поддерживается N начиная с 1.1.

ALIGN (<DIV>, <Hn>, <HR>, <P>, элементы таблицы)

Задает выравнивание содержимого элемента страницы.

```
<DIV ALIGN="left|center|right|justify">
```

Доступны четыре значения:

- left — выравнивание по левой границе (значение по умолчанию);
- center — выравнивание по центру (значение по умолчанию для <TH>);
- right — выравнивание по правой границе;
- justify — полное выравнивание по ширине (доступно только для <DIV>, <Hn> и <P>; N не поддерживает это значение).

Поддерживается IE начиная с 3.02 для <COL>, <COLGROUP>, <DIV>, <HR>, <P>, <TD>, <TH>, <TR> и начиная с 4.0 для <Hn>, <TBODY>, <TFOOT>, <THEAD>. Поддерживается N начиная с 1.1 для <P>, <HR> и элементов таблиц, а начиная с 2.0 для <DIV>.

ALIGN (<IFRAME> и <TABLE>)

Задает выравнивание "плавающего" фрейма (<IFRAME>) или таблицы (<TABLE>).

```
<IFRAME ALIGN="left|center|right">
```

Доступны три значения:

- `left` — выравнивание по левому краю доступного пространства (значение по умолчанию);
- `center` — выравнивание по центру доступного пространства;
- `right` — выравнивание по правому краю доступного пространства.

Поддерживается IE начиная с 3.02 для `<TABLE>` и начиная с 4.0 для `<IFRAME>`. Поддерживается N начиная с 4.0 для `<TABLE>`.

ALIGN (<IMG>, <SPACER>, элементы управления, внедренные элементы)

Задает местоположение рисунка, элемента управления или внедренного элемента относительно "обтекающего" его текста.

```
<IMG ALIGN="left|right|top|texttop|middle|absmiddle|baseline|bottom|
↳absbottom">
```

Доступны следующие значения:

- `left` — элемент смещается влево, а текст обтекает его справа (значение по умолчанию);
- `right` — элемент смещается вправо, а текст обтекает его слева;
- `top` — вершина элемента выравнивается по верхней границе строки;
- `texttop` — вершина элемента выравнивается по вершине самого высокого символа строки;
- `middle` — элемент центрируется в строке;
- `absmiddle` — центр элемента выравнивается точно по центру строки;
- `baseline` — низ элемента выравнивается по базовой линии строки;
- `bottom` — низ элемента выравнивается по низу строки;
- `absbottom` — низ изображения выравнивается по низу самого низко сидящего символа строки.

Поддерживается IE начиная с 3.02 для `<APPLET>`, `<IMG>`, `<INPUT>`, `<OBJECT>` и начиная с 4.0 для `<EMBED>`, `<FIELDSET>`, `<SELECT>`. Поддерживается N начиная с 1.1; значения атрибута `absmiddle`, `absbottom`, `texttop`, `middle` и `baseline` реализованы начиная с 3.0 для всех элементов, кроме `<EMBED>` и `<OBJECT>`.

ALINK

Задает цвет активных гиперссылок в документе.

```
<BODY ALINK="{Цвет}">
```

Поддерживается IE начиная с 4.0 и N начиная с 1.0.

ALLOWTRANSPARENCY

Задает прозрачность фрейма, обычного или "плавающего".

```
<IFRAME [ALLOWTRANSPARENCY]>
```

По умолчанию фон фрейма определяется отображаемой в нем Web-страницей. Если же в тег включен атрибут ALLOWTRANSPARENCY, то фон фрейма можно задать каким угодно; если же он не задан, используется цвет фона Web-страницы, где определен фрейм.

Пример использования атрибута ALLOWTRANSPARENCY.

```
<BODY STYLE="background-color: red">
<IFRAME ID="Frame1" SRC="somedoc.htm" allowTransparency="true">
</IFRAME>
<IFRAME ID="Frame2" SRC="somedoc.htm" allowTransparency="true"
↳STYLE="background-color: green">
</IFRAME>
<IFRAME ID="Frame3" SRC="somedoc.htm">
</IFRAME>
<IFRAME ID="Frame4" SRC="somedoc.htm" STYLE="background-color: green">
</IFRAME>
</BODY>
```

Фрейм Frame1 будет иметь красный фон, т. к. атрибут ALLOWTRANSPARENCY имеет значение true; красный фон страницы будет "просвечивать" сквозь него. Фрейм Frame2 будет иметь зеленый фон, т. к. атрибут ALLOWTRANSPARENCY имеет значение true, и в определении стиля задан фоновым цветом зеленый. А вот фреймы Frame3 и Frame4 будут иметь цвет фона, определяемый загруженным в них документом, т. к. атрибут ALLOWTRANSPARENCY у них имеет значение false. И хотя в определении стиля фрейма Frame4 задан зеленый фон, это не будет иметь значения.

Поддерживается IE начиная с 5.5.

ALT

Задает "альтернативный" текст, отображаемый на месте изображения, если оно еще не загружено.

```
<IMG ALT="{"Альтернативный" текст}">
```

Поддерживается IE начиная с 3.02 для <IMG> и <INPUT TYPE="image">, а начиная с 4.0 для <AREA>. Поддерживается N начиная с 1.0 для <INPUT TYPE="image"> и <IMG>, а начиная с 3.0 для <APPLET>.

ARCHIVE

Задает адрес архивного файла в формате ZIP, в котором находится файл Java-класса.

```
<APPLET ARCHIVE="{Интернет-адрес архивного файла}">
```

Поддерживается IE начиная с 3.0.

AUTOCOMPLETE

Задает или отменяет возможность "автозаполнения" форм и полей ввода текста.

```
<FORM AUTOCOMPLETE="on|off">
```

IE имеет возможность так называемого "автозаполнения" форм и полей ввода. Когда пользователь начинает вводить текст в поле, ниже появляется всплывающее меню, содержащее ранее введенные в него значения. Пользователь может выбрать значение из этого меню или ввести новое; в последнем случае оно также будет сохранено.

Формы и поля ввода текста идентифицируются их именами, заданными в атрибуте NAME. IE определяет по имени формы или поля ввода значения, которые были введены в них ранее.

Доступны два значения: on включает автозаполнение, а off отключает. Значения по умолчанию нет, точнее, оно зависит от настроек пользователя.

Поддерживается IE начиная с 5.0.

BACKGROUND

Задает фоновый рисунок для документа или таблицы.

```
<BODY BACKGROUND="{Интернет-адрес файла изображения}">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.1.

BALANCE

Задает стереобаланс при воспроизведении фонового звука с использованием тега <BGSOUND>.

```
<BGSOUND BALANCE="{Значение баланса}">
```

Может принимать значения от −10 000 (крайняя левая позиция) до 10 000 (крайняя правая позиция). Значение по умолчанию 0 (центр).

Поддерживается IE начиная с 4.0.

BEHAVIOR

Задает поведение прокручивающегося текста в элементе <MARQUEE>.

```
<MARQUEE BEHAVIOR="scroll|alternate|slide">
```

Доступны три значения:

- scroll (значение по умолчанию) — задает обычную прокрутку, т. е. текст уходит в одну сторону и появляется с противоположной;
- alternate — заставляет текст двигаться взад-вперед;
- slide — заставляет текст прокрутиться до конца и остановиться.

Поддерживается IE начиная с 3.02.

BELOW

Задает слой, которым будет перекрываться текущий слой.

```
<LAYER BELOW="{Имя слоя}">
```

Поддерживается N начиная с 4.0.

BGCOLOR

Задает цвет фона.

```
<BODY BGCOLOR="{Цвет}">
```

Поддерживается IE начиная с 3.02. Поддерживается N начиная с 1.1 для <BODY> и <TD>, а начиная с 3.0 для <TABLE> и <TH>.

BGPROPERTIES

Задает поведение фонового рисунка документа: будет ли он прокручиваться при прокрутке содержимого или нет.

```
<BODY [BGPROPERTIES="fixed"]>
```

Если в качестве значения этого атрибута указана пустая строка (значение по умолчанию), фоновый рисунок будет прокручиваться при прокрутке содержимого документа. Если указано значение fixed, то фон прокручиваться не будет.

Поддерживается IE начиная с 3.02.

BORDER (<EMBED>)

Задает толщину в пикселах границы, отображаемой вокруг внедренного элемента. Значение 0 убирает границу совсем.

```
<EMBED BORDER="{Толщина границы в пикселах}">
```

Поддерживается N начиная с 2.0.

BORDER (<FRAMESET> и <IFRAME>)

Задает толщину границы между фреймами в пикселах. Значение 0 убирает границу совсем.

```
<FRAMESET BORDER="{Толщина границы в пикселах}">
```

Значение по умолчанию 5.

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

BORDER (<IMG> и <TABLE>)

Задает толщину в пикселах границы, отображаемой вокруг рисунка или таблицы. Значение 0 убирает границу совсем.

```
<IMG BORDER="{Толщина границы в пикселах}">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.1.

BORDERCOLOR

Задает цвет границы.

```
<TABLE BORDERCOLOR="{Цвет}">
```

Поддерживается IE начиная с 3.02 для таблиц и их элементов и начиная с 4.0 для фреймов. Поддерживается N начиная с 3.0.

BOTTOMMARGIN

Задает расстояние в пикселах от нижней границы окна Web-обозревателя до нижней границы содержимого Web-страницы.

```
<BODY BOTTOMMARGIN="{Расстояние}">
```

Значение по умолчанию 15.

Поддерживается IE начиная с 4.0.

CELLPADDING

Задает расстояние между границей и содержимым ячейки таблицы в пикселах либо в процентах от доступного пространства.

```
<TABLE CELLPADDING="{Расстояние}">
```

Значение по умолчанию 1.

Поддерживается IE начиная с 3.02 и N начиная с 1.1.

CELLSPACING

Задает расстояние между ячейками таблицы в пикселах либо в процентах от доступного пространства.

```
<TABLE CELLSPACING="{Расстояние}">
```

Значение по умолчанию 2.

Поддерживается IE начиная с 3.02 и N начиная с 1.1.

CHALLENGE

Задает строку вызова для тега <KEYGEN>.

```
<KEYGEN CHALLENGE="{Строка вызова}">
```

Поддерживается N начиная с 1.0.

CHECKED

Задает включенное состояние флажка или радиокнопки.

```
<INPUT TYPE="checkbox" [CHECKED]>
```

По умолчанию все флажки и радиокнопки выключены. Чтобы включить флажок или радиокнопку, включите в тег атрибут CHECKED.

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

CLASS

Задает стилевой класс элемента страницы.

```
<DIV CLASS="{Стилевой класс, определенный в таблице стилей}">
```

Поддерживается IE и N начиная с 4.0.

CLASSID

Задает уникальный идентификатор класса элемента ActiveX.

```
<OBJECT CLASSID="{Уникальный идентификатор класса}">
```

Уникальный идентификатор класса является шестнадцатеричным представлением 128-разрядного двоичного числа и используется для однозначного определения файла, в котором находится код нужного элемента ActiveX. Уникальный идентификатор класса записывается в Реестр Windows при установке и регистрации элемента ActiveX на компьютере клиента.

Поддерживается IE начиная с 3.02 и N начиная с 4.0.

CLEAR

Задает поведение текста, разорванного тегом <BR>, при "обтекании" им некоторых элементов страницы, таких как изображения.

```
<BR CLEAR="all|left|right|none">
```

Доступны четыре значения:

- `all` — запрещает тексту "обтекать" элемент страницы с обеих сторон;
- `left` — с левой стороны;
- `right` — с правой стороны;
- `none` — разрешает тексту "обтекать" элемент страницы.

Поддерживается IE начиная с 3.02 и N начиная с 1.1.

CLIP

Задает прямоугольник, за пределами которого содержимое слоя не будет видно (видимая область слоя). Представляет собой список из четырех координат в пикселах, разделенных запятыми.

```
<LAYER CLIP="{X1, Y1, X2, Y2}">
```

Поддерживается N начиная с 4.0.

CODE

Задает интернет-адрес файла, содержащего откомпилированный класс Java.

```
<APPLET CODE="{Интернет-адрес файла с классом Java}">
```

Поддерживается IE начиная с 4.0 и N начиная с 2.0.

CODEBASE (<APPLET>)

Задает интернет-адрес папки, в которой содержится файл Java-класса.

```
<APPLET CODEBASE="{Интернет-адрес дистрибутивного файла}">
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

CODEBASE (<OBJECT>)

Задает интернет-адрес папки, в которой содержится дистрибутивный файл элемента ActiveX, Java-класса или расширения (plug-in) Web-обозревателя.

```
<OBJECT CODEBASE="{Интернет-адрес дистрибутивного файла элемента ActiveX
↳{#version={Номер версии элемента ActiveX}]}">
```

В IE значение этого атрибута может содержать номер требуемой версии элемента ActiveX. Он имеет вид "1,2,3,4".

```
<OBJECT CODEBASE="http://www.somesite.ru/dists/elem.cab#version=1,1,0,0">
```

Поддерживается IE начиная с 3.02 и N начиная с 4.0.

CODETYPE

Задает тип данных MIME элемента ActiveX.

```
<OBJECT CODETYPE="{Тип данных MIME}">
```

Поддерживается IE начиная с 3.02.

COLOR

Задает цвет шрифта или горизонтальной линии.

```
<FONT COLOR="{Цвет}">
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

COLS (<FRAMESET>)

Задает количество и значения ширины всех фреймов-колонок в наборе фреймов.

```
<FRAMESET COLS="Список значений ширины фреймов-колонок, разделенных
↳запятыми">
```

Значения ширины фреймов-колонок могут быть заданы как в пикселах, так и в процентах от ширины всего набора фреймов. Также может быть использовано значение *, предписывающее Web-обозревателю отвести соответствующему фрейму все оставшееся место.

```
<FRAMESET COLS="100, 20%, *">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

COLS (<MULTICOL>)

Задает количество колонок, в которых будет располагаться текст.

```
<MULTICOL COLS="{Количество колонок}">
```

Поддерживается N начиная с 3.0.

COLS (<PRE>)

Задает ширину в символах текста с заданным форматом, выводимым с помощью тега <PRE>. Если длина строки превысит заданную ширину, и если в тег <PRE> включен атрибут WRAP, Web-обозреватель перенесет эту строку.

```
<PRE COLS="{Ширина в символах}">
```

Поддерживается N начиная с 1.0.

COLS (<TABLE>)

Задает количество колонок в таблице. Указание количества колонок позволяет ускорить обработку таблицы.

```
<TABLE COLS="{Количество колонок}">
```

Поддерживается IE начиная с 3.02 и N начиная с 3.0.

COLS (<TEXTAREA>)

Задает ширину в символах текста области редактирования.

```
<TEXTAREA COLS="{Ширина в символах}">
```

Значение по умолчанию 20.

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

COLSPAN

Задает количество ячеек, которые должны быть объединены в одну.

```
<TD COLSPAN="{Количество ячеек, объединяемых в одну}">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.1.

COMPACT

Задает компактное отображение позиций списков.

```
<DL [COMPACT]>
```

По умолчанию все списки отображаются в обычном режиме. Чтобы включить компактное отображение (т. е. без дополнительных пробелов), включите в тег списка атрибут COMPACT.

Поддерживается IE начиная с 4.0 и N начиная с 1.0.

CONTENT

Задает метаданные в теге <META>. Тип метаданных зависит от значений атрибутов HTTP-EQUIV или NAME.

```
<META CONTENT="{Метаданные}">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.1.

CONTENTEDITABLE

Задает или отменяет возможность редактирования содержимого Web-страницы или ее элемента пользователем.

```
<DIV CONTENTEDITABLE="inherit|true|false">
```

Доступны следующие значения:

- `inherit` (значение по умолчанию) — заставляет элемент страницы "наследовать" эту возможность от родителя;
- `true` разрешает, а `false` запрещает пользователю редактировать содержимое элемента страницы;

Если значение этого свойства равно `true` (или `inherit`, и при этом родитель допускает редактирование), пользователь сможет редактировать содержимое элемента или всей страницы, как в обычном текстовом редакторе.

Поддерживается IE начиная с 5.5.

COORDS

Задает координаты "горячей" области в карте-изображении. Зависит от значения атрибута SHAPE.

```
<AREA COORDS="{Координаты "горячей" области}">
```

Доступны три типа "горячих" областей и, соответственно, три типа значения атрибута COORDS:

- `rect` — прямоугольник. Атрибут COORDS имеет формат `COORDS="{X1}, {Y1}, {X2}, {Y2}"`, где `X1` и `Y1` — координаты верхнего левого, а `X2` и `Y2` — правого нижнего угла прямоугольника;
- `circle` — круг. Атрибут COORDS имеет формат `COORDS="{X центра}, {Y центра}, {Радиус}"`;

- poly — многоугольник. Атрибут COORDS имеет формат COORDS="{X1}, {Y1}, {X2}, {Y3}, {X3}, {Y3}...", где Xn и Yn — координаты соответствующей вершины.

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

DATA

Задает интернет-адрес файла данных, требуемых элементом ActiveX.

```
<OBJECT DATA="{Интернет-адрес файла данных}">
```

Поддерживается IE начиная с 3.02 и N начиная с 4.0.

DATAFLD

Задает поле таблицы базы данных, значение которого будет отображаться в элементе страницы. Требует задания источника данных в атрибуте DATASRC.

```
<DIV DATAFLD="{Имя поля таблицы базы данных}">
```

Поддерживается IE начиная с 4.0.

DATAFORMATAS

Задает представление, используемое для отображения данных из полей таблицы.

```
<DIV DATAFORMATAS="text|html|localized-text">
```

Доступны три значения:

- text (значение по умолчанию) — задает текстовое представление;
- html — заставляет Web-обозреватель учитывать все теги HTML, встречающиеся в значениях полей;
- localized-text — работает аналогично text, за тем исключением, что при этом используются региональные настройки Windows.

Поддерживается IE начиная с 4.0.

DATAPAGESIZE

Задает размер страницы данных, выводимых за один раз, в записях. Требуется, если нужно привязать таблицу к данным.

```
<TABLE DATAPAGESIZE="{Размер страницы данных}">
```

Поддерживается IE начиная с 4.0.

DATASRC

Задает объект-источник данных.

```
<DIV DATASRC="#{Имя объекта-источника данных}">
```

Поддерживается IE начиная с 4.0.

DEFER

Позволяет отложить выполнение скриптов, пока Web-страница не будет загружена полностью.

```
<SCRIPT [DEFER]>
```

По умолчанию скрипты выполняются при загрузке Web-страницы. Если вы хотите отложить их выполнение до полной загрузки страницы, включите в тег нужного скрипта атрибут `DEFER`. Это может быть полезно при разработке дизайна страницы или для увеличения скорости ее загрузки.

Поддерживается IE начиная с 4.0.

DIR

Задает порядок чтения текста.

```
<DIV DIR="ltr|rtl">
```

По умолчанию текст читается слева направо (`ltr`). Значение `rtl` позволяет задать порядок чтения справа налево. Для документов, составленных на европейских языках, порядок чтения всегда слева направо (`ltr`).

Поддерживается IE начиная с 5.0.

DIRECTION

Задает направление прокрутки текста в элементе `<MARQUEE>`.

```
<MARQUEE DIRECTION="{left|right|up|down}">
```

Текст может прокручиваться налево (`left`; значение по умолчанию), направо (`right`), вверх (`up`) и вниз (`down`).

Поддерживается IE начиная с 3.02.

DISABLED

Запрещает доступ к элементу страницы.

```
<TEXTAREA [DISABLED]>
```

По умолчанию доступ к элементу страницы разрешен. При этом пользователь может редактировать текст в поле ввода, включать и отключать флажки и радиокнопки, выбирать пункты из списков, нажимать командные кнопки и редактировать содержимое остальных элементов страницы, если для них было задано соответствующее значение атрибута CONTENTEDITABLE. Чтобы запретить доступ к элементу страницы, включите в его тег атрибут DISABLED.

Поддерживается IE начиная с 4.0 для элементов управления и начиная с 5.5 для остальных элементов страницы.

DYNSRC

Задает интернет-адрес динамического содержимого, отображаемого вместо статичного рисунка. Динамическим содержимым для рисунка может быть видеофильм.

```
<IMG DYNSRC="{Интернет-адрес динамичного содержимого}">
```

Если задан этот атрибут, вместо обычного статичного рисунка будет отображен видеофильм, содержащийся в файле, интернет-адрес которого был задан в этом атрибуте.

```
<IMG SRC="SBlogo.jpg" DYNSRC="SantaBarbara.avi">
```

Поддерживается IE начиная с 3.02.

ENCTYPE

Задает метод кодирования данных, отправляемых формой в виде MIME.

```
<FORM ENCTYPE="{Метод кодировки данных}">
```

Значение по умолчанию application/x-www-form-urlencoded. Может также использоваться значение multipart/form-data, если в форме применен элемент <INPUT TYPE="file">.

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

EVENT

Задает событие, к которому привязан скрипт-обработчик.

```
<SCRIPT EVENT="{Тип события}">
```

Этот способ создания обработчиков событий специфичен для Internet Explorer.

```
<SCRIPT EVENT="onclick()" FOR="pSome">
```

Поддерживается IE начиная с 3.02.

FACE

Задает имя шрифта.

```
<FONT FACE="{Имя шрифта}">
```

Поддерживается IE начиная с 3.02 для <FONT> и начиная с 4.0 для <BASEFONT>. Поддерживается N начиная с 3.0.

FOR (<LABEL>)

Задает элемент управления, к которому привязана метка <LABEL>.

```
<LABEL FOR="{Имя элемента управления}">
```

Поддерживается IE начиная с 4.0.

FOR (<SCRIPT>)

Задает элемент страницы, к которому привязан скрипт.

```
<SCRIPT FOR="{Имя элемента страницы}">
```

Этот способ создания обработчиков событий специфичен для IE.

```
<SCRIPT EVENT="onclick()" FOR="pSome">
```

Поддерживается IE начиная с 3.02.

FRAME

Задает отображение внешних рамок таблицы.

```
<TABLE FRAME="void|above|below|border|hsides|lhs|rhs|vsides|box">
```

Доступны девять значений:

- `void` — внешней рамки нет вообще (значение по умолчанию);
- `above` — рисуется только верхняя линия внешней рамки;
- `below` — рисуется только нижняя линия внешней рамки;
- `hsides` — рисуются только горизонтальные линии внешней рамки, т. е. верхняя и нижняя;
- `lhs` — рисуется только левая линия внешней рамки;
- `rhs` — рисуется только правая линия внешней рамки;
- `vsides` — рисуются только вертикальные линии внешней рамки, т. е. левая и правая;
- `box` — рисуются все линии внешней рамки.

Поддерживается IE начиная с 3.02.

FRAMEBORDER (<EMBED>)

Задает или отменяет отображение границы вокруг внедренного элемента.

```
<FRAMESET [FRAMEBORDER="no"]>
```

По умолчанию граница вокруг внедренного элемента отображается. Чтобы убрать ее, включите в тег <EMBED> этот атрибут.

Поддерживается N начиная с 2.0.

FRAMEBORDER (фреймы)

Задает или отменяет отображение границ между фреймами.

```
<FRAMESET FRAMEBORDER="1|0|yes|no">
```

По умолчанию границы между фреймами отображаются (значения `1` или `yes`). Значения `0` или `no` отключают отображение границ.

Поддерживается IE начиная с 3.02 и N начиная с 3.0. В N не реализованы значения `1` и `0`.

FRAMESPACING

Задает дополнительное пространство между фреймами в пикселах.

```
<FRAMESET FRAMESPACING="{Дополнительное пространство между фреймами}">
```

Значение по умолчанию 2.

Поддерживается IE начиная с 3.02.

GUTTER

Задает расстояние между колонками текста в пикселах.

```
<MULTICOL GUTTER="{Расстояние между колонками}">
```

Значение по умолчанию 2.

Поддерживается N начиная с 3.0.

HEIGHT

Задает высоту элемента страницы в пикселах или как процент от доступного пространства.

```
<TABLE HEIGHT="{Высота}">
```

Поддерживается IE начиная с 3.02 для <EMBED>, <IMG>, <MARQUEE>, <OBJECT>, <TD> и <TH>, начиная с 4.0 для <TABLE>, начиная с 5.0 для <TR> и начиная с 5.5 для <FRAME>. Поддерживается N начиная с 1.1.

HIDDEN

Задает или отменяет показ внедренного элемента на Web-странице.

```
<EMBED HIDDEN="true|false">
```

По умолчанию внедренный элемент показывается на странице (значение `false`). Чтобы скрыть его, присвойте этому атрибуту значение `true`.

Поддерживается N начиная с 2.0.

HIDEFOCUS

Отменяет показ фокуса ввода (текстового курсора).

```
<TEXTAREA [HIDEFOCUS]>
```

По умолчанию фокус ввода всегда показывается. Чтобы скрыть его, включите в тег элемента страницы атрибут `HIDEFOCUS`.

Поддерживается IE начиная с 5.5.

HREF (<BASE>)

Задает базовый интернет-адрес в теге `<BASE>`, относительно которого отсчитываются все ссылки.

```
<BASE HREF="{Базовый интернет-адрес}">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

HREF (гиперссылки)

Задает интернет-адрес назначения, куда пользователь попадет после активизации гиперссылки.

```
<A HREF="{Интернет-адрес назначения}">
```

Поддерживается IE и N начиная с 1.0.

HSPACE

Задает горизонтальное расстояние от элемента страницы до окружающего его текста.

```
<IMG HSPACE="{Горизонтальный отступ}">
```

Значение по умолчанию `0`.

Поддерживается IE начиная с 3.02 для `<APPLET>`, `<IMG>`, `<INPUT TYPE="image">`, `<MARQUEE>` и `<OBJECT>` и начиная с 4.0 для `<IFRAME>`. Поддерживается N начиная с 1.1.

HTTP-EQUIV

Задает тип метаданных, задаваемых в теге <META>.

```
<META HTTP-EQUIV="description|refresh|url|mimetype|charset|Expires|
↳Refresh|Set-Cookie">
```

Для IE доступны пять предопределенных значений:

- description — задает описание Web-страницы;
- refresh — задает в секундах интервал обновления Web-страницы, отображенной Web-обозревателем;
- url — задает адрес, откуда будет загружаться Web-страница при обновлении. Требует наличия тега <META>, задающего интервал обновления (refresh);
- mimetype — задает тип данных MIME Web-страницы (всегда text/html);
- charset — задает кодовую страницу, использовавшуюся при написании текста.

N поддерживает три значения:

- Expires — задает дату, после которой Web-страница, сохраненная в памяти Web-обозревателя, считается устаревшей и должна быть обновлена. Если дата задана некорректно (например, "0"), страница считается устаревшей сразу после первого посещения;
- Refresh — задает в секундах интервал обновления Web-страницы, отображенной Web-обозревателем;
- Set-Cookie — передает Web-обозревателю cookie.

Также могут поддерживаться другие, нестандартные типы метаданных.

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

ID

Задает уникальное имя элемента страницы для доступа к нему из скрипта.

```
<P ID="{Имя элемента страницы}">
```

Поддерживается IE и N начиная с 4.0.

ISMAP

Задает поведение рисунка <IMG> как карты-изображения.

```
<IMG [ISMAP]>
```

По умолчанию рисунок не является картой-изображением. Чтобы сделать его картой-изображением, включите в тег атрибут ISMAP.

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

LANG

Задает язык, на котором набран текст.

```
<P LANG="{Код языка}">
```

Список доступных кодов языка приведен в конце этого приложения.

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

LANGUAGE

Задает язык программирования, на котором написан скрипт, привязанный к элементу страницы.

```
<DIV LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка}">
```

Доступны пять предопределенных значений:

- JScript;
- javascript;
- vbs;
- vbscript;
- XML.

Также могут поддерживаться другие, нестандартные языки.

N поддерживает указание после имени языка — через пробел — его версии. Например, "javascript 1.1".

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

LEFT

Задает горизонтальную координату левого верхнего угла слоя относительно родителя.

```
<LAYER LEFT="{Координата X}">
```

Поддерживается N начиная с 4.0.

LEFTMARGIN

Задает расстояние от левой границы окна Web-обозревателя до левой границы содержимого Web-страницы в пикселах.

```
<BODY LEFTMARGIN="{Расстояние}">
```

Значение по умолчанию 10.

Поддерживается IE начиная с 4.0.

LINK

Задает цвет гиперссылок в документе.

```
<BODY LINK="{Цвет}">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

LOOP (<BGSOUND>, <IMG>, <INPUT TYPE="image">)

Задает количество повторений фонового звука или видеофильма.

```
<BGSOUND LOOP="-1|0|{Количество}">
```

Значение `-1` или `0` задает бесконечное повторение. Значение по умолчанию `1`.

Поддерживается IE начиная с 3.02.

LOOP (<MARQUEE>)

Задает, сколько раз в элементе `<MARQUEE>` будет прокручиваться текст.

```
<MARQUEE LOOP="-1|0|{Количество}">
```

Значение `-1` или `0` задает бесконечное повторение. Значение по умолчанию `-1`.

Поддерживается IE начиная с 3.02.

LOWSRC

Задает интернет-адрес файла с изображением низкого разрешения.

```
<IMG LOWSRC="{Интернет-адрес файла изображения}">
```

Поддерживается IE начиная с 4.0 и N начиная с 1.0.

MARGINHEIGHT

Задает в пикселах отступ по вертикали между границей фрейма и его содержимым.

```
<FRAME MARGINHEIGHT="{Отступ}">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

MARGINWIDTH

Задает в пикселах отступ по горизонтали между границей фрейма и его содержимым.

```
<FRAME MARGINWIDTH="{Отступ}">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

MAXLENGTH

Задает максимальное количество символов, которое пользователь может ввести в поле ввода.

```
<INPUT TYPE="text" MAXLENGTH="{Максимальное количество символов}">
```

Поддерживается IE начиная с 4.0 и N начиная с 1.0.

MAYSCRIPT

Разрешает или запрещает доступ из скрипта к Java-апплету.

```
<APPLET [MAYSCRIPT]>
```

По умолчанию скрипты не могут управлять Java-апплетом. Чтобы разрешить скриптам доступ к апплету, нужно включить в тег `<APPLET>` атрибут `MAYSCRIPT`.

Поддерживается N начиная с 3.0.

MEDIA

Задает способ применения таблицы стилей: только для просмотра Web-страницы на экране компьютера, только для печати на принтере или и для того, и для другого.

```
<STYLE MEDIA="screen|print|all">
```

Доступны три значения:

- `screen` — заставляет Web-обозреватель использовать таблицу стилей только для просмотра Web-страницы на экране;
- `print` — только для печати на принтере;
- `all` — и для просмотра, и для печати (значение по умолчанию).

Поддерживается IE начиная с 4.0 для `<LINK>` и начиная с 5.0 для `<STYLE>`.

METHOD

Задает метод передачи данных формой: `get` или `post`.

```
<FORM METHOD="get|post">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

METHODS

Задает список методов протокола HTTP, поддерживаемых гиперссылкой.

```
<A METHODS="{Список методов HTTP, разделенных запятыми}">
```

Поддерживается IE начиная с 4.0.

MULTIPLE

Задает возможность выбора пользователем нескольких значений в списке `<SELECT>`.

```
<SELECT [MULTIPLE]>
```

По умолчанию в списке может быть выбрано только одно значение. Чтобы дать возможность пользователю выбирать из списка одновременно несколько значений, включите в тег списка атрибут `MULTIPLE`.

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

NAME (<A>)

Задает имя "якоря".

```
<A NAME="{Имя "якоря"}">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

NAME (<FRAME>)

Задает имя фрейма.

```
<FRAME NAME="{Имя фрейма}">
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

NAME (<IMG>, элементы управления, внедренные объекты)

Задает имя элемента страницы, используемое в скриптах.

```
<SELECT NAME="{Имя}">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

NAME (<META>)

Задает в теге <META> тип метаданных. Сами метаданные передаются значением атрибута CONTENT.

```
<META NAME="Author|Description|Generator|Keywords|ProgID|Robots|Template|
↳{Тип}">
```

Для IE доступны шесть предопределенных значений:

- Description — обозначает описание Web-страницы;
- Generator — задает имя приложения, создавшего Web-страницу;
- Keywords — задает список ключевых слов, разделенных запятыми. Эти ключевые слова будут использоваться поисковыми машинами;
- ProgID — идентифицирует программу для редактирования Web-страниц;
- Robots — разрешает или запрещает индексацию Web-страницы поисковыми машинами. Доступны два значения атрибута CONTENT: all и noindex, соответственно разрешающее и запрещающее индексацию;
- Template — задает имя файла шаблона, примененного для создания Web-страницы. Используется вместе с ProgID.

N поддерживает только три предопределенных значения:

- Description — обозначает описание Web-страницы;
- Keywords — задает список ключевых слов, разделенных запятыми. Эти ключевые слова будут использоваться поисковыми машинами;
- Author — указывает имя автора Web-страницы.

Также допускаются любые другие, не предопределенные типы метаданных.

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

NAME (<PARAM>)

Задает имя параметра внедренного объекта.

```
<PARAM NAME="{Имя}">
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

NOHREF

Используется при создании "пустой", не привязанной ни к чему области на карте-изображении.

```
<AREA [NOHREF]>
```

По умолчанию область карты-изображения привязана к какой-то гиперссылке. Чтобы сделать "пустую" область, включите в ее тег атрибут NOHREF.

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

NORESIZE

Используется при создании фрейма, размер которого не может быть изменен пользователем.

```
<FRAME [NORESIZE]>
```

По умолчанию пользователь может изменять размер фрейма, перетаскивая мышью его границу. Чтобы запретить это, включите в тег фрейма атрибут NORESIZE.

Поддерживается IE начиная с 4.0 и N начиная с 1.0.

NOSHADE

Отключает тень у горизонтальной линейки.

```
<HR [NOSHADE]>
```

По умолчанию горизонтальная линейка имеет трехмерный вид. Чтобы сделать ее обычной, "плоской", включите в тег <HR> атрибут NOSHADE.

Поддерживается IE начиная с 3.02 и N начиная с 1.1.

NOWRAP

Отключает автоматический перенос слов в тексте.

```
<TD [NOWRAP]>
```

По умолчанию слова в тексте переносятся на новую строку автоматически, если ширина элемента страницы недостаточна для отображения строки целиком. Чтобы отменить автоматический перенос, включите в тег атрибут NOWRAP.

Поддерживается IE начиная с 3.02 для <TD> и <TH> и начиная с 4.0 для <BODY>, <DD>, <DIV> и <DT>. Поддерживается N начиная с 1.1.

PAGEX

Задает горизонтальную координату левого верхнего угла слоя относительно окна Web-обозревателя.

```
<LAYER PAGEX="{Координата X}">
```

Поддерживается N начиная с 4.0.

PAGEY

Задает вертикальную координату левого верхнего угла слоя относительно окна Web-обозревателя.

```
<LAYER PAGEY="{Координата Y}">
```

Поддерживается N начиная с 4.0.

PALETTE

Задает цветовую палитру Windows, которую будет использовать внедренный элемент.

```
<EMBED PALETTE="foreground|background">
```

По умолчанию внедренный элемент выбирает фоновую палитру (значение `background`). Чтобы заставить его прибегнуть к основной палитре, используемой для отображения содержимого страницы, задайте этому атрибуту значение `foreground`.

Поддерживается N начиная с 2.0.

PLUGINSPAGE

Задает интернет-адрес Web-страницы, содержащей инструкции по инсталляции расширения Web-обозревателя.

```
<EMBED PLUGINSPAGE="{Интернет-адрес инструкций по инсталляции}">
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

PLUGINURL

Задает интернет-адрес дистрибутивного файла расширения Web-обозревателя. Рекомендуется использовать вместо `PLUGINSPAGE`.

```
<EMBED PLUGINURL="{Интернет-адрес дистрибутивного файла расширения}">
```

Поддерживается N начиная с 4.0.

POINT-SIZE

Задает размер шрифта в пикселах.

```
<FONT POINT-SIZE="{Размер шрифта в пикселах}">
```

Поддерживается N начиная с 4.0.

PROMPT

Задает подсказку для тега <ISINDEX>.

```
<ISINDEX PROMPT="{Подсказка}">
```

Поддерживается N начиная с 1.1.

READONLY

Делает невозможным редактирование текста в поле ввода.

```
<TEXTAREA [READONLY]>
```

По умолчанию пользователь имеет возможность редактировать текст в поле ввода. Если вы хотите сделать этот текст нередактируемым, включите в тег поля ввода атрибут READONLY. Обратите внимание: при этом поле ввода останется разрешенным для доступа; пользователь сможет выделять и копировать из него текст (READONLY не то же самое, что DISABLED).

Поддерживается IE начиная с 4.0.

REL

Задает связь между текущей Web-страницей и той, на которую указывает гиперссылка.

```
<LINK REL="Alternate|Appendix|Bookmark|Chapter|Contents|Copyright|
↳Glossary|Help|Index|Next|Offline|Prev|Section|Shortcut Icon|
↳Start|Stylesheet|Subsection|fontdef|stylesheet">
```

Для IE доступно семнадцать возможных значений:

- Alternate — гиперссылка указывает на страницу, заменяющую текущую;
- Appendix — на приложение к большому многостраничному документу;
- Bookmark — на закладку;
- Chapter — на отдельную часть большого многостраничного документа;
- Contents — на содержание большого многостраничного документа;
- Copyright — на соглашение об авторских правах;
- Glossary — на словарь терминов, используемых в большом многостраничном документе;
- Help — на справочную страницу;
- Index — на список всех страниц большого многостраничного документа;
- Next — на следующий документ в последовательности;
- Offline — на CDF-файл "канала";

- Prev — на предыдущий документ в последовательности;
- Section — на отдельную часть большого многостраничного документа;
- Shortcut Icon — на файл значка, используемого для обозначения ссылки на текущую страницу;
- Start — на первый документ в последовательности;
- Stylesheet — на таблицу стилей;
- Subsection — на отдельный подраздел большого многостраничного документа.

N поддерживает только два возможных значения:

- fontdef — гиперссылка указывает на файл загружаемого шрифта;
- stylesheet — на таблицу стилей.

Поддерживается IE начиная с 3.02 для <A> и начиная с 4.0 для <LINK>. Поддерживается N начиная с 4.0.

REV

Аналогичен REL, но направлен в обратную сторону, т. е. сообщает, чем текущая страница является для той, на которую указывает гиперссылка.

```
<LINK REV="Alternate|Appendix|Bookmark|Chapter|Contents|Copyright|
↪Glossary|Help|Index|Next|Prev|Section|Start|Stylesheet|Subsection">
```

Доступно пятнадцать возможных значений:

- Alternate — страница, заменяющая другую;
- Appendix — приложение к большому многостраничному документу;
- Bookmark — закладка;
- Chapter — отдельная часть большого многостраничного документа;
- Contents — содержание большого многостраничного документа;
- Copyright — соглашение об авторских правах;
- Glossary — словарь терминов, используемых в большом многостраничном документе;
- Help — справочная страница;
- Index — список всех страниц большого многостраничного документа;
- Next — следующий документ в последовательности;
- Prev — предыдущий документ в последовательности;
- Section — отдельная часть большого многостраничного документа;
- Start — первый документ в последовательности;

- `Stylesheet` — таблица стилей;
- `Subsection` — отдельный подраздел большого многостраничного документа.

Поддерживается IE начиная с 3.02 для `<A>` и начиная с 4.0 для `<LINK>`.

RIGHTMARGIN

Задает расстояние от правой границы окна Web-обозревателя до правой границы содержимого Web-страницы в пикселах.

```
<BODY RIGHTMARGIN="{Расстояние}">
```

Значение по умолчанию 10.

Поддерживается IE начиная с 4.0.

ROWS (<FRAMESET>)

Задает количество и высоту всех фреймов-строк в наборе фреймов.

```
<FRAMESET ROWS="Список значений высоты фреймов-строк, разделенных
↪запятыми">
```

Значения высоты фреймов-строк могут быть заданы как в пикселах, так и в процентах от высоты всего набора фреймов. Также может быть использовано значение `*`, предписывающее Web-обозревателю отвести соответствующему фрейму все оставшееся место.

```
<FRAMESET ROWS="20, 40%, *">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

ROWS (<TEXTAREA>)

Задает высоту области редактирования в строках текста.

```
<TEXTAREA ROWS="{Высота в строках}">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

ROWSPAN

Задает количество строк в таблице, которые должны быть объединены в одну.

```
<TD ROWSPAN="{Количество строк, объединяемых в одну}">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.1.

RULES

Задает отображение внутренних рамок таблицы.

```
<TABLE RULES="none|rows|cols|groups|all">
```

Доступны пять значений:

- none — внутренних рамок нет совсем;
- rows — рисуются только горизонтальные линии (между строками);
- cols — рисуются только вертикальные линии (между столбцами);
- groups — рисуются вертикальные линии между группами колонок <COLGROUP> и горизонтальные линии между секциями <THEAD>, <TBODY> и <TFOOT>;
- all — рисуются все внутренние рамки.

Поддерживается IE начиная с 4.0.

SCROLL

Задает, будет ли окно Web-обозревателя иметь полосы прокрутки.

```
<BODY SCROLL="yes|no|auto">
```

Доступны три значения:

- yes — полосы прокрутки есть всегда, даже если они не нужны (значение по умолчанию);
- no — полос прокрутки нет, даже если содержимое Web-страницы не помещается в окно;
- auto — полосы прокрутки появляются только тогда, когда в них есть необходимость.

Поддерживается IE начиная с 4.0.

SCROLLAMOUNT

Задает шаг в пикселах, на который будет смещаться текст в <MARQUEE> за один раз.

```
<MARQUEE SCROLLAMOUNT="{Шаг в пикселах}">
```

Значение по умолчанию 6.

Поддерживается IE начиная с 3.02.

SCROLLDELAY

Задает задержку в миллисекундах перед каждым шагом прокрутки текста в <MARQUEE>.

```
<MARQUEE SCROLLDELAY="{Задержка в миллисекундах}">
```

Значение по умолчанию 85.

Поддерживается IE начиная с 3.02.

SCROLLING

Задает, будет ли прокручиваться при необходимости содержимое фрейма.

```
<FRAME SCROLLING="yes|no|auto">
```

Доступны три значения:

- `yes` — будет прокручиваться (появятся полосы прокрутки);
- `no` — не будет прокручиваться (полосы прокрутки не появятся, даже если в них возникнет необходимость);
- `auto` — полосы прокрутки появляются только тогда, когда в них есть необходимость (значение по умолчанию).

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

SELECTED

Задает, будет ли текущий пункт списка выбран изначально.

```
<OPTION [SELECTED]>
```

По умолчанию в списке будет выбран первый пункт. Если вы хотите, чтобы был выбран другой пункт, включите в его тег атрибут `SELECTED`.

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

SHAPE

Задает форму "горячей" области на карте-изображении.

```
<AREA SHAPE="circ|circle|poly|polygon|rect|rectangle">
```

Доступны три значения:

- `circ` (или `circle`) — круг;
- `poly` (или `polygon`) — многоугольник;
- `rect` (или `rectangle`) — прямоугольник.

Координаты "горячей" области задаются в атрибуте `COORDS`.

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

SIZE (<BASEFONT> и <FONT>)

Задает размер шрифта от 1 (минимальный) до 7 (максимальный).

```
<FONT SIZE="{Размер шрифта}">
```

Значение по умолчанию 3.

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

SIZE (<HR>)

Задает высоту горизонтальной линейки в пикселах.

```
<HR SIZE="{Высота в пикселах}">
```

Значение по умолчанию 2.

Поддерживается IE начиная с 3.02 и N начиная с 1.1.

SIZE (<SPACER>)

Задает величину дополнительного расстояния, добавляемого между символами в строке или между строками в абзаце, в зависимости от значения атрибута TYPE. Значение задается в пикселах.

```
<SPACER SIZE="{Дополнительное расстояние}">
```

Поддерживается N начиная с 3.0.

SIZE (элементы управления)

Задает размер элемента управления. Если это поле ввода, его размер указывается в символах текста. Если список или всплывающее меню — в пунктах. Во всех остальных случаях никак не влияет на внешний вид элемента управления.

```
<INPUT TYPE="text" SIZE="{Размер в символах}">
<SELECT SIZE="{Количество одновременно видимых пунктов}">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

SPAN

Задает количество колонок, на которое распространяется действие параметров, заданных в тегах <COL> или <COLGROUP>.

```
<COLGROUP SPAN="{Количество колонок}">
```

Значение по умолчанию 1.

Поддерживается IE начиная с 3.02.

SRC (<APPLET>, <BGSOUND>, <EMBED>, <XML>, фреймы, рисунки, слои)

Задает интернет-адрес файла, содержащего загружаемый элемент страницы: Java-апплет, данные, обрабатываемые с помощью расширения Web-обозревателя, содержимое фрейма или слоя, фоновый звук или рисунок.

```
<APPLET SRC="{Интернет-адрес файла с Java-апплетом}">
<IMG SRC="{Интернет-адрес файла рисунка}">
```

Поддерживается IE начиная с 3.02 для <BGSOUND>, <EMBED>, <FRAME>, <IFRAME> и <IMG>, начиная с 4.0 для <APPLET> и <INPUT TYPE="image">, а начиная с 5.0 для <XML>. Поддерживается N начиная с 1.0 для <FRAME>, <IMG> и <INPUT TYPE="image">, начиная с 2.0 для <EMBED>, а начиная с 4.0 для <ILAYER> и <LAYER>.

SRC (<LINK>)

Задает интернет-адрес файла, на который ссылается текущая страница.

```
<LINK SRC="{Интернет-адрес файла}">
```

Поддерживается N начиная с 4.0.

SRC (<SCRIPT>)

Задает интернет-адрес файла с текстом скрипта.

```
<SCRIPT SRC="{Интернет-адрес файла с текстом скрипта}">
```

Поддерживается IE начиная с 3.02 и N начиная с 3.0.

START (<IMG>)

Задает момент, с которого начнет воспроизводиться видеоклип, имя файла которого было задано в атрибуте DYNSRC.

```
<IMG START="fileopen|mouseover">
```

Доступно два значения: fileopen (значение по умолчанию) запускает клип сразу после его загрузки, а mouseover — только после того, как пользователь разместит над ним курсор мыши.

Поддерживается IE начиная с 3.02.

START (<OL>)

Задает начальную позицию нумерации элементов пронумерованного списка <OL>.

```
<OL START="{Начальный номер}">
```

Значение по умолчанию 1.

Поддерживается IE начиная с 3.02 и N начиная с 1.1.

STYLE

Задает встроенный стиль для элемента страницы.

```
<DIV STYLE="{Определение встроенного стиля}">
```

Поддерживается IE начиная с 3.02 для текстовых элементов страницы и начиная с 4.0 для нетекстовых (рисунки, фреймы, расширения и т. п.). Поддерживается N начиная с 4.0.

SUPPRESS

Включает или отключает отображение "альтернативного" текста и значка в левом правом углу рамки, когда изображение еще не загружено.

```
<IMG SUPPRESS="true|false">
```

По умолчанию и "альтернативный" текст, и значок отображаются (значение `false`). Если вы хотите убрать их, присвойте этому атрибуту значение `true`.

Поддерживается N начиная с 4.0.

TABINDEX

Задает номер в последовательности обхода элементов управления при нажатии клавиши <Tab>.

```
<TEXTAREA TABINDEX="{Порядковый номер в последовательности}">
```

Значение по умолчанию 0. Если значения `TABINDEX` одинаковы или вообще не заданы, обход выполняется в порядке появления элементов управления в исходном тексте Web-страницы. Если задано отрицательное значение, элемент управления исключается из последовательности.

Поддерживается IE начиная с 4.0 для элементов управления и начиная с 5.0 для остальных элементов страницы. Отрицательные значения атрибута поддерживаются начиная с 5.01.

TARGET

Задает имя фрейма или окна Web-обозревателя, куда будет загружена Web-страница, указанная в гиперссылке.

```
<A TARGET="{Имя окна или фрейма}|_blank|_parent|_search|_self|_top">
```

Кроме собственно имени окна или фрейма доступны также пять предопределенных значений:

- `_blank` — загружает Web-страницу в новое окно;
- `_parent` — загружает Web-страницу в родительское окно;

- _search — загружает Web-страницу в панель поиска Web-обозревателя (доступно в IE начиная с версии 5.0);
- _self — загружает Web-страницу в то же самое окно (значение по умолчанию);
- _top — загружает Web-страницу в самое верхнее окно в иерархии.

Поддерживается IE начиная с 3.02 для <A>, <AREA>, <BASE> и <FORM>, а начиная с 4.0 для <LINK>. Поддерживается N начиная с 2.0.

TEXT

Задает цвет текста документа.

```
<BODY TEXT="{Цвет}">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

TITLE

Задает текст "всплывающей" подсказки для элемента страницы. Эта подсказка появляется при наведении на элемент курсора мыши.

```
<P TITLE="{Текст подсказки}">
```

Поддерживается IE начиная с 4.0.

TITLE (<STYLESHEET>)

Задает заголовок для таблицы стилей. Используется только для ее идентификации.

```
<STYLESHEET TITLE="{Текст заголовка}">
```

Поддерживается IE начиная с 4.0.

TOP

Задает вертикальную координату левого верхнего угла слоя относительно родителя.

```
<LAYER TOP="{Координата Y}">
```

Поддерживается N начиная с 4.0.

TOPMARGIN

Задает расстояние в пикселах от верхней границы окна Web-обозревателя до верхней границы содержимого Web-страницы.

```
<BODY TOPMARGIN="{Расстояние}">
```

Значение по умолчанию 15.

Поддерживается IE начиная с 3.02.

TRUESPEED

Задает способ вычисления нового положения прокручиваемого текста в <MARQUEE>.

```
<MARQUEE [TRUESPEED]>
```

По умолчанию новое положение вычисляется каждые 60 миллисекунд, независимо от значений атрибутов SCROLLAMOUNT и SCROLLDELAY. Если вы хотите, чтобы новое положение текста вычислялось каждые {SCROLLDELAY} миллисекунд (более точно), включите в тег атрибут TRUESPEED. При этом текст будет двигаться более плавно, но прокрутка будет отнимать больше ресурсов у компьютера.

Поддерживается IE начиная с 4.0.

TYPE (<BUTTON>)

Задает тип кнопки <BUTTON>.

```
<BUTTON TYPE="button|reset|submit">
```

Доступны три значения:

- button — обычная кнопка (значение по умолчанию);
- reset — кнопка сброса формы;
- submit — кнопка отправки на сервер данных формы.

Поддерживается IE начиная с 4.0.

TYPE (<EMBED>, <LINK> и <OBJECT>)

Задает тип данных MIME элемента ActiveX.

```
<OBJECT TYPE="{Тип данных MIME}">
```

Поддерживается IE начиная с 3.02. Поддерживается N начиная с 2.0 для <EMBED> и начиная с 4.0 для <LINK> и <OBJECT>.

TYPE (<INPUT>)

Задает тип элемента управления: кнопка, флажок, поле ввода или что-то другое.

```
<INPUT TYPE="button|checkbox|file|hidden|image|password|radio|reset|
↳submit|text">
```

Доступны десять значений, перечисленных в табл. П1.1.

Таблица П1.1. *Значения атрибута* `TYPE` *тега* `<INPUT>`

Значение	Описание
`button`	Обычная командная кнопка
`checkbox`	Флажок
`file`	Поле ввода имени файла и кнопка **Открыть**. Позволяет отправить файл на Web-сервер
`hidden`	"Скрытое" поле. Оно никак не отображается в форме, но его значение посылается вместе с остальными данными. Может использоваться, например, для уникальной идентификации Web-страницы
`image`	Изображение. Аналогично по действию кнопке `submit`
`password`	Текстовое поле для ввода паролей. Вводимый текст отображается в виде звездочек
`radio`	Радиокнопка (кнопка-переключатель)
`reset`	Командная кнопка, аналогичная `button`, при нажатии на которую вся форма очищается. Называется также кнопкой `reset`
`submit`	Командная кнопка, аналогичная `button`, при нажатии на которую происходит отправка серверу данных, введенных в форму. Называется также кнопкой `submit`
`text`	Поле ввода

Поддерживается IE начиная с 3.02.

TYPE (<LI>, <OL> и <UL>)

Задает тип маркера маркированного и тип нумерации нумерованного списка.

```
<OL TYPE="A|a|I|i|1">
<UL TYPE="disc|circle|square">
```

Для нумерованного списка `<OL>` доступны пять значений:

- `A` — нумерует позиции списка большими латинскими буквами;
- `a` — малыми латинскими буквами;
- `I` — большими римскими цифрами;
- `i` — малыми римскими цифрами;
- `1` — арабскими цифрами (значение по умолчанию).

Для маркированного списка <UL> доступны три значения:

- disc — маркирует позиции списка сплошным кружком (значение по умолчанию);
- circle — окружностью без заливки;
- square — квадратиком.

Поддерживается IE начиная с 3.02 для <LI> и <OL> и начиная с 4.0 для <UL>. Поддерживается N начиная с 1.0 для <LI> и начиная с 1.1 для <OL> и <UL>.

TYPE (<SCRIPT>)

Задает тип интерпретатора (виртуальной машины) скриптового языка в стандарте MIME. Должен соответствовать языку, указанному атрибутом LANGUAGE.

```
<SCRIPT TYPE="{Тип интерпретатора в стандарте MIME}">
```

Доступны шесть значений, перечисленных в табл. П1.2.

Таблица П1.2. *Значения атрибута* TYPE *тега* <SCRIPT>

Значение	Описание
text/ecmascript	ECMAScript
text/Jscript	Microsoft JScript
text/javascript	Microsoft JScript
text/vbs	Microsoft VBScript
text/vbscript	Microsoft VBScript
text/xml	XML

Поддерживается IE начиная с 4.0.

TYPE (<SPACER>)

Задает тип дополнительного свободного пространства, размещаемого на странице. Это может быть дополнительное пространство между символами в строке, между строками или просто прямоугольное пространство, подобное невидимому изображению.

```
<SPACER TYPE="horizontal|vertical|block">
```

Доступны три значения:

- horizontal — добавляет дополнительное пространство между символами в строке. Используйте атрибут SIZE для его задания;

- `vertical` — добавляет дополнительное пространство между строками в абзаце. Используйте атрибут `SIZE` для его задания;
- `block` — размещает на странице невидимый прямоугольник. Используйте атрибуты `ALIGN`, `HEIGHT` и `WIDTH` для задания его параметров.

Поддерживается N начиная с 3.0.

TYPE (<STYLE>)

Задает тип таблицы стилей в стандарте MIME.

```
<STYLE TYPE="{Тип данных MIME}">
```

Для таблицы стилей он должен быть равен `text/css`. N также поддерживает таблицы стилей JavaScript; для них значение этого атрибута будет равно `text/javascript`.

Поддерживается IE и N начиная с 4.0.

UNITS

Задает единицу измерения ширины и высоты внедренного объекта `<EMBED>`.

```
<EMBED UNITS="px|em">
```

Доступны два значения:

- `px` — задает пикселы в качестве единицы измерения;
- `em` — задает в качестве единицы измерения ширину и высоту символа текущего шрифта.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

URN

Задает имя сетевого ресурса (Uniform Resource Name).

```
<A URN="{Имя сетевого ресурса}">
```

Поддерживается IE начиная с 4.0.

USEMAP

Задает имя (и, возможно, интернет-адрес) списка "горячих" областей для карты-изображения.

```
<IMG USEMAP="{Интернет-адрес и имя списка "горячих" областей}">
```

Пример использования атрибута USEMAP.

```
<MAP NAME="somemap">
. . . Список "горячих" областей
</MAP>
<IMG USEMAP="#somemap" ID=idImg SRC="someimage.gif">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

VALIGN (<CAPTION>)

Задает местонахождение заголовка таблицы: вверху или внизу.

```
<CAPTION VALIGN="top|bottom">
```

Доступны два значения:

- top — помещает заголовок вверху таблицы (значение по умолчанию);
- bottom — внизу таблицы.

Поддерживается IE начиная с 4.0.

VALIGN (элементы таблицы)

Задает вертикальное выравнивание текста в таблице и ее элементах.

```
<TD VALIGN="middle|baseline|bottom|top">
```

Доступны четыре значения:

- middle — выравнивает текст посередине элемента таблицы (значение по умолчанию);
- baseline — выравнивает базовую линию первой строки по базовой линии элемента таблицы;
- bottom — выравнивает текст по низу элемента таблицы;
- top — по верху элемента таблицы.

Поддерживается IE начиная с 3.02 для <TD> и <TR> и начиная с 4.0 для <COL>, <COLGROUP>, <TBODY>, <TFOOT>, <TH> и <THEAD>. Поддерживается N начиная с 1.1.

VALUE (<LI>)

Задает номер позиции нумерованного списка.

```
<LI VALUE="{Номер}">
```

По умолчанию Web-обозреватель сам нумерует позиции списка.

Поддерживается IE начиная с 3.02.

VALUE (<OPTION>)

Задает значение, которое будет послано формой серверу.

```
<OPTION VALUE="{Значение для передачи серверной программе}">
```

По умолчанию, если атрибут VALUE не указан, форма передает серверной программе значение, помещенное в теге <OPTION>. Если атрибут VALUE указан, форма передаст его значение.

Пример использования атрибута VALUE.

```
<SELECT>
<OPTION>Красный</OPTION>
<OPTION VALUE="yellow">Желтый</OPTION>
<OPTION>Зеленый</OPTION>
</SELECT>
```

Если пользователь выберет первый или третий пункты списка, будут переданы соответственно значения Красный и Зеленый. Если пользователь выберет второй пункт, будет передано значение yellow.

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

VALUE (<PARAM>)

Задает значение параметра внедренного объекта.

```
<PARAM VALUE="{Значение}">
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

VALUE (кнопки)

Задает надпись, помещаемую на кнопке.

```
<INPUT TYPE="reset" VALUE="{Надпись}">
```

Надпись, помещаемая на кнопке по умолчанию, зависит от программы Web-обозревателя.

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

VALUE (скрытое поле)

Задает значение, посылаемое серверной программе скрытым полем.

```
<INPUT TYPE="hidden" VALUE="{Значение}">
```

Значения по умолчанию нет.

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

VALUE (текстовые поля)

Задает начальное значение, помещаемое в текстовое поле.

```
<INPUT TYPE="text" VALUE="{Начальное значение}">
```

По умолчанию текстовое поле пусто.

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

VALUE (флажки и радиокнопки)

Задает значение, которое форма отправит серверной программе, если пользователь включит флажок или радиокнопку.

```
<INPUT TYPE="checkbox" VALUE="{Значение}">
```

По умолчанию для флажка посылается значение `on`, если он выбран. Для радиокнопки значение по умолчанию отсутствует.

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

VCARD_NAME

Задает значение одного из свойств объекта `vCard` (сведений о пользователе) для использования во всплывающем списке автозаполнения полей ввода.

```
<INPUT TYPE="text" VCARD_NAME="{Свойство объекта vCard}">
```

Все доступные для указания в этом атрибуте свойства перечислены в табл. П1.3.

Таблица П1.3. *Свойства объекта* `vCard`

Свойство	Описание
`vCard.Business.City`	Город, где расположено место работы
`vCard.Business.Country`	Страна, где расположено место работы
`vCard.Business.Fax`	Номер факса места работы
`vCard.Business.Phone`	Номер телефона места работы
`vCard.Business.State`	Штат, область или территория, где расположено место работы
`vCard.Business.StreetAddress`	Адрес по улице места работы
`vCard.Business.URL`	Адрес Web-страницы места работы
`vCard.Business.Zipcode`	Почтовый индекс места работы
`vCard.Cellular`	Номер сотового телефона

Таблица П1.3 *(окончание)*

Свойство	Описание
vCard.Company	Название фирмы, предприятия или учреждения
vCard.Department	Название отдела фирмы, предприятия или учреждения
vCard.DisplayName	Имя, которое будет отображаться в списке адресов адресной книги
vCard.Email	Адрес электронной почты
vCard.FirstName	Имя
vCard.Gender	Пол
vCard.Home.City	Город проживания
vCard.Home.Country	Страна проживания
vCard.Home.Fax	Номер домашнего факса
vCard.Home.Phone	Номер домашнего телефона
vCard.Home.State	Штат, область или территория, где расположено место проживания
vCard.Home.StreetAddress	Адрес места проживания
vCard.Home.Zipcode	Почтовый индекс места проживания
vCard.Homepage	Адрес домашней Web-страницы
vCard.JobTitle	Название должности
vCard.LastName	Фамилия
vCard.MiddleName	Отчество
vCard.Notes	Примечания
vCard.Office	Местонахождение места работы
vCard.Pager	Номер пейджера

Поддерживается IE начиная с 5.0.

VISIBILITY

Задает видимость или невидимость слоя.

```
<LAYER VISIBILITY="show|hidden">
```

Доступны три значения: `show` делает слой видимым, `hidden` — невидимым, а `inherit` "наследует" это свойство от слоя-родителя или самого документа (значение по умолчанию).

Поддерживается N начиная с 4.0.

VLINK

Задает цвет посещенных гиперссылок в документе.

```
<BODY VLINK="{Цвет}">
```

Поддерживается IE начиная с 3.02 и N начиная с 1.0.

VOLUME

Задает громкость проигрывания фонового звука `<BGSOUND>`.

```
<BGSOUND VOLUME="{Громкость}">
```

Значение громкости может быть от −10 000 до 0 (полная громкость).

Поддерживается IE начиная с 4.0.

VSPACE

Задает вертикальное расстояние в пикселах от элемента страницы до окружающего его текста.

```
<IMG VSPACE="{Вертикальный отступ}">
```

Поддерживается IE начиная с 3.02 для `<APPLET>`, `<IMG>`, `<INPUT TYPE="image">`, `<MARQUEE>` и `<OBJECT>`, а начиная с 4.0 для `<IFRAME>`. Поддерживается N начиная с 1.1.

WEIGHT

Задает "жирность" шрифта от 100 (самый светлый) до 900 (самый жирный).

```
<FONT WEIGHT="{"Жирность" шрифта}">
```

Значение по умолчанию 300.

Поддерживается N начиная с 4.0.

WIDTH

Задает ширину элемента страницы либо в пикселах, либо в процентах от доступного пространства. Если встречается в теге `<MULTICOL>`, то задает ширину отдельной колонки.

```
<TABLE WIDTH="{Ширина}">
```

Поддерживается IE начиная с 4.0 и N начиная с 1.1.

WRAP (<PRE>)

Включает перенос строк текста с заданным форматом в теге <PRE>.

```
<PRE [WRAP]>
```

По умолчанию строки текста с заданным форматом не переносятся. Если вы хотите, чтобы они переносились, включите в тег атрибут WRAP.

Поддерживается IE начиная с 5.5 и N начиная с 1.0.

WRAP (<TEXTAREA>)

Задает перенос строк в области редактирования.

```
<TEXTAREA WRAP="soft|hard|off">
```

Доступны три значения:

- soft — включает перенос строк; при этом результирующий текст не будет содержать символы возврата каретки (значение по умолчанию);
- hard — включает перенос строк и заставляет область редактирования вставлять в нужные места результирующего текста символы возврата каретки;
- off — отключает перенос строк.

Поддерживается IE начиная с 4.0 и N начиная с 2.0.

Z-INDEX

Задает порядок перекрытия слоев. Слои с большим значением этого атрибута перекрывают слои с меньшим значением.

```
<LAYER Z-INDEX="{Номер в порядке перекрытия}">
```

Поддерживается N начиная с 4.0.

Теги

Теги перечислены в алфавитном порядке. Для каждого приводится формальное описание формата с перечнем используемых атрибутов, краткая характеристика и указания о поддержке основными Web-обозревателями.

<!-- -->

Вставляет в код HTML неотображаемые Web-обозревателем комментарии.

```
<COMMENT [ID="{Имя}"] [LANG="{Код языка}"]>
. . . Текст комментария
</COMMENT>
```

Блочный парный тег. Атрибутов не имеет.

Поддерживается IE начиная с 3.0.

<!DOCTYPE>

Задает описание типа документа (DTD, Document Type Definition).

```
<!DOCTYPE {Описание типа документа}>
```

Одинарный тег, атрибутов не имеет.

Поддерживается IE начиная с 3.0.

<A>

Задает либо гиперссылку, либо "якорь" для ссылки.

```
<A HREF="{Интернет-адрес назначения}"|NAME="{Имя "якоря"}"
[ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"]
[DATAFLD="{Имя поля таблицы базы данных}]"
[DATASRC="#{Имя объекта-источника данных}]"
[DIR="ltr|rtl"] [DISABLED] [HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[METHODS="{Список методов HTTP, разделенных запятыми}"]
[REL="Alternate|Appendix|Bookmark|Chapter|Contents|Copyright|
↳Glossary|Help|Index|Next|Offline|Prev|Section|Shortcut|Icon|
↳Start|Stylesheet|Subsection"]
[REV="Alternate|Appendix|Bookmark|Chapter|Contents|Copyright|
↳Glossary|Help|Index|Next|Prev|Section|Start|Stylesheet|Subsection"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TARGET="{Имя окна или фрейма}|_blank|_parent|_search|_self|_top"]
[TITLE="{Текст подсказки}"] [URN="{Имя сетевого ресурса}"]>
. . . Содержимое гиперссылки (текст или изображение)
</A>
```

Встроенный парный тег. Обязательным является один из двух атрибутов: либо в атрибуте HREF задается интернет-адрес файла назначения (содержащего Web-страницу, рисунок или что-нибудь другое), либо в атрибуте NAME задается имя "якоря" для ссылок на нее из других страниц или другого места текущей страницы.

Поддерживается IE и N начиная с 1.0. В N реализованы только атрибуты HREF, NAME и TARGET.

<ACRONYM>

Используется для создания условных обозначений.

```
<ACRONYM [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</ACRONYM>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 4.0.

<ADDRESS>

Используется для размещения на странице адресов, предупреждений об авторских правах и т. п. IE и N отображают такой текст курсивом.

```
<ADDRESS [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</ADDRESS>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N отсутствуют атрибуты ACCESSKEY, CONTENTEDITABLE, DIR, DISABLED, HIDEFOCUS, LANGUAGE, TABINDEX и TITLE.

<APPLET>

Помещает на страницу Java-апплет.

```
<APPLET CODE="{Интернет-адрес файла с классом Java}|
SRC="{Интернет-адрес файла с классом Java}
CODEBASE="{Интернет-адрес папки, где содержится файл Java-класса}"
[ACCESSKEY="{Клавиша-ускоритель}"]
[ALIGN="left|right|top|texttop|middle|absmiddle|baseline|bottom|
↳absbottom"] [ALT="{"Альтернативный" текст}"]
```

```
[ARCHIVE="{Интернет-адрес архивного файла}"] [CLASS="{Стилевой класс}"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATASRC="#{Имя объекта-источника данных}"] [HIDEFOCUS]
[HSPACE="{Горизонтальный отступ}"] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[MAYSCRIPT] [NAME="{Имя}"] [STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [VSPACE="{Вертикальный отступ}"]>
. . . Определения параметров
</APPLET>
```

Блочный парный тег. Требует задания атрибутов CODE (интернет-адрес файла Java-класса; можно также использовать атрибут SRC) и CODEBASE (интернет-адрес папки, где содержится этот файл).

Поддерживается IE начиная с 3.0 и N начиная с 2.0. В IE не реализованы атрибуты ALT, ARCHIVE и MAYSCRIPT. N не поддерживает атрибуты ACCESSKEY, CLASS, DATAFLD, DATASRC, HIDEFOCUS, ID, LANG, LANGUAGE, SRC, STYLE, TABINDEX и TITLE. В настоящее время этот тег признан устаревшим и не рекомендован к применению, однако все еще часто используется в N.

<AREA>

Задает "горячую" область карты-изображения. Используется внутри парного тега <MAP>.

```
<AREA COORDS="{Координаты "горячей" области}"
HREF="{Интернет-адрес назначения}"|NOHREF SHAPE="circ|poly|rect"
[ACCESSKEY="{Клавиша-ускоритель}"] [ALT="{"Альтернативный" текст}"]
[CLASS="{Стилевой класс}"] [DIR="ltr|rtl"] [HIDEFOCUS] [ID="{Имя}"]
[LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TARGET="{Имя окна или фрейма}|_blank|_parent|_search|_self|_top"]
[TITLE="{Текст подсказки}"]>
```

Неотображаемый одинарный тег. Обязательные атрибуты: COORDS, задающий координаты "горячей" области в зависимости от значения атрибута SHAPE, определяющего форму "горячей" области, и атрибут HREF, указывающий интернет-адрес назначения. Вместо него можно задать атрибут NOHREF; в этом случае "горячая" область будет указывать "в никуда" (неактивная область).

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты ACCESSKEY, ALT, CLASS, DIR, HIDEFOCUS, ID, LANG, LANGUAGE, STYLE, TABINDEX и TITLE.

<B>

Используется для выделения текста жирным шрифтом.

```
<B [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</B>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N отсутствуют атрибуты.

<BASE>

Задает базовый интернет-адрес, относительно которого будут вычисляться все относительные интернет-адреса, встречающиеся в коде страницы. Задается в HTML-заголовке страницы (внутри тега <HEAD>) перед всеми гиперссылками.

```
<BASE HREF="{Базовый интернет-адрес}" [ID="{Имя}"]
[TARGET="{Имя окна или фрейма}|_blank|_parent|_search|_self|_top"]>
```

Одинарный тег. Обязательный атрибут — HREF, задающий интернет-адрес.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N отсутствует атрибут ID.

<BASEFONT>

Задает шрифт по умолчанию. Задается в HTML-заголовке страницы (внутри тега <HEAD>) или в ее теге (внутри тега <BODY>) перед любым текстом.

```
<BASEFONT [COLOR="{Цвет}"] [FACE="{Имя шрифта}"] [ID="{Имя}"]
[SIZE="{Размер шрифта}"]>
```

Одинарный тег. Обязательных атрибутов нет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты COLOR, FACE и ID. В настоящее время этот тег признан устаревшим и не рекомендован к использованию.

<BDO>

Позволяет сменить порядок чтения для выделенного текста.

```
<BDO DIR="ltr|rtl" [ACCESSKEY="{Клавиша-ускоритель}"]
[CLASS="{Стилевой класс}"] [CONTENTEDITABLE="inherit|true|false"]
[DISABLED] [HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</BDO>
```

Встроенный парный тег. Обязательный атрибут — DIR, задающий направление чтения текста.

Поддерживается IE начиная с 5.0.

<BGSOUND>

Помещает на страницу звуковой файл, проигрываемый в фоновом режиме. Задается в HTML-заголовке страницы (внутри тега <HEAD>).

```
<BGSOUND SRC="{Интернет-адрес аудиофайла}" [BALANCE="{Значение
↳стереобаланса}"]
[ID="{Имя}"] [LOOP="-1|0|{Количество}"] [VOLUME="{Громкость}"]>
```

Неотображаемый одинарный тег. Обязательный тег — SRC, задающий интернет-адрес аудиофайла.

Поддерживается IE начиная с 3.0.

<BIG>

Задает отображение текста крупным шрифтом.

```
<BIG [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</BIG>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 2.0. В N не реализованы атрибуты.

<BLINK>

Задает отображение текста мерцающим шрифтом.

```
<BLINK>
. . . Содержимое
</BLINK>
```

Встроенный парный тег. Никаких атрибутов не имеет.

Поддерживается N начиная с 1.0.

<BLOCKQUOTE>

Форматирует текст как цитату. IE и N отображают его с отступом слева.

```
<BLOCKQUOTE [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</BLOCKQUOTE>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты ACCESSKEY, CONTENTEDITABLE, DIR, DISABLED, HIDEFOCUS, LANGUAGE, TABINDEX и TITLE.

<BODY>

Задает тело документа.

```
<BODY [ACCESSKEY="{Клавиша-ускоритель}"] [ALINK="{Цвет}"]
[BACKGROUND="{Интернет-адрес файла изображения}"] [BGCOLOR="{Цвет}"]
[BGPROPERTIES="fixed"] [BOTTOMMARGIN="{Расстояние}"]
[CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[LEFTMARGIN="{Расстояние}"] [LINK="{Цвет}"] [NOWRAP]
[RIGHTMARGIN="{Расстояние}"] [SCROLL="yes|no|auto"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"] [TEXT="{Цвет}"]
[TITLE="{Текст подсказки}"] [TOPMARGIN="{Расстояние}"] [VLINK="{Цвет}"]>
. . . Содержимое страницы
</BODY>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE и N начиная с 1.0. В N не реализованы атрибуты ACCESSKEY, BGPROPERTIES, BOTTOMMARGIN, CONTENTEDITABLE, DIR, DISABLED, HIDEFOCUS, LANGUAGE, LEFTMARGIN, NOWRAP, RIGHTMARGIN, SCROLL, TABINDEX, TITLE и TOPMARGIN.

Вставляет "жесткий" разрыв строки.

```
<BR [CLASS="{Стилевой класс}"] [CLEAR="all|left|right|none"]
[ID="{Имя}"] [STYLE="{Определение встроенного стиля}"]>
```

Одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N реализован только атрибут CLEAR.

<BUTTON>

Помещает на странице кнопку с любым содержанием (таблица, рисунок, фрагмент текста с HTML-форматированием).

```
<BUTTON [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATAFORMATAS="text|html|localized-text"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NAME="{Имя}"] [STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [TYPE="button|reset|submit"]>
. . . Содержимое, отображаемое на кнопке
</BUTTON>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 4.0.

<CAPTION>

Помещает в таблицу заголовок.

```
<CAPTION [ACCESSKEY="{Клавиша-ускоритель}"]
[ALIGN="bottom|center|left|right|top"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
```

```
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [VALIGN="top|bottom"]>
. . . Собственно заголовок
</CAPTION>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.1. В N реализован только атрибут ALIGN.

<CENTER>

Используется для центрирования блока текста.

```
<CENTER [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</CENTER>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.1. N не предусматривает атрибутов. В настоящее время этот тег признан устаревшим и не рекомендован к применению.

<CITE>

Используется для создания цитат. IE отображает цитаты курсивом.

```
<CITE [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</CITE>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. N не поддерживает атрибуты.

<CODE>

Используется для помещения на страницу примеров, например, исходных текстов программ. IE и N отображают их моноширинным шрифтом.

```
<CODE [CLASS="{Стилевой класс}"] [CONTENTEDITABLE="inherit|true|false"]
[DIR="ltr|rtl"] [DISABLED] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</CODE>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты.

<COL>

Задает параметры одной или нескольких колонок таблицы. Может быть вложено в группу колонок <COLGROUP>. Атрибут SPAN задает количество колонок, на которое распространяются эти параметры.

```
<COL [ALIGN="left|center|right|justify"] [BGCOLOR="{Цвет}"]
[CLASS="{Стилевой класс}"] [DIR="ltr|rtl"] [ID="{Имя}"]
[LANG="{Код языка}"] [SPAN="{Количество колонок}"]
[STYLE="{Определение встроенного стиля}"]
[VALIGN="middle|baseline|bottom|top"]>
```

Блочный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0.

<COLGROUP>

Задает параметры группы колонок таблицы. Может содержать в себе теги <COL>. Атрибут SPAN задает количество колонок, содержащихся в группе.

```
<COLGROUP [ALIGN="left|center|right|justify"] [BGCOLOR="{Цвет}"]
[CLASS="{Стилевой класс}"] [DIR="ltr|rtl"] [ID="{Имя}"]
[LANG="{Код языка}"] [SPAN="{Количество колонок}"]
[STYLE="{Определение встроенного стиля}"]
[VALIGN="middle|baseline|bottom|top"]>
. . . Может включать теги <COL>
</COLGROUP>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0.

<COMMENT>

Вставляет в код HTML невидимые комментарии.

```
<COMMENT [ID="{Имя}"] [LANG="{Код языка}"]>
. . . Текст комментария
</COMMENT>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0.

<DD>

Помечает текст определения в списке определений <DL>.

```
<DD [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NOWRAP] [STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Текст определения
```

Блочный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты ACCESSKEY, CONTENTEDITABLE, DIR, DISABLED, HIDEFOCUS, LANGUAGE, NOWRAP, TABINDEX и TITLE.

<DEL>

Помечает удаленный из документа текст.

```
<DEL [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</DEL>
```

Встроенный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 4.0.

<DFN>

Помечает описание термина.

```
<DFN [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</DFN>
```

Встроенный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0.

<DIR>

Используется для создания списка коротких строк, аналогичного списку файлов. Включает в себя теги <LI>.

```
<DIR [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое списка
</DIR>
```

Блочный одинарный тег. Обязательных атрибутов не имеет.

Пример использования тега <DIR>.

```
<DIR>
<LI>Отчет 1999.doc
<LI>Отчет 2000.doc
<LI>Отчет 2001.doc
</DIR>
```

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты ACCESSKEY, CONTENTEDITABLE, DIR, DISABLED, HIDEFOCUS, LANGUAGE, TABINDEX и TITLE.

<DIV>

Определяет отдельный элемент страницы: простой текстовый абзац или более сложный фрагмент с HTML-форматированием. Его содержимое может быть сколь угодно сложным.

```
<DIV [ACCESSKEY="{Клавиша-ускоритель}"]
[ALIGN="left|center|right|justify"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATAFORMATAS="text|html|localized-text"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NOWRAP] [STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</DIV>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 2.0. В N не реализованы атрибуты ACCESSKEY, CONTENTEDITABLE, DATAFLD, DATAFORMATAS, DATASRC, DIR, DISABLED, HIDEFOCUS, LANGUAGE, NOWRAP, TABINDEX и TITLE.

<DL>

Используется для создания списка определений. Включает в себя теги <DT> (термин) и <DD> (определение термина).

```
<DL [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[COMPACT] [CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"]
[DISABLED] [HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое списка
</DL>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты ACCESSKEY, CONTENTEDITABLE, DIR, DISABLED, HIDEFOCUS, LANGUAGE, TABINDEX и TITLE.

<DT>

Помечает текст термина в списке определений <DL>.

```
<DT [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NOWRAP] [STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Текст термина
```

Блочный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты ACCESSKEY, CONTENTEDITABLE, DIR, DISABLED, HIDEFOCUS, LANGUAGE, NOWRAP, TABINDEX и TITLE.

<EM>

Выделяет текст терминов. IE и N отображают его курсивом.

```
<EM [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</EM>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты.

<EMBED>

Помещает на страницу нетекстовый элемент, обрабатываемый с помощью расширения Web-обозревателя.

```
<EMBED SRC="{Интернет-адрес файла данных}"
[ACCESSKEY="{Клавиша-ускоритель}"]
[ALIGN="left|right|top|texttop|middle|absmiddle|baseline|bottom|
absbottom"] [BORDER="{Толщина границы в пикселах}"]
```

```
[CLASS="{Стилевой класс}"] [FRAMEBORDER="no"] [HEIGHT="{Высота}"]
[HIDDEN="true|false"] [HIDEFOCUS] [HSPACE="{Горизонтальный отступ}"]
[ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NAME="{Имя}"] [PALETTE="foreground|background"]
[PLUGINSPAGE="{Интернет-адрес инструкций по инсталляции}"]
[PLUGINURL="{Интернет-адрес дистрибутивного файла расширения}"]
[STYLE="{Определение встроенного стиля}"]
[TITLE="{Текст подсказки}"] [TYPE="{Тип данных MIME}"] [UNITS="px|em"]
[VSPACE="{Вертикальный отступ}"] [WIDTH="{Ширина}"]>
. . . Определения параметров
</EMBED>
```

Блочный парный тег, однако, в некоторых случаях может использоваться как одинарный, без списка параметров. Требует задания атрибутов SRC (интернет-адрес файла данных) и HEIGHT (высота). Может также потребоваться задание атрибута TYPE (MIME-тип данных).

Поддерживается IE начиная с 3.0 и N начиная с 2.0. В IE не реализованы атрибуты BORDER, FRAMEBORDER, HIDDEN, HSPACE, PALETTE, PLUGINURL, TYPE и VSPACE. N не поддерживает атрибуты ACCESSKEY, CLASS, HIDEFOCUS, ID, LANG, LANGUAGE, STYLE и TITLE.

<FIELDSET>

Используется для объединения нескольких элементов управления в группу. Такая группа окружается рамкой и имеет заголовок, задаваемый тегом <LEGEND>.

```
<FIELDSET [ACCESSKEY="{Клавиша-ускоритель}"]
ALIGN="left|right|top|texttop|middle|absmiddle|baseline|bottom|
↳absbottom" [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"]
[DATAFLD="{Имя поля таблицы базы данных}"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое группы
</FIELDSET>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 4.0.

<FONT>

Изменяет шрифт содержимого текста.

```
<FONT [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[COLOR="{Цвет}"] [CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"]
[DISABLED] [FACE="{Имя шрифта}"] [HIDEFOCUS] [ID="{Имя}"]
[LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[POINT-SIZE="{Размер шрифта в пикселах}"] [SIZE="{Размер шрифта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[WEIGHT="{"Жирность" шрифта}"]>
. . . Содержимое
</FONT>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.1. В IE не реализованы атрибуты POINT-SIZE и WEIGHT. N не поддерживает атрибуты ACCESSKEY, CLASS, CONTENTEDITABLE, DIR, DISABLED, HIDEFOCUS, ID, LANG, LANGUAGE, STYLE и TABINDEX. В настоящее время этот тег признан устаревшим и не рекомендован к использованию.

<FORM>

Помещает на страницу Web-форму.

```
<FORM ACTION="{Интернет-адрес серверной программы}" METHOD="get|post"
[AUTOCOMPLETE="on|off"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[ENCTYPE="{Метод кодировки данных}"] [HIDEFOCUS] [ID="{Имя}"]
[LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NAME="{Имя}"] [STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TARGET="{Имя окна или фрейма}|_blank|_parent|_search|_self|_top"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое формы
</FORM>
```

Блочный парный тег. Обязательные атрибуты: ACTION (адрес серверной программы, которой будут пересланы введенные в форму данные) и METHOD (метод передачи данных).

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты AUTOCOMPLETE, CLASS, CONTENTEDITABLE, DIR, DISABLED, HIDEFOCUS, ID, LANG, LANGUAGE, STYLE, TABINDEX и TITLE.

<FRAME>

Помещает фрейм в набор фреймов <FRAMESET>. Атрибут NAME задает имя фрейма, а SRC — интернет-адрес Web-страницы, которая будет в нем отображаться.

```
<FRAME [ALLOWTRANSPARENCY] [BORDERCOLOR="{Цвет}"]
[CLASS="{Стилевой класс}"] [DATAFLD="{Имя поля таблицы базы данных}"]
[DATASRC="#{Имя объекта-источника данных}"] [FRAMEBORDER="1|0|yes|no"]
[HEIGHT="{Высота}"] [HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[MARGINHEIGHT="{Отступ}"] [MARGINWIDTH="{Отступ}"] [NAME="{Имя фрейма}"]
[NORESIZE] [SCROLLING="yes|no|auto"]
[SRC="{Интернет-адрес Web-страницы}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [WIDTH="{Ширина}"]>
```

Блочный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты ALLOWTRANSPARENCY, CLASS, DATAFLD, DATASRC, HEIGHT, HIDEFOCUS, ID, LANG, LANGUAGE, TABINDEX, TITLE и WIDTH.

<FRAMESET>

Задает набор фреймов.

```
<FRAMESET COLS="Список значений ширины фреймов-колонок,
↳разделенных запятыми"|
↳ROWS="Список значений высоты фреймов-строк, разделенных запятыми"
[BORDER="{Толщина границы в пикселах}"] [BORDERCOLOR="{Цвет}"]
[CLASS="{Стилевой класс}"] [FRAMEBORDER="1|0|yes|no"]
[FRAMESPACING="{Дополнительное пространство между фреймами}"] [HIDEFOCUS]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NAME="{Имя фрейма}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [WIDTH="{Ширина}"]>
. . . Список фреймов
</FRAMESET>
```

Блочный парный тег. Требует задания атрибутов ROWS (список значений высоты фреймов-строк) или COLS (список значений ширины фреймов-столбцов).

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N реализованы только атрибуты BORDER, BORDERCOLOR, COLS, FRAMEBORDER и ROWS.

<HEAD>

Задает HTML-заголовок документа.

```
<HEAD [CLASS="{Стилевой класс}"] [ID="{Имя}"] [LANG="{Код языка}"]>
. . . Содержимое заголовка
</HEAD>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE и N начиная с 1.0. В N не реализованы атрибуты CLASS, DIR и ID.

<Hn>

Форматирует текст как заголовок n-го уровня. Всего доступны шесть уровней: от <H1> до <H6>.

```
<Hn [ACCESSKEY="{Клавиша-ускоритель}"]
[ALIGN="left|center|right|justify"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</Hn>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты ACCESSKEY, CONTENTEDITABLE, DIR, DISABLED, HIDEFOCUS, LANGUAGE, TABINDEX и TITLE.

<HR>

Вставляет в текст Web-страницы горизонтальную линейку.

```
<HR [ACCESSKEY="{Клавиша-ускоритель}"]
[ALIGN="left|center|right|justify"] [CLASS="{Стилевой класс}"]
[COLOR="{Цвет}"] [HIDEFOCUS] [ID="{Имя}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NOSHADE] [SIZE="{Высота в пикселах}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [WIDTH="{Ширина}"]>
```

Блочный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. N поддерживает только атрибуты ALIGN, NOSHADE, SIZE и WIDTH.

<HTML>

Заключает в себя весь текст HTML-документа. Собственно, весь HTML-документ есть содержимое этого тега.

```
<HTML [CLASS="{Стилевой класс}"] [DIR="ltr|rtl"] [ID="{Имя}"]>
. . . Содержимое документа
</HTML>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE и N начиная с 1.0. В N не реализованы атрибуты CLASS, DIR и ID.

<I>

Используется для выделения текста курсивом.

```
<I [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</I>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты.

<IFRAME>

Вставляет на страницу "плавающий" фрейм.

```
<IFRAME [ALIGN="left|center|right"] [ALLOWTRANSPARENCY]
[BORDER="{Толщина границы в пикселах}"] [CLASS="{Стилевой класс}"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATASRC="#{Имя объекта-источника данных}"] [FRAMEBORDER="1|0|yes|no"]
[HIDEFOCUS] [HSPACE="{Горизонтальный отступ}"] [ID="{Имя}"]
[LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
```

```
[MARGINHEIGHT="{Отступ}"] [MARGINWIDTH="{Отступ}"] [NAME="{Имя фрейма}"]
[SCROLLING="yes|no|auto"] [SRC="{Интернет-адрес Web-страницы}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [VSPACE="{Вертикальный отступ}"]
[WIDTH="{Ширина}"]>
. . . Содержимое, отображаемое Web-обозревателями, не поддерживающими
"плавающие" фреймы
</IFRAME>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 4.0.

<ILAYER>

Помещает на страницу относительно позиционированный слой.

```
<ILAYER LEFT="{Координата X}"|PAGEX="{Координата X}"
TOP="{Координата Y}"|PAGEY="{Координата Y}" [ABOVE="{Имя слоя}"]
[BACKGROUND="{Интернет-адрес файла изображения}"] [BELOW="{Имя слоя}"]
[BGCOLOR="{Цвет}"] [CLIP="{X1, Y1, X2, Y2}"] [HEIGHT="{Высота}"]
[ID="{Имя}"] [SRC="{Интернет-адрес файла Web-страницы}"]
[VISIBILITY="show|hidden"] [WIDTH="{Ширина}"]
[Z-INDEX="{Номер в порядке перекрытия}"]>
. . . Содержимое слоя
</LAYER>
```

Блочный парный тег. Обязательные атрибуты: LEFT и TOP либо PAGEX и PAGEY (координаты левого верхнего угла). Если слой не имеет содержимого, определенного в HTML-коде страницы, следует задать атрибут SRC (интернет-адрес файла Web-страницы). Для задания порядка перекрытия слоя можно использовать один из атрибутов: ABOVE, BELOW или Z-INDEX.

Поддерживается N начиная с 4.0.

<IMG>

Помещает на страницу рисунок.

```
<IMG SRC="{Интернет-адрес файла рисунка}
[ALIGN="left|right|top|texttop|middle|absmiddle|baseline|bottom|
↳absbottom"] [ALT="{"Альтернативный" текст}"]
[BORDER="{Толщина границы в пикселах}"] [CLASS="{Стилевой класс}"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"]
[DYNSRC="{Интернет-адрес динамичного содержимого}"] [HEIGHT="{Высота}"]
```

```
[HIDEFOCUS] [HSPACE="{Горизонтальный отступ}"] [ID="{Имя}"]
[ISMAP] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[LOOP="-1|0|{Количество}"] [LOWSRC="{Интернет-адрес файла изображения}"]
[NAME="{Имя}"] [STYLE="{Определение встроенного стиля}"]
[start="fileopen|mouseover] [SUPPRESS="true|false"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]
[USEMAP="{Интернет-адрес и имя списка "горячих" областей}"]
[VSPACE="{Вертикальный отступ}"] [WIDTH="{Ширина}"]>
```

Встроенный одинарный тег. Обязательный атрибут — SRC, задающий интернет-адрес файла изображения.

В IE с помощью тега <IMG> можно поместить на страницу видеофильм. Для этого нужно указать интернет-адрес файла фильма в атрибуте DYNSRC. Тег <IMG> используется и для создания карт-изображений при помощи атрибутов ISMAP и USEMAP.

Если файл изображения очень велик, можно в атрибуте LOWSRC задать интернет-адрес его более компактной версии с низким качеством. При этом сначала будет запрошен и показан файл с низким качеством, а уже потом — полноразмерный.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В IE не реализован атрибут SUPPRESS. В N отсутствуют атрибуты CLASS, DATAFLD, DATASRC, DIR, DYNSRC, HIDEFOCUS, ID, LANG, LANGUAGE, LOOP, start, STYLE, TABINDEX и TITLE.

<INPUT TYPE="button">

Помещает на страницу обычную командную кнопку.

```
<INPUT TYPE="button" [ACCESSKEY="{Клавиша-ускоритель}"]
[CLASS="{Стилевой класс}"] [CONTENTEDITABLE="inherit|true|false"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATAFORMATAS="text|html|localized-text"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NAME="{Имя}"] [SIZE="{Размер}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [VALUE="{Надпись}"] [WIDTH="{Ширина}"]>
```

Встроенный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N реализованы только атрибуты NAME и VALUE.

<INPUT TYPE="checkbox">

Помещает на страницу флажок.

```
<INPUT TYPE="checkbox" [ACCESSKEY="{Клавиша-ускоритель}"] [CHECKED]
[CLASS="{Стилевой класс}"] [CONTENTEDITABLE="inherit|true|false"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NAME="{Имя}"] [SIZE="{Размер}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [VALUE="{Значение}"] [WIDTH="{Ширина}"]>
```

Встроенный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N реализованы только атрибуты CHECKED, NAME и VALUE.

<INPUT TYPE="file">

Помещает на страницу поле ввода имени файла и кнопку активизации стандартного диалогового окна открытия файла. Позволяет послать серверной программе файл.

```
<INPUT TYPE="file" [ACCESSKEY="{Клавиша-ускоритель}"]
[CLASS="{Стилевой класс}"] [CONTENTEDITABLE="inherit|true|false"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NAME="{Имя}"] [SIZE="{Размер}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [WIDTH="{Ширина}"]>
```

Встроенный одинарный тег. Обязательных атрибутов не имеет.

Для нормальной работы этого элемента управления требуется, чтобы атрибут METHOD тега формы был установлен в post, а ENCTYPE — в multipart/form-data.

Поддерживается IE начиная с 4.0 и N начиная с 2.0. В N реализованы только атрибуты NAME и VALUE.

<INPUT TYPE="hidden">

Помещает на страницу скрытое поле.

```
<INPUT TYPE="hidden" [CLASS="{Стилевой класс}"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATASRC="#{Имя объекта-источника данных}"] [HIDEFOCUS] [ID="{Имя}"]
[LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NAME="{Имя}"] [STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[VALUE="{Значение}"]>
```

Встроенный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N реализованы только атрибуты NAME и VALUE.

<INPUT TYPE="image">

Помещает на страницу изображение. Серверной программе при этом будет послана пара значений вида

```
{Имя}.x={Координата X}
{Имя}.y={Координата Y}
```

При этом координаты отсчитываются от левого верхнего угла изображения до точки, по которой пользователь щелкнул мышью.

```
<INPUT TYPE="image" SRC="{Интернет-адрес файла рисунка}
[ALIGN="left|right|top|texttop|middle|absmiddle|baseline|bottom|
↪absbottom"] [ALT="{"Альтернативный" текст}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"] [DISABLED]
[DYNSRC="{Интернет-адрес динамичного содержимого}"] [HIDEFOCUS]
[HSPACE="{Горизонтальный отступ}"] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[LOOP="-1|0|{Количество}"] [LOWSRC="{Интернет-адрес файла изображения}"]
[NAME="{Имя}"] [SIZE="{Размер}"] [start="fileopen|mouseover]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]
[VSPACE="{Вертикальный отступ}"] [WIDTH="{Ширина}"]>
```

Встроенный одинарный тег. Обязательный атрибут — SRC, задающий интернет-адрес файла изображения.

В IE с помощью тега <INPUT TYPE="image"> можно поместить на страницу видеофильм. Для этого нужно указать интернет-адрес файла фильма в атрибуте DYNSRC. Если файл изображения очень велик, можно в атрибуте LOWSRC задать интернет-адрес его более компактной версии с низким качеством. При этом сначала будет запрошен и показан файл с низким качеством, а уже потом — полноразмерный.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N реализованы только атрибуты ALIGN, NAME и SRC.

<INPUT TYPE="password">

Помещает на страницу поле ввода пароля. Любой вводимый в нем текст будет представлен в виде звездочек.

```
<INPUT TYPE="password" [ACCESSKEY="{Клавиша-ускоритель}"]
[AUTOCOMPLETE="on|off"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[MAXLENGTH="{Максимальное количество символов}"] [NAME="{Имя}"]
[READONLY] [SIZE="{Размер}"] [STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [VALUE="{Начальное значение}"]
[VCARD_NAME="{Свойство объекта vCard}"] [WIDTH="{Ширина}"]>
```

Встроенный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N реализованы только атрибуты MAXLENGTH, NAME, SIZE и VALUE.

<INPUT TYPE="radio">

Помещает на страницу радиокнопку.

```
<INPUT TYPE="radio" [ACCESSKEY="{Клавиша-ускоритель}"] [CHECKED]
[CLASS="{Стилевой класс}"] [CONTENTEDITABLE="inherit|true|false"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NAME="{Имя}"] [SIZE="{Размер}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [VALUE="{Значение}"] [WIDTH="{Ширина}"]>
```

Встроенный одинарный тег. Обязательных атрибутов не имеет.

Все радиокнопки, входящие в одну группу, должны иметь одинаковое имя NAME. Только одна радиокнопка в группе может быть включена.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N реализованы только атрибуты CHECKED, NAME и VALUE.

<INPUT TYPE="reset">

Помещает на страницу командную кнопку сброса reset. При нажатии на эту кнопку происходит очистка формы.

```
<INPUT TYPE="reset" [ACCESSKEY="{Клавиша-ускоритель}"]
[CLASS="{Стилевой класс}"] [CONTENTEDITABLE="inherit|true|false"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NAME="{Имя}"] [SIZE="{Размер}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [VALUE="{Надпись}"] [WIDTH="{Ширина}"]>
```

Встроенный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N реализованы только атрибуты NAME и VALUE.

<INPUT TYPE="submit">

Помещает на страницу командную кнопку отправки данных submit. При нажатии на эту кнопку происходит отправка данных, введенных в форму, серверной программе.

```
<INPUT TYPE="submit" [ACCESSKEY="{Клавиша-ускоритель}"]
[CLASS="{Стилевой класс}"] [CONTENTEDITABLE="inherit|true|false"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NAME="{Имя}"] [SIZE="{Размер}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [VALUE="{Надпись}"] [WIDTH="{Ширина}"]>
```

Встроенный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N реализованы только атрибуты NAME и VALUE.

<INPUT TYPE="text">

Помещает на страницу обычное поле ввода.

```
<INPUT TYPE="text" [ACCESSKEY="{Клавиша-ускоритель}"]
[AUTOCOMPLETE="on|off"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[MAXLENGTH="{Максимальное количество символов}"] [NAME="{Имя}"]
[READONLY] [SIZE="{Размер}"] [STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [VALUE="{Начальное значение}"]
[VCARD_NAME="{Свойство объекта vCard}"] [WIDTH="{Ширина}"]>
```

Встроенный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N реализованы только атрибуты MAXLENGTH, NAME, SIZE и VALUE.

<INS>

Помечает вновь добавленный в документ текст.

```
<INS [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</INS>
```

Встроенный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 4.0.

<ISINDEX>

Указывает Web-обозревателю отобразить в окне поле поиска, в котором пользователь должен будет что-то ввести.

```
<ISINDEX [ACCESSKEY="{Клавиша-ускоритель}"]
[ACTION="{Интернет-адрес серверной программы}"]
[CLASS="{Стилевой класс}"] [CONTENTEDITABLE="inherit|true|false"]
[DISABLED] [HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
```

```
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[PROMPT="{Подсказка}"] [STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]>
```

Встроенный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В IE отсутствует атрибут PROMPT. В N реализован только атрибут PROMPT. В настоящее время этот тег признан устаревшим и не рекомендован к использованию.

<KBD>

Выводит текст моноширинным шрифтом.

```
<KBD [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</KBD>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N отсутствуют атрибуты.

<KEYGEN>

Используется для генерации цифровых сертификатов стандарта кодировки DER. Помещает в форму всплывающее меню значений длины ключа. По нажатии кнопки submit генерирует закрытый и открытый ключи; закрытый ключ сохраняется в базе данных ключей на компьютере пользователя, а открытый передается серверной программе.

```
<KEYGEN [CHALLENGE="{Строка вызова}"] [NAME="{Имя}"]>
```

Встроенный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается N начиная с 1.0.

<LABEL>

Задает текстовую метку элемента управления.

```
<LABEL FOR="{Имя элемента управления}" [ACCESSKEY="{Клавиша-ускоритель}"]
[CLASS="{Стилевой класс}"] [CONTENTEDITABLE="inherit|true|false"]
```

```
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATAFORMATAS="text|html|localized-text"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Текст метки
</LABEL>
```

Встроенный парный тег. Обязательный атрибут — FOR, задающий имя элемента управления, к которому привязывается метка. Внимание: это имя должно быть задано атрибутом ID!

Поддерживается IE начиная с 4.0.

<LAYER>

Помещает на страницу абсолютно позиционированный слой.

```
<LAYER LEFT="{Координата X}"|PAGEX="{Координата X}"
TOP="{Координата Y}"|PAGEY="{Координата Y}" [ABOVE="{Имя слоя}"]
[BACKGROUND="{Интернет-адрес файла изображения}"] [BELOW="{Имя слоя}"]
[BGCOLOR="{Цвет}"] [CLIP="{X1, Y1, X2, Y2}"] [HEIGHT="{Высота}"]
[ID="{Имя}"] [SRC="{Интернет-адрес файла Web-страницы}"]
[VISIBILITY="show|hidden"] [WIDTH="{Ширина}"]
[Z-INDEX="{Номер в порядке перекрытия}"]>
. . . Содержимое слоя
</LAYER>
```

Блочный парный тег. Обязательные атрибуты: LEFT и TOP либо PAGEX и PAGEY (координаты левого верхнего угла). Если слой не имеет содержимого, определенного в HTML-коде страницы, следует задать атрибут SRC (интернет-адрес файла Web-страницы). Для задания порядка перекрытия слоя можно использовать один из атрибутов: ABOVE, BELOW или Z-INDEX.

Поддерживается N начиная с 4.0.

<LEGEND>

Задает текстовую метку группе элемента управления <FIELDSET>. Должен быть первым тегом в группе.

```
<LEGEND [ACCESSKEY="{Клавиша-ускоритель}"]
[ALIGN="bottom|center|left|right|top"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
```

```
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Текст метки
</LEGEND>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 4.0.

<LI>

Помечает позицию списка.

```
<LI [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [TYPE="A|a|I|i|1|disc|circle|square"]
[VALUE="{Номер}"]>
. . . Текст позиции
```

Встроенный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты ACCESSKEY, CONTENTEDITABLE, DIR, DISABLED, HIDEFOCUS, LANGUAGE, TABINDEX и TITLE.

<LINK>

Устанавливает связь текущей страницы с другими файлами. Может находиться только в секции HTML-заголовка <HEAD>.

```
<LINK HREF="{Интернет-адрес назначения}"
REL="Alternate|Appendix|Bookmark|Chapter|Contents|Copyright|
↳Glossary|Help|Index|Next|Offline|Prev|Section|Shortcut Icon|
↳Start|Stylesheet|Subsection|fontdef|stylesheet"|
REV="Alternate|Appendix|Bookmark|Chapter|Contents|Copyright|Glossary|
↳Help|Index|Next|Prev|Section|Start|Stylesheet|Subsection"
[ID="{Имя}"] [NAME="{Имя}"]
[TARGET="{Имя окна или фрейма}|_blank|_parent|_search|_self|_top"]
[TYPE="{Тип данных MIME}"]>
```

Блочный одинарный тег. Обязательные атрибуты: HREF, задающий интернет-адрес назначения, и либо REL, указывающий, как относится документ с ин-

тернет-адресом HREF к текущей странице, либо REV, указывающий обратное отношение.

Поддерживается IE начиная с 3.0 и N начиная с 4.0. В N не реализованы атрибуты REV, ID, NAME и TARGET.

<LISTING>

Выводит блок текста моноширинным шрифтом.

```
<LISTING [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</LISTING>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0. В настоящее время признан устаревшим и не рекомендован к использованию.

<MAP>

Задает список "горячих" областей карты-изображения. Собственно "горячие" области описываются внутри него тегами <AREA>.

```
<MAP NAME="{Имя}" [CLASS="{Стилевой класс}"] [DIR="ltr|rtl"]
[ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"] [TITLE="{Текст подсказки}"]>
. . . Описания "горячих" областей
</MAP>
```

Блочный парный тег. Обязательный атрибут — NAME, задающий имя списка "горячих" областей для использования атрибутом USEMAP тега <IMG>.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N предусмотрен только атрибут NAME.

<MARQUEE>

Помещает на страницу прокручивающийся текст.

```
<MARQUEE [ACCESSKEY="{Клавиша-ускоритель}"]
[BEHAVIOR="scroll|alternate|slide"] [BGCOLOR="{Цвет}"]
```

```
[CLASS="{Стилевой класс}"] [CONTENTEDITABLE="inherit|true|false"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATAFORMATAS="text|html|localized-text"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"]
[DIRECTION="{left|right|up|down}"] [DISABLED] [HEIGHT="{Высота}"]
[HIDEFOCUS] [HSPACE="{Горизонтальный отступ}"]
[ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[LOOP="-1|0|{Количество}"] [SCROLLAMOUNT="{Шаг в пикселах}"]
[SCROLLDELAY="{Задержка в миллисекундах}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [TRUESPEED] [VSPACE="{Вертикальный отступ}"]
[WIDTH="{Ширина}"]>
. . . Прокручиваемый текст
</MARQUEE>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0.

<MENU>

Используется для создания неупорядоченного списка.

```
<MENU [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое списка
</MENU>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты ACCESSKEY, CONTENTEDITABLE, DIR, DISABLED, HIDEFOCUS, TABINDEX и TITLE. В настоящее время признан устаревшим и не рекомендован к использованию.

<META>

Служит для размещения дополнительной информации о Web-странице: описания, списка ключевых слов для поисковых машин, сведений о программе, в которой была создана Web-страница, и т. п. Может находиться только в HTML-заголовке документа (внутри тега <HEAD>).

```
<META CONTENT="{Метаданные}"
HTTP-EQUIV="description|refresh|url|mimetype|charset|Expires|
↳Refresh|Set-Cookie"|
NAME="Author|Description|Generator|Keywords|ProgID|Robots|Template|
↳{Тип}">
```

Блочный одинарный тег. Обязательные атрибуты: HTTP-EQUIV или NAME, задающие тип метаданных, и CONTENT, где указываются сами метаданные нужного типа.

Примеры использования тега <META>.

```
<META HTTP-EQUIV="refresh" CONTENT="2">
```

Эта конструкция заставляет Web-обозреватель обновлять (перезагружать с сайта) текущую Web-страницу каждые две секунды.

```
<META HTTP-EQUIV="Content-Type"
↳CONTENT="text/html; CHARSET=Windows-1251">
```

А это стандартный тег задания кодировки текста. В данном случае задана кодировка Windows 1251 — стандартная кодировка русского языка в системах Windows.

Поддерживается IE начиная с 3.0 и N начиная с 1.0.

<MULTICOL>

Выводит текст в несколько колонок.

```
<MULTICOL COLS="{Количество колонок}"
[GUTTER="{Расстояние между колонками}"] [WIDTH="{Ширина}"]>
. . . Содержимое, выводимое в несколько колонок
</MULTICOL>
```

Блочный парный тег. Требует задания атрибута COLS — количества колонок.

Поддерживается N начиная с 3.0.

<NOBR>

Выводит текст в одну строку без разрывов.

```
<NOBR [CLASS="{Стилевой класс}"] [CONTENTEDITABLE="inherit|true|false"]
[DIR="ltr|rtl"] [DISABLED] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]>
. . . Содержимое
</NOBR>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 4.0 и N начиная с 1.1. В N отсутствуют атрибуты.

<NOEMBED>

Задает текст, который будет выводиться Web-обозревателями, не поддерживающими расширения.

```
<NOEMBED>
. . . Заменяющий текст
</NOEMBED>
```

Блочный парный тег.

Поддерживается N начиная с 2.0.

<NOFRAMES>

Задает текст, который будет выводиться Web-обозревателями, не поддерживающими фреймы. Может применяться только на страницах, описывающих набор фреймов.

```
<NOFRAMES [ID="{Имя}"]>
. . . Заменяющий текст
</NOFRAMES>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0.

<NOLAYER>

Задает текст, который будет выводиться Web-обозревателями, не поддерживающими слои.

```
<NOLAYER>
. . . Заменяющий текст
</NOLAYER>
```

Блочный парный тег.

Поддерживается N начиная с 4.0.

<NOSCRIPT>

Задает текст, который будет выводиться Web-обозревателями, не поддерживающими скрипты.

```
<NOSCRIPT [ID="{Имя}"]>
. . . Заменяющий текст
</NOSCRIPT>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 4.0 и N начиная с 3.0. N не поддерживает атрибут ID.

<OBJECT>

Помещает на страницу внедренный объект: элемент ActiveX или Java-класс.

```
<OBJECT CLASSID="{Уникальный идентификатор класса}"
[ACCESSKEY="{Клавиша-ускоритель}"]
[ALIGN="left|right|top|texttop|middle|absmiddle|baseline|bottom|
↳absbottom"] [CLASS="{Стилевой класс}"]
[CODE="{Интернет-адрес файла дистрибутива}"]
[CODEBASE="{Интернет-адрес папки, где содержится дистрибутив}"]
[CODETYPE="{Тип данных MIME}"] [DATA="{Интернет-адрес файла данных}"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"]
[HEIGHT="{Высота}"] [HIDEFOCUS] [HSPACE="{Горизонтальный отступ}"]
[ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NAME="{Имя}"] [STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TYPE="{Тип данных MIME}"] [TITLE="{Текст подсказки}"]
[VSPACE="{Вертикальный отступ}"] [WIDTH="{Ширина}"]>
. . . Определения параметров
</OBJECT>
```

Блочный парный тег. Требует задания атрибута CLASSID — уникального идентификатора класса.

Поддерживается IE начиная с 3.0 и N начиная с 4.0. В N реализованы атрибуты ACCESSKEY, CLASS, CODE, CODETYPE, DATAFLD, DATASRC, HIDEFOCUS, HSPACE, LANG, LANGUAGE, NAME, STYLE, TABINDEX, TITLE и VSPACE.

<OL>

Используется для создания нумерованного списка.

```
<OL [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[COMPACT] [CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"]
[DISABLED] [HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[START="{Начальный номер}"] [STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [TYPE="A|a|I|i|1"]>
. . . Содержимое списка
</OL>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты ACCESSKEY, COMPACT, CONTENTEDITABLE, DIR, DISABLED, HIDEFOCUS, LANGUAGE, TABINDEX и TITLE.

<OPTION>

Задает пункт списка или всплывающего меню <SELECT>.

```
<OPTION [CLASS="{Стилевой класс}"] [DIR="ltr|rtl"]
[ID="{Имя}"][LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[SELECTED] [VALUE="{Значение для передачи серверной программе}"]>
. . . Текст пункта
```

Блочный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N допустимы только значения SELECTED и VALUE.

<P>

Определяет текстовый абзац.

```
<P [ACCESSKEY="{Клавиша-ускоритель}"]
[ALIGN="left|center|right|justify"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</P>
```

Блочный парный тег, однако, в некоторых случаях может использоваться как одинарный. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты ACCESSKEY, CONTENTEDITABLE, DIR, DISABLED, HIDEFOCUS, LANGUAGE, TABINDEX и TITLE.

<PARAM>

Задает параметр для внедренных объектов <APPLET>, <EMBED> и <OBJECT>.

```
<PARAM NAME="{Имя}" [DATAFLD="{Имя поля таблицы базы данных}"]
[DATAFORMATAS="text|html|localized-text"]
[DATASRC="#{Имя объекта-источника данных}"] [VALUE="{Значение}"]>
```

Блочный одинарный тег. Обязательный атрибут — NAME, задающий имя параметра. Для задания значения параметра применяется либо атрибут VALUE, либо привязка к данным.

Поддерживается IE начиная с 3.0 и N начиная с 2.0. В N не реализованы атрибуты DATAFLD, DATAFORMATAS и DATASRC.

<PLAINTEXT> (для IE)

Выводит блок текста моноширинным шрифтом без обработки HTML-тегов.

```
<PLAINTEXT [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</PLAINTEXT>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0. В настоящее время признан устаревшим и не рекомендован к использованию.

<PLAINTEXT> (для N)

Выводит остальной текст документа без обработки HTML-тегов.

```
<PLAINTEXT>
. . . Остальной текст документа
```

Блочный одинарный тег. Никаких атрибутов не имеет.

Поддерживается N начиная с 1.0. В настоящее время признан устаревшим и не рекомендован к использованию.

<PRE>

Выводит блок текста моноширинным шрифтом с сохранением всего заданного форматирования.

```
<PRE [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[COLS="{Ширина в символах}"] [CONTENTEDITABLE="inherit|true|false"]
[DIR="ltr|rtl"] [DISABLED] [HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
```

```
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [WRAP]|[WRAP="soft|hard|off"]>
. . . Содержимое
</PRE>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В IE не реализованы атрибуты COLS и WRAP без значения. В N не предусмотрены атрибуты ACCESSKEY, CONTENTEDITABLE, DIR, DISABLED, HIDEFOCUS, LANGUAGE, TABINDEX, TITLE и WRAP со значением.

<Q>

Форматирует текст как цитату.

```
<Q [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</Q>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 4.0.

<RT>

Задает текст аннотации для элемента <RUBY>.

```
<RT [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NAME="{Имя}"] [STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Текст сноски
```

Встроенный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 5.0.

<RUBY>

Применяется для создания аннотации к какому-либо фрагменту текста. Аннотация может при этом появляться выше текста или рядом с ним.

```
<RUBY [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NAME="{Имя}"] [STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</RUBY>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Пример использования тега <RUBY>.

```
<RUBY>Это основной текст.
<RT>А это аннотация.
</RUBY>
```

Поддерживается IE начиная с 5.0.

<S>

Выводит текст зачеркнутым.

```
<S [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</S>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE и N начиная с 3.0. В N отсутствуют атрибуты. Этот тег признан устаревшим и не рекомендован к использованию.

<SAMP>

Форматирует текст как пример исходного кода программы.

```
<SAMP [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
```

```
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</SAMP>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0.

<SCRIPT>

Используется для помещения на страницу скриптовой программы, исполняемой Web-обозревателем.

```
<SCRIPT [DEFER] [EVENT="{Тип события}"] [FOR="{Имя элемента страницы}"]
[ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[SRC="{Интернет-адрес файла с текстом скрипта}"]
[TYPE="{Тип интерпретатора в стандарте MIME}"]>
. . . Исходный текст скриптовой программы
</SCRIPT>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N реализованы только атрибуты LANGUAGE и SRC.

<SELECT>

Помещает на страницу всплывающее меню или список с набором пунктов. Пункты определяются внутри тега <SELECT> с помощью тегов <OPTION>.

```
<SELECT SIZE="{Количество одновременно видимых пунктов}"
[ACCESSKEY="{Клавиша-ускоритель}"]
[ALIGN="left|right|top|texttop|middle|absmiddle|baseline|bottom|
↳absbottom"] [CLASS="{Стилевой класс}"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[MULTIPLE] [NAME="{Имя}"] [STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[VALUE="{Значение для передачи серверной программе}"]>
. . . Список пунктов
</SELECT>
```

Встроенный парный тег. Обязательный атрибут — SIZE, задающий количество одновременно отображаемых пунктов. Если его значение равно 1, выводится всплывающее меню, если больше 1 — список.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N реализованы только атрибуты MULTIPLE, NAME и SIZE.

<SERVER>

Задает серверный скрипт, т. е. скрипт, исполняемый Web-сервером перед тем, как отправить Web-страницу клиенту. Описание серверных скриптов и их создания выходит за рамки этой книги.

```
<SERVER>
. . . Текст серверного скрипта
</SERVER>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается N начиная с 1.0.

<SMALL>

Задает отображение текста мелким шрифтом.

```
<SMALL [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</SMALL>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 2.0. В N отсутствуют атрибуты.

<SPACER>

Задает дополнительное свободное пространство между символами в строке, между строками или просто невидимый прямоугольник.

```
<SPACER ALIGN="left|right|top|texttop|middle|absmiddle|baseline|bottom|
absbottom" HEIGHT="{Высота}" SIZE="{Дополнительное расстояние}"
TYPE="horizontal|vertical|block" WIDTH="{Ширина}">
```

Встроенный одинарный тег. Требует задания атрибута TYPE — тип свободного пространства. Если значение атрибута равно horizontal или vertical, требуется задание также атрибута SIZE; если block — атрибутов ALIGN, HEIGHT и WIDTH.

Примеры использования тега <SPACER>.

```
Здесь будет <SPACER TYPE="horizontal" SIZE="30">пропуск.
<SPACER TYPE="vertical" SIZE="20">Здесь — тоже.
<SPACER TYPE="block" ALIGN="middle" HEIGHT="100" WIDTH="200">
```

Первый тег <SPACER> вставляет пробел в 30 пикселов перед словом "пропуск". Второй тег — дополнительное пространство между строками. Третий помещает на страницу пустой прямоугольник размером 200×100 пикселов.

Поддерживается N начиная с 3.0.

<SPAN>

Определяет встроенный элемент страницы, например, фрагмент текстового абзаца.

```
<SPAN [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATAFORMATAS="text|html|localized-text"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</SPAN>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 4.0. В N реализованы только атрибуты CLASS, ID и STYLE.

<STRIKE>

Выводит текст зачеркнутым. Аналогичен <S>.

```
<STRIKE [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
```

```
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</STRIKE>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE и N начиная с 3.0. В N не реализованы атрибуты. Признан устаревшим и не рекомендован к использованию.

<STRONG>

Применяется для выделения текста. IE и N отображают такой текст жирным шрифтом.

```
<STRONG [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</STRONG>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты.

<STYLE>

Задает таблицу стиля для Web-страницы. Может присутствовать только в HTML-заголовке (внутри тега <HEAD>).

```
<STYLE [MEDIA="screen|print|all"] [TYPE="{Тип данных MIME}"]>
. . . Описания стилей
</STYLE>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 4.0. В N не реализован атрибут MEDIA.

<SUB>

Отображает текст нижним индексом.

```
<SUB [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</SUB>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 2.0. В N отсутствуют атрибуты.

<SUP>

Отображает текст верхним индексом.

```
<SUP [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</SUP>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 2.0. В N отсутствуют атрибуты.

<TABLE>

Помещает на Web-страницу таблицу.

```
<TABLE [ACCESSKEY="{Клавиша-ускоритель}"] [ALIGN="left|center|right"]
[BACKGROUND="{Интернет-адрес файла изображения}"] [BGCOLOR="{Цвет}"]
[BORDER="{Толщина границы в пикселах}"] [BORDERCOLOR="{Цвет}"]
[CELLPADDING="{Расстояние}"] [CELLSPACING="{Расстояние}"]
[CLASS="{Стилевой класс}"] [COLS="{Количество колонок}"]
[DATAFLD="{Имя поля таблицы базы данных}"]
```

```
[DATAPAGESIZE="{Размер страницы данных}"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"]
[FRAME="none|above|below|hsides|lhs|rhs|vsides|vsides|box"]
[HEIGHT="{Высота}"] [HIDEFOCUS] [HSPACE="{Горизонтальный отступ}"]
[ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[RULES="none|rows|cols|groups|all"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [VSPACE="{Вертикальный отступ}"]
[WIDTH="{Ширина}"]>
. . . Определения строк таблицы
</TABLE>
```

Блочный парный тег. Обязательных атрибутов не имеет, за исключением атрибута DATAPAGESIZE, который обязателен при привязке таблицы к данным.

Поддерживается IE начиная с 3.0 и N начиная с 1.1. В IE не реализованы атрибуты HSPACE и VSPACE. В N отсутствуют атрибуты ACCESSKEY, BACKGROUND, BORDERCOLOR, CLASS, DATAFLD, DATAPAGESIZE, DATASRC, FRAME, HIDEFOCUS, ID, LANG, LANGUAGE, RULES, STYLE, TABINDEX и TITLE.

<TBODY>

Используется для обозначения тела таблицы, т. е. массива строк, где отображаются полезные данные. Может появляться лишь внутри тега <TABLE> и только однажды. Необходим, если таблица привязана к данным.

```
<TBODY [ACCESSKEY="{Клавиша-ускоритель}"]
[ALIGN="left|center|right|justify"] [BGCOLOR="{Цвет}"]
[CLASS="{Стилевой класс}"] [DIR="ltr|rtl"] [HIDEFOCUS]
[ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [VALIGN="middle|baseline|bottom|top"]>
. . . Определения строк таблицы
</TBODY>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0.

<TD>

Задает ячейку таблицы и ее содержимое. Может появляться только внутри тега <TR>.

```
<TD [ACCESSKEY="{Клавиша-ускоритель}"]
[ALIGN="left|center|right|justify"]
[BACKGROUND="{Интернет-адрес файла изображения}"] [BGCOLOR="{Цвет}"]
[BORDERCOLOR="{Цвет}"] [CLASS="{Стилевой класс}"]
[COLSPAN="{Количество колонок, объединяемых в одну}"] [DIR="ltr|rtl"]
[HEIGHT="{Высота}"] [HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NOWRAP] [ROWSPAN="{Количество строк, объединяемых в одну}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [VALIGN="middle|baseline|bottom|top"]
[WIDTH="{Ширина}"]>
. . . Содержимое ячейки таблицы
</TD>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.1. В N не реализованы атрибуты ACCESSKEY, BACKGROUND, BORDERCOLOR, CLASS, DIR, FRAME, HIDEFOCUS, ID, LANG, LANGUAGE, STYLE, TABINDEX и TITLE.

<TEXTAREA>

Помещает на страницу многострочную область редактирования текста.

```
<TEXTAREA [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[COLS="{Ширина в символах}"] [CONTENTEDITABLE="inherit|true|false"]
[DATAFLD="{Имя поля таблицы базы данных}"]
[DATASRC="#{Имя объекта-источника данных}"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NAME="{Имя}"] [READONLY] [ROWS="{Высота в строках}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [WRAP="soft|hard|off"]>
. . . Начальное содержимое области редактирования
<TEXTAREA>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N реализованы только атрибуты COLS, NAME, ROWS и WRAP.

<TFOOT>

Используется для обозначения основания таблицы, т. е. строки или строк внизу, в которых отображаются итоги или примечания. Может появляться лишь внутри тега <TABLE> и только однажды.

```
<TFOOT [ACCESSKEY="{Клавиша-ускоритель}"]
[ALIGN="left|center|right|justify"] [BGCOLOR="{Цвет}"]
[CLASS="{Стилевой класс}"] [DIR="ltr|rtl"] [HIDEFOCUS]
[ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [VALIGN="middle|baseline|bottom|top"]>
. . . Определения строк таблицы
</TFOOT>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0.

<TH>

Задает ячейку заголовка таблицы. Содержимое такой ячейки выравнивается по центру и отображается жирным шрифтом. Может появляться только внутри тега <TR>.

```
<TH [ACCESSKEY="{Клавиша-ускоритель}"]
[ALIGN="left|center|right|justify"]
[BACKGROUND="{Интернет-адрес файла изображения}"] [BGCOLOR="{Цвет}"]
[BORDERCOLOR="{Цвет}"] [CLASS="{Стилевой класс}"]
[COLSPAN="{Количество колонок, объединяемых в одну}"] [DIR="ltr|rtl"]
[HEIGHT="{Высота}"] [HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[NOWRAP] [ROWSPAN="{Количество строк, объединяемых в одну}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [VALIGN="middle|baseline|bottom|top"]
[WIDTH="{Ширина}"]>
. . . Содержимое ячейки таблицы
</TH>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.1. В N не реализованы атрибуты ACCESSKEY, BACKGROUND, BORDERCOLOR, CLASS, DIR, FRAME, HIDEFOCUS, ID, LANG, LANGUAGE, STYLE, TABINDEX и TITLE.

<THEAD>

Используется для обозначения заголовка таблицы. Может появляться лишь внутри тега <TABLE> и только однажды.

```
<THEAD [ACCESSKEY="{Клавиша-ускоритель}"]
[ALIGN="left|center|right|justify"] [BGCOLOR="{Цвет}"]
[CLASS="{Стилевой класс}"] [DIR="ltr|rtl"] [HIDEFOCUS]
[ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [VALIGN="middle|baseline|bottom|top"]>
. . . Определения строк таблицы
</THEAD>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0.

<TITLE>

Определяет заголовок Web-страницы. Может применяться только в HTML-заголовке (внутри тега <HEAD>).

```
<TITLE [ID="{Имя}"] [LANG="{Код языка}"]>
. . . Текст заголовка
<TITLE>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE и N начиная с 1.0. В N не реализованы атрибуты ID и LANG.

<TR>

Задает строку таблицы. Может появляться только внутри тегов <TABLE>, <THEAD>, <TBODY> и <TFOOT>.

```
<TR [ACCESSKEY="{Клавиша-ускоритель}"]
[ALIGN="left|center|right|justify"] [BGCOLOR="{Цвет}"]
[BORDERCOLOR="{Цвет}"] [CLASS="{Стилевой класс}"] [DIR="ltr|rtl"]
[HEIGHT="{Высота}"] [HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [VALIGN="middle|baseline|bottom|top"]
[WIDTH="{Ширина}"]>
. . . Определения ячеек таблицы
</TR>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.1. В N реализованы только атрибуты ALIGN, BGCOLOR и VALIGN.

<TT>

Выводит текст моноширинным шрифтом.

```
<TT [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</TT>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N отсутствуют атрибуты.

<U>

Выводит текст подчеркнутым.

```
<U [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</U>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE и N начиная с 3.0. В N не реализованы атрибуты. В настоящее время признан устаревшим и не рекомендован к использованию.

<UL>

Используется для создания маркированного списка.

```
<UL [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[COMPACT] [CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"]
[DISABLED] [HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
```

```
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"] [TYPE="disc|circle|square"]>
. . . Содержимое списка
</UL>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N не реализованы атрибуты ACCESSKEY, COMPACT, CONTENTEDITABLE, DIR, DISABLED, HIDEFOCUS, LANGUAGE, TABINDEX и TITLE.

<VAR>

Используется для форматирования имен переменных и функций языков программирования. IE и N выводят их курсивом.

```
<VAR [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</VAR>
```

Встроенный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. В N отсутствуют атрибуты.

<WBR>

Задает точку в тексте, отформатированном с использованием тега <NOBR>, где будет возможен разрыв строки. Если строка текста помещается по ширине в родительском элементе, разрыва не произойдет. Этот тег называется "мягким" разрывом строки в отличие от "жесткого", устанавливаемого тегом <BR>.

```
<WBR [ID="{Имя}"]>
```

Встроенный одинарный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.1. В N не реализован атрибут ID.

<XML>

Помещает фрагмент XML-кода в HTML-код. XML (eXtensible Markup Language, расширенный язык разметки) — это возможный преемник HTML, используемый для форматирования разнородных данных. Описание языка XML выходит за рамки этой книги.

```
<XML [ID="{Имя}"] [SRC="{Интернет-адрес файла с XML-кодом}"]>
. . . XML-код
</XML>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 5.0.

<XMP>

Выводит блок текста моноширинным шрифтом с сохранением всего форматирования и без учета HTML-тегов.

```
<XMP [ACCESSKEY="{Клавиша-ускоритель}"] [CLASS="{Стилевой класс}"]
[CONTENTEDITABLE="inherit|true|false"] [DIR="ltr|rtl"] [DISABLED]
[HIDEFOCUS] [ID="{Имя}"] [LANG="{Код языка}"]
[LANGUAGE="JScript|javascript|vbs|vbscript|XML|{Код языка скрипта}"]
[STYLE="{Определение встроенного стиля}"]
[TABINDEX="{Порядковый номер в последовательности}"]
[TITLE="{Текст подсказки}"]>
. . . Содержимое
</XMP>
```

Блочный парный тег. Обязательных атрибутов не имеет.

Поддерживается IE начиная с 3.0 и N начиная с 1.0. N не поддерживает атрибуты ACCESSKEY, CONTENTEDITABLE, DIR, DISABLED, HIDEFOCUS, LANGUAGE, TABINDEX и TITLE. В настоящее время этот тег признан устаревшим и не рекомендован к использованию.

Специальные символы HTML

В табл. П1.4 представлены коды специальных символов, доступные для использования в HTML-документах.

Таблица П1.4. *Специальные символы HTML*

Символ	Десятичный код	Имя	Описание
"	&#34;	&quot;	Кавычка

Таблица П1.4 (продолжение)

Символ	Десятичный код	Имя	Описание
&	&#38;	&amp;	Амперсанд
<	&#60;	&lt;	Знак "меньше"
>	&#62;	&gt;	Знак "больше"
			Неразрывный пробел (по нему никогда не производится перенос строки)
¡	&#161;	&iexcl;	Перевернутый восклицательный знак
¢	&#162;	&cent;	Цент
£	&#163;	&pound;	Фунт стерлингов
¤	&#164;	&curren;	Общий символ валюты
¥	&#165;	&yen;	Йена
¦	&#166;	¦ или &brkbar;	Разорванная вертикальная черта
§	&#167;	&sect;	Параграф
¨	&#168;	¨ or ¨	Умляут
©	&#169;	&copy;	Знак авторского права
ª	&#170;	&ordf;	Знак женского рода
«	&#171;	&laquo;	Открывающая угловая кавычка
¬	&#172;	&not	Знак логической инверсии
	&#173;	&shy;	"Мягкий" перенос
®	&#174;	&reg;	Зарегистрированная торговая марка
¯	&#175;	¯ или &hibar;	Знак долготы над гласным
°	&#176;	&deg;	Градус
±	&#177;	&plusmn;	Плюс-минус
²	&#178;	&sup2;	Квадрат (вторая степень)
³	&#179;	&sup3;	Куб (третья степень)
´	&#180;	&acute;	Ударение
µ	&#181;	&micro;	Микро (мю)
¶	&#182;	&para;	Конец абзаца

Таблица П1.4 *(продолжение)*

Символ	Десятичный код	Имя	Описание
·	&#183;	&middot;	Точка сверху
¸	&#184;	&cedil;	Седиль
¹	&#185;	&sup1;	Первая степень
º	&#186;	&ordm;	Знак мужского рода
»	&#187;	&raquo;	Закрывающая угловая кавычка
¼	&#188;	&frac14;	Одна четвертая
½	&#189;	&frac12;	Одна вторая
¾	&#190;	&frac34;	Три четвертых
¿	&#191;	&iquest;	Перевернутый вопросительный знак
À	&#192;	&Agrave;	Большая A глухая
Á	&#193;	&Aacute;	Большая A звонкая
Â	&#194;	&Acirc;	Большая A с циркумфлексом
Ã	&#195;	&Atilde;	Большая A с тильдой
Ä	&#196;	&Auml;	Большая A с умляутом
Å	&#197;	&Aring;	Большая A с кружком
Æ	&#198;	&AElig;	Большая лигатура AE
Ç	&#199;	&Ccedil;	Большая C с седилью
È	&#200;	&Egrave;	Большая E глухая
É	&#201;	&Eacute;	Большая E звонкая
Ê	&#202;	&Ecirc;	Большая E с циркумфлексом
Ë	&#203;	&Euml;	Большая E с умляутом
Ì	&#204;	&Igrave;	Большая I глухая
Í	&#205;	&Iacute;	Большая I звонкая
Î	&#206;	&Icirc;	Большая I с циркумфлексом
Ï	&#207;	&Iuml;	Большая I с умляутом
Ð	&#208;	&ETH;	Большая исландская "Эт"
Ñ	&#209;	&Ntilde;	Большая N с тильдой
Ò	&#210;	&Ograve;	Большая O глухая
Ó	&#211;	&Oacute;	Большая O звонкая

Таблица П1.4 (продолжение)

Символ	Десятичный код	Имя	Описание
Ô	&#212;	&Ocirc;	Большая O с циркумфлексом
Õ	&#213;	&Otilde;	Большая O с тильдой
Ö	&#214;	&Ouml;	Большая O с умляутом
×	&#215;	&times;	Знак умножения
Ø	&#216;	&Oslash;	Большая O зачеркнутая
Ù	&#217;	&Ugrave;	Большая U глухая
Ú	&#218;	&Uacute;	Большая U звонкая
Û	&#219;	&Ucirc;	Большая U с циркумфлексом
Ü	&#220;	&Uuml;	Большая U с умляутом
Ý	&#221;	&Yacute;	Большая Y звонкая
Þ	&#222;	&THORN;	Большая исландская "Торн"
ß	&#223;	&szlig;	Лигатура ss
à	&#224;	&agrave;	Малая A глухая
á	&#225;	&aacute;	Малая A звонкая
â	&#226;	&acirc;	Малая A с циркумфлексом
ã	&#227;	&atilde;	Малая A с тильдой
ä	&#228;	&auml;	Малая A с умляутом
å	&#229;	&aring;	Малая A с кружком
æ	&#230;	&aelig;	Малая лигатура AE
ç	&#231;	&ccedil;	Малая C с седилью
è	&#232;	&egrave;	Малая E глухая
é	&#233;	&eacute;	Малая E звонкая
ê	&#234;	&ecirc;	Малая E с циркумфлексом
ë	&#235;	&euml;	Малая E с умляутом
ì	&#236;	&igrave;	Малая I глухая
í	&#237;	&iacute;	Малая I звонкая
î	&#238;	&icirc;	Малая I с циркумфлексом
ï	&#239;	&iuml;	Малая I с умляутом
ð	&#240;	&eth;	Малая исландская "Эт"

Таблица П1.4 (окончание)

Символ	Десятичный код	Имя	Описание
ñ	&#241;	&ntilde;	Малая N с тильдой
ò	&#242;	&ograve;	Малая O глухая
ó	&#243;	&oacute;	Малая O звонкая
ô	&#244;	&ocirc;	Малая O с циркумфлексом
õ	&#245;	&otilde;	Малая O с тильдой
ö	&#246;	&ouml;	Малая O с умляутом
÷	&#247;	&divide;	Знак деления
ø	&#248;	&oslash;	Малая O зачеркнутая
ù	&#249;	&ugrave;	Малая U глухая
ú	&#250;	&uacute;	Малая U звонкая
û	&#251;	&ucirc;	Малая U с циркумфлексом
ü	&#252;	&uuml;	Малая U с умляутом
ý	&#253;	&yacute;	Малая Y звонкая
þ	&#254;	&thorn;	Малая исландская "Торн"
ÿ	&#255;	&yuml;	Малая Y с умляутом

Коды языков HTML

В табл. П1.5 представлены коды языков, доступные для использования в атрибуте LANG.

***Таблица П1.5.** Коды языков HTML*

Язык	Код
Албанский	sq
Английский	en
Английский (Австралия)	en-au
Английский (Белиз)	en-bz
Английский (Великобритания)	en-gb
Английский (Ирландия)	en-ie

Таблица П1.5 *(продолжение)*

Язык	Код
Английский (Канада)	en-ca
Английский (Новая Зеландия)	en-nz
Английский (США)	en-us
Английский (Тринидад)	en-tt
Английский (Южная Африка)	en-za
Английский (Ямайка)	en-jm
Арабский (Алжир)	ar-dz
Арабский (Бахрейн)	ar-bh
Арабский (Египет)	ar-eg
Арабский (Иордания)	ar-jo
Арабский (Ирак)	ar-iq
Арабский (Йемен)	ar-ye
Арабский (Катар)	ar-qa
Арабский (Кувейт)	ar-kw
Арабский (Ливан)	ar-lb
Арабский (Ливия)	ar-ly
Арабский (Марокко)	ar-ma
Арабский (ОАЕ)	ar-ae
Арабский (Оман)	ar-om
Арабский (Саудовская Аравия)	ar-sa
Арабский (Сирия)	ar-sy
Арабский (Тунис)	ar-tn
Африкаанс	af
Баскский	eu
Белорусский	be
Болгарский	bg
Венгерский	hu
Венда	ve

***Таблица П1.5** (продолжение)*

Язык	Код
Вендский	sb
Вьетнамский	vi
Голландский	nl
Голландский (Бельгия)	nl-be
Греческий	el
Гэльский (Ирландия)	gd-ie
Гэльский (Шотландия)	gd
Датский	da
Зулусский	zu
Иврит	he
Идиш	ji
Индонезийский	in
Исландский	is
Испанский	es
Испанский (Аргентина)	es-ar
Испанский (Боливия)	es-bo
Испанский (Венесуэла)	es-ve
Испанский (Гватемала)	es-gt
Испанский (Гондурас)	es-hn
Испанский (Доминиканская республика)	es-do
Испанский (Колумбия)	es-co
Испанский (Коста-Рика)	es-cr
Испанский (Мексика)	es-mx
Испанский (Никарагуа)	es-ni
Испанский (Панама)	es-pa
Испанский (Парагвай)	es-py
Испанский (Перу)	es-pe
Испанский (Пуэрто-Рико)	es-pr

Таблица П1.5 (продолжение)

Язык	Код
Испанский (Сальвадор)	es-sv
Испанский (Уругвай)	es-uy
Испанский (Чили)	es-cl
Испанский (Эквадор)	es-ec
Итальянский	it
Итальянский (Швейцария)	it-ch
Каталонский	ca
Китайский (Аомынь)	zh-cn
Китайский (Гонконг, КНР)	zh-hk
Китайский (Сингапур)	zh-sg
Китайский (Тайвань)	zh-tw
Корейский	ko
Коса	xh
Латышский	lv
Литовский	lt
Македонский	mk
Малайский	ms
Мальтийский	mt
Немецкий	de
Немецкий (Австрия)	de-at
Немецкий (Лихтенштейн)	de-li
Немецкий (Люксембург)	de-lu
Немецкий (Швейцария)	de-ch
Норвежский	no
Польский	pl
Португальский	pt
Португальский (Бразилия)	pt-br
Реторомансний	rm

Таблица П1.5 *(окончание)*

Язык	Код
Румынский	ro
Румынский (Молдавия)	ro-mo
Русский	ru
Русский (Молдавия)	ru-mo
Саамский (лапландский)	sz
Сербский	sr
Словацкий	sk
Словенский	sl
Суто	sx
Тайский	th
Тсвана	tn
Тсонга	ts
Турецкий	tr
Украинский	uk
Урду	ur
Фарерский	fo
Фарси	fa
Финский	fi
Французский	fr
Французский (Бельгия)	fr-be
Французский (Канада)	fr-ca
Французский (Люксембург)	fr-lu
Французский (Швейцария)	fr-ch
Хинди	hi
Хорватский	hr
Чешский	cs
Шведский	sv
Шведский (Финляндия)	sv-fi
Эстонский	et
Японский	ja

Коды текстовых кодировок HTML

В табл. П1.6 представлены коды текстовых кодировок, доступные для использования в тегах <META>.

Таблица П1.6. *Коды текстовых кодировок HTML*

Язык/кодировка	Код	Кодовая страница
IBM EBCDIC (английский, Великобритания)	x-EBCDIC-UK	20285
IBM EBCDIC (английский, Великобритания)	x-ebcdic-uk-euro	1146
IBM EBCDIC (английский, США и Канада)	ebcdic-cp-us	37
IBM EBCDIC (английский, США и Канада)	x-ebcdic-cp-us-euro	1140
IBM EBCDIC (арабский)	x-EBCDIC-Arabic	20420
IBM EBCDIC (греческий современный)	x-EBCDIC-GreekModern	875
IBM EBCDIC (греческий)	x-EBCDIC-Greek	20423
IBM EBCDIC (датский, норвежский)	x-EBCDIC-DenmarkNorway	20277
IBM EBCDIC (датский, норвежский)	x-ebcdic-denmarknorway-euro	1142
IBM EBCDIC (иврит)	x-EBCDIC-Hebrew	20424
IBM EBCDIC (исландский)	x-EBCDIC-Icelandic	20871
IBM EBCDIC (исландский)	x-ebcdic-icelandic-euro	1149
IBM EBCDIC (испанский)	X-EBCDIC-Spain	20284
IBM EBCDIC (испанский)	x-ebcdic-spain-euro	1145
IBM EBCDIC (итальянский)	x-EBCDIC-Italy	20280
IBM EBCDIC (итальянский)	x-ebcdic-italy-euro	1144
IBM EBCDIC (китайский традиционный)	x-EBCDIC-TraditionalChinese	50937
IBM EBCDIC (китайский упрощенный)	x-EBCDIC-SimplifiedChinese	50935
IBM EBCDIC (корейский — обычный и расширенный)	x-EBCDIC-KoreanAndKoreanExtended	50933

Таблица П1.6 (продолжение)

Язык/кодировка	Код	Кодовая страница
IBM EBCDIC (корейский расширенный)	x-EBCDIC-KoreanExtended	20833
IBM EBCDIC (международный)	x-ebcdic-international-euro	1148
IBM EBCDIC (многоязыковый латинский-2)	CP870	870
IBM EBCDIC (немецкий)	x-EBCDIC-Germany	20273
IBM EBCDIC (немецкий)	x-ebcdic-germany-euro	1141
IBM EBCDIC (русский)	x-EBCDIC-CyrillicRussian	20880
IBM EBCDIC (сербский, болгарский)	x-EBCDIC-CyrillicSerbianBulgarian	21025
IBM EBCDIC (тайский)	x-EBCDIC-Thai	20838
IBM EBCDIC (турецкий латиница-5)	CP1026	1026
IBM EBCDIC (турецкий)	x-EBCDIC-Turkish	20905
IBM EBCDIC (финский, шведский)	x-EBCDIC-FinlandSweden	20278
IBM EBCDIC (финский, шведский)	x-ebcdic-finlandsweden-euro	1143
IBM EBCDIC (французский)	x-ebcdic-france-euro	1147
IBM EBCDIC (японский — иероглифы и катакана)	x-EBCDIC-JapaneseAndKana	50930
IBM EBCDIC (японский — иероглифы и латиница)	x-EBCDIC-JapaneseAndJapaneseLatin	50939
IBM EBCDIC (японский — катакана)	x-EBCDIC-JapaneseKatakana	20290
IBM EBCDIC (японский и английский)	x-EBCDIC-JapaneseAndUSCanada	50931
ISCII ассамес	x-iscii-as	57006
ISCII бенгали	x-iscii-be	57003
ISCII гуджарати	x-iscii-gu	57010
ISCII деванагари	x-iscii-de	57002
ISCII каннада	x-iscii-ka	57008

Таблица П1.6 *(продолжение)*

Язык/кодировка	Код	Кодовая страница
ISCII малаялам	x-iscii-ma	57009
ISCII ория	x-iscii-or	57007
ISCII пенджаби	x-iscii-pa	57011
ISCII тамильский	x-iscii-ta	57004
ISCII телугу	x-iscii-te	57005
Latin 3 (ISO)	iso-8859-3	28593
Latin 9 (ISO)	iso-8859-15	28605
OEM США	IBM437	437
Unicode	unicode	1200
Unicode (Big-Endian)	unicodeFFFE	1201
Unicode (UTF-7)	utf-7	65000
Unicode (UTF-8)	utf-8	65001
US-ASCII	us-ascii	20127
Арабский (ASMO 708)	ASMO-708	708
Арабский (DOS)	DOS-720	720
Арабский (ISO)	iso-8859-6	28596
Арабский (Mac)	x-mac-arabic	10004
Арабский (Windows)	windows-1256	1256
Балтийские языки (DOS)	ibm775	775
Балтийские языки (ISO)	iso-8859-4	28594
Балтийские языки (Windows)	windows-1257	1257
Вьетнамский (Windows)	windows-1258	1258
Греческий (DOS)	ibm737	737
Греческий (ISO)	iso-8859-7	28597
Греческий (Mac)	x-mac-greek	10006
Греческий (Windows)	windows-1253	1253
Греческий современный (DOS)	ibm869	869
Европейские языки	x-Europa	29001
Западноевропейские языки (DOS)	ibm850	850

***Таблица П1.6** (продолжение)*

Язык/кодировка	Код	Кодовая страница
Западноевропейские языки (IA5)	x-IA5	20105
Западноевропейские языки (ISO)	iso-8859-1	28591
Западноевропейские языки (Mac)	macintosh	10000
Западноевропейские языки (Windows)	Windows-1252	1252
Иврит (DOS)	DOS-862	862
Иврит (ISO-Logical)	iso-8859-8-i	38598
Иврит (ISO-Visual)	iso-8859-8	28598
Иврит (Mac)	x-mac-hebrew	10005
Иврит (Windows)	windows-1255	1255
Исландский (DOS)	ibm861	861
Исландский (Mac)	x-mac-icelandic	10079
Кириллица (DOS)	cp866	866
Кириллица (ISO)	iso-8859-5	28595
Кириллица (KOI8-R)	koi8-r	20866
Кириллица (KOI8-U)	koi8-u	21866
Кириллица (Mac)	x-mac-cyrillic	10007
Кириллица (Windows)	windows-1251	1251
Китайский традиционный (Big5)	big5	950
Китайский традиционный (CNS)	x-Chinese-CNS	20000
Китайский традиционный (Eten)	x-Chinese-Eten	20002
Китайский традиционный (Mac)	x-mac-chinesetrad	10002
Китайский упрощенный (EUC)	EUC-CN	51936
Китайский упрощенный (GB2312)	gb2312	936

Таблица П1.6 *(окончание)*

Язык/кодировка	Код	Кодовая страница
Китайский упрощенный (HZ)	hz-gb-2312	52936
Китайский упрощенный (Mac)	x-mac-chinesesimp	10008
Корейский	ks_c_5601-1987	949
Корейский (EUC)	euc-kr	51949
Корейский (ISO)	iso-2022-kr	50225
Корейский (Johab)	Johab	1361
Корейский (Mac)	x-mac-korean	10003
Немецкий (IA5)	x-IA5-German	20106
Норвежский (IA5)	x-IA5-Norwegian	20108
Тайский (Windows)	windows-874	874
Турецкий (DOS)	ibm857	857
Турецкий (ISO)	iso-8859-9	28599
Турецкий (Mac)	x-mac-turkish	10081
Турецкий (Windows)	windows-1254	1254
Центральноевропейские языки (DOS)	ibm852	852
Центральноевропейские языки (ISO)	iso-8859-2	28592
Центральноевропейские языки (Mac)	x-mac-ce	10029
Центральноевропейские языки (Windows)	windows-1250	1250
Шведский (IA5)	x-IA5-Swedish	20107
Японский (EUC)	euc-jp	51932
Японский (JIS)	iso-2022-jp	50220
Японский (JIS-Allow 1 byte Kana — SO/SI)	iso-2022-jp	50222
Японский (JIS-Allow 1 byte Kana)	csISO2022JP	50221
Японский (Mac)	x-mac-japanese	10001
Японский (Shift-JIS)	shift_jis	932

ПРИЛОЖЕНИЕ 2

CSS

В этом приложении описываются все атрибуты каскадных таблиц стилей CSS, поддерживаемые двумя популярнейшими программами Web-обозревателей.

Атрибуты

Суть таблиц стилей заключается в совокупности атрибутов, которые могут применяться к выбранным элементам. Поясняется, как должен работать каждый из атрибутов в соответствии со спецификацией. Но на практике большинство из них будут по-разному вести себя на различных Web-обозревателях, а многие вообще не будут поддерживаться. В связи с этим описывается совместимость каждого атрибута с наиболее распространенными Web-обозревателями.

ACCELERATOR

Позволяет указать, содержит ли текст элемента страницы клавишу-ускоритель. Клавиша-ускоритель — это особая клавиша на клавиатуре, при нажатии которой вместе с клавишей <Alt> происходит переход к данному элементу страницы.

```
ACCELERATOR: true|false;
```

Доступны два значения: `true` указывает, что текст содержит клавишу-ускоритель, а `false` — что не содержит. Значения по умолчанию нет.

Пример использования атрибута ACCELERATOR.

```
<LABEL FOR="txtName"><U STYLE="ACCELERATOR: true">И</U>мя</LABEL>
<INPUT TYPE="text" ID="txtName" ACCESSKEY="B" VALUE="Имя пользователя">
```

В этом случае символ "И" в слове "Имя" будет подчеркнут. Если в региональных настройках операционной системы Windows 2000 была выбрана

опция **Скрыть индикаторы клавиш-ускорителей до нажатия Alt**, этот символ не будет подчеркнут, пока пользователь не нажмет клавишу <Alt> на клавиатуре.

Поддерживается IE начиная с 5.0.

background

Комбинированный атрибут, заменяющий атрибуты background-attachment, background-color, background-image, background-position и background-repeat. Задает все свойства фона элемента страницы в один прием. Значения этих свойств могут располагаться в любом порядке.

```
background: [{background-color}] [{background-image}]
↳[{background-repeat}] [{background-attachment}] [{background-position}];
```

Значение по умолчанию transparent none repeat scroll 0% 0%.

Поддерживается IE начиная с 3.02; задание значений background-position и background-repeat поддерживается начиная с 4.0.

background-attachment

Позволяет "зафиксировать" фоновый рисунок, чтобы он не прокручивался вместе с содержимым Web-страницы. Применяется только для тега <BODY>.

```
background-attachment: scroll|fixed;
```

Доступны два значения: scroll (значение по умолчанию) заставляет фоновый рисунок прокручиваться вместе с содержимым страницы, а fixed "фиксирует" его.

Поддерживается IE начиная с 4.0, в составе атрибута background — с 3.02.

background-color

Задает фоновый цвет Web-страницы или ее элемента.

```
background-color: {Цвет}|transparent;
```

Предопределенное значение transparent задает "прозрачный" фон. Оно же является значением по умолчанию.

Поддерживается IE начиная с 4.0, в составе атрибута background — с 3.02. Поддерживается N начиная с 4.0.

background-image

Задает фоновый рисунок Web-страницы или ее элемента.

```
background-image: url({Интернет-адрес файла рисунка})|none;
```

Предопределенное значение none отключает фоновый рисунок. Оно же является значением по умолчанию.

Поддерживается IE начиная с 4.0, в составе атрибута background — с 3.02. Поддерживается N начиная с 4.0.

background-position

Комбинированный атрибут, заменяющий атрибуты background-position-x и background-position-y. Задает местонахождение фонового рисунка.

```
background-position: [{background-position-x}] [{background-position-y}];
```

Значение по умолчанию 0% 0%. Если задана только одна координата, то она считается горизонтальной, а для вертикальной принимается значение 50%.

Поддерживается IE начиная с 4.0.

background-position-x

Задает горизонтальную координату фонового рисунка.

```
background-position-x: {X}|{X}%|left|center|right;
```

Координата может быть задана целым числом (абсолютная координата), процентом от соответствующего размера фонового рисунка (относительная координата) или предопределенным значением. Доступны три предопределенных значения: left, center и right, задающие выравнивание фонового рисунка на странице по левому краю, по центру и по правому краю соответственно.

Значение по умолчанию 0%.

Поддерживается IE начиная с 4.0.

background-position-y

Задает вертикальную координату фонового рисунка.

```
background-position-y: {Y}|{Y}%|top|center|bottom;
```

Координата может быть задана целым числом (абсолютная координата), процентом от соответствующего размера фонового рисунка (относительная координата) или предопределенным значением. Доступны три предопределенных значения: top, center и bottom, задающие выравнивание по верху, по центру и по низу страницы соответственно.

Значение по умолчанию 0%.

Поддерживается IE начиная с 4.0.

background-repeat

Устанавливает порядок повторения фонового рисунка на Web-странице или ее элементе. (Фоновый рисунок повторяется, чтобы занять все свободное пространство, если он слишком мал, чтобы покрыть его целиком без повторения.)

```
background-repeat: repeat|no-repeat|repeat-x|repeat-y;
```

Доступно четыре значения:

- `repeat` (значение по умолчанию) — заставляет фоновый рисунок повторяться по горизонтали и вертикали;
- `no-repeat` — запрещает фоновому рисунку повторяться;
- `repeat-x` — заставляет фоновый рисунок повторяться только по горизонтали;
- `repeat-y` — заставляет фоновый рисунок повторяться только по вертикали.

Поддерживается IE начиная с 4.0.

behavior

Задает поведение DHTML для элемента страницы.

```
behavior: url({Интернет-адрес файла поведения})|
↪url(#{Имя элемента ActiveX, реализующего это поведение})|
↪url(#default#{Имя встроенного поведения});
```

Поддерживается IE начиная с 5.0.

border

Задает все свойства границ элемента страницы в один прием. Заменяет атрибуты `border-color`, `border-style` и `border-width`. Значения этих атрибутов могут располагаться в любом порядке.

```
border: [{border-color}] [{border-style}] [{border-width}];
```

Значение по умолчанию `medium none`.

Поддерживается IE начиная с 4.0.

border-bottom

Задает все свойства нижней границы элемента страницы в один прием. Заменяет атрибуты `border-bottom-color`, `border-bottom-style` и `border-bottom-width`. Значения этих атрибутов могут располагаться в любом порядке.

```
border-bottom: [{border-bottom-color}] [{border-bottom-style}]
↳[{border-bottom-width}];
```

Значение по умолчанию `medium none`.

Поддерживается IE начиная с 4.0.

border-bottom-color

Задает цвет нижней границы элемента страницы.

```
border-bottom-color: {Цвет};
```

Поддерживается IE начиная с 4.0.

border-bottom-style

Задает тип нижней границы элемента страницы.

```
border-bottom-style: none|dotted|dashed|solid|double|groove|ridge|inset|
↳outset;
```

Доступны девять значений:

- `none` (значение по умолчанию) — запрещает рисование границы;
- `dotted` — рисует точечную линию;
- `dashed` — рисует штриховую линию;
- `solid` — рисует сплошную линию;
- `double` — рисует двойную сплошную линию;
- `groove` — рисует трехмерную вдавленную границу;
- `ridge` — рисует трехмерную выпуклую границу;
- `inset` — рисует трехмерную "ступеньку вверх";
- `outset` — рисует трехмерную "ступеньку вниз".

Поддерживается IE начиная с 4.0.

border-bottom-width

Задает толщину нижней границы элемента страницы.

```
border-bottom-width: medium|thin|thick|{Толщина};
```

Толщина может быть задана числовым значением. Также доступны три предопределенных значения: `thin`, `medium` и `thick`, рисующие тонкую, среднюю и толстую линию соответственно. Значение по умолчанию `medium`.

Поддерживается IE и N начиная с 4.0.

border-collapse

Задает, будут ли границы ячеек и общая граница таблицы сливаться в одну или нет. Применяется только для тега <TABLE>.

```
border-collapse: separate|collapse;
```

Доступны два значения: separate (значение по умолчанию) разделяет границу таблицы и границы ее ячеек, а collapse по возможности объединяет их.

Поддерживается IE начиная с 5.0.

border-color (IE)

Задает цвет всех границ элемента страницы. Заменяет атрибуты border-top-color, border-right-color, border-bottom-color и border-left-color.

```
border-color: {border-top-color} [{border-right-color}]
↳[{border-bottom-color}] [{border-left-color}];
```

Может быть задано от одного до четырех значений. Если задано одно значение, оно применяется ко всем четырем границам. Если задано два значения, первое относится к верхней и нижней границам, а второе — к левой и правой. Если задано три значения, то первое применяется к верхней границе, второе — к левой и правой, третье — к нижней.

Поддерживается IE начиная с 4.0.

border-color (N)

Задает цвет границ элемента страницы.

```
border-color: none|{Цвет};
```

Может быть задано либо цветовое значение, либо none, обозначающее отсутствие границ.

Поддерживается N начиная с 4.0.

border-left

Задает все свойства левой границы элемента страницы в один прием. Заменяет атрибуты border-left-color, border-left-style и border-left-width. Значения этих атрибутов могут располагаться в любом порядке.

```
border-left: [{border-left-color}] [{border-left-style}]
↳[{border-left-width}];
```

Значение по умолчанию medium none.

Поддерживается IE начиная с 4.0.

border-left-color

Задает цвет левой границы элемента страницы.

```
border-left-color: {Цвет};
```

Поддерживается IE начиная с 4.0.

border-left-style

Задает тип левой границы элемента страницы.

```
border-left-style: none|dotted|dashed|solid|double|groove|ridge|inset|
↳outset;
```

Доступны девять значений:

- `none` (значение по умолчанию) — запрещает рисование границы;
- `dotted` — рисует точечную линию;
- `dashed` — рисует штриховую линию;
- `solid` — рисует сплошную линию;
- `double` — рисует двойную сплошную линию;
- `groove` — рисует трехмерную вдавленную границу;
- `ridge` — рисует трехмерную выпуклую границу;
- `inset` — рисует трехмерную "ступеньку вверх";
- `outset` — рисует трехмерную "ступеньку вниз".

Поддерживается IE начиная с 4.0.

border-left-width

Задает толщину левой границы элемента страницы.

```
border-left-width: medium|thin|thick|{Толщина};
```

Толщина может быть задана числовым значением. Также доступны три предопределенных значения: `thin`, `medium` и `thick`, рисующие тонкую, среднюю и толстую линию соответственно. Значение по умолчанию `medium`.

Поддерживается IE и N начиная с 4.0.

border-right

Задает все свойства правой границы элемента страницы в один прием. Заменяет атрибуты `border-right-color`, `border-right-style` и `border-right-width`. Значения этих атрибутов могут располагаться в любом порядке.

```
border-right: [{border-right-color}] [{border-right-style}]
↳[{border-right-width}];
```

Значение по умолчанию `medium none`.

Поддерживается IE начиная с 4.0.

border-right-color

Задает цвет правой границы элемента страницы.

```
border-right-color: {Цвет};
```

Поддерживается IE начиная с 4.0.

border-right-style

Задает тип правой границы элемента страницы.

```
border-right-style: none|dotted|dashed|solid|double|groove|ridge|inset|
↳outset;
```

Доступны девять значений:

- `none` (значение по умолчанию) — запрещает рисование границы;
- `dotted` — рисует точечную линию;
- `dashed` — рисует штриховую линию;
- `solid` — рисует сплошную линию;
- `double` — рисует двойную сплошную линию;
- `groove` — рисует трехмерную вдавленную границу;
- `ridge` — рисует трехмерную выпуклую границу;
- `inset` — рисует трехмерную "ступеньку вверх";
- `outset` — рисует трехмерную "ступеньку вниз".

Поддерживается IE начиная с 4.0.

border-right-width

Задает толщину правой границы элемента страницы.

```
border-right-width: medium|thin|thick|{Толщина};
```

Толщина может быть задана числовым значением. Также доступны три предопределенных значения: `thin`, `medium` и `thick`, рисующие тонкую, среднюю и толстую линию соответственно. Значение по умолчанию `medium`.

Поддерживается IE и N начиная с 4.0.

border-style

Задает тип сразу всех границ элемента страницы в один прием. Заменяет атрибуты `border-top-style`, `border-right-style`, `border-bottom-style` и `border-left-style`.

```
border-style: none|dotted|dashed|solid|double|groove|ridge|inset|outset;
```

Доступны девять значений:

- `none` (значение по умолчанию) — запрещает рисование границы;
- `dotted` — рисует точечную линию (N не поддерживается);
- `dashed` — рисует штриховую линию (N не поддерживается);
- `solid` — рисует сплошную линию;
- `double` — рисует двойную сплошную линию;
- `groove` — рисует трехмерную вдавленную границу;
- `ridge` — рисует трехмерную выпуклую границу;
- `inset` — рисует трехмерную "ступеньку вверх";
- `outset` — рисует трехмерную "ступеньку вниз".

Поддерживается IE и N начиная с 4.0.

border-top

Задает все свойства верхней границы элемента страницы в один прием. Заменяет атрибуты `border-top-color`, `border-top-style` и `border-top-width`. Значения этих атрибутов могут располагаться в любом порядке.

```
border-top: [{border-top-color}] [{border-top-style}]
↳[{border-top-width}];
```

Значение по умолчанию `medium none`.

Поддерживается IE начиная с 4.0.

border-top-color

Задает цвет верхней границы элемента страницы.

```
border-top-color: {Цвет};
```

Поддерживается IE начиная с 4.0.

border-top-style

Задает тип верхней границы элемента страницы.

```
border-top-style: none|dotted|dashed|solid|double|groove|ridge|inset|
↳outset;
```

Доступны девять значений:

- `none` (значение по умолчанию) — запрещает рисование границы;
- `dotted` — рисует точечную линию;
- `dashed` — рисует штриховую линию;
- `solid` — рисует сплошную линию;
- `double` — рисует двойную сплошную линию;
- `groove` — рисует трехмерную вдавленную границу;
- `ridge` — рисует трехмерную выпуклую границу;
- `inset` — рисует трехмерную "ступеньку вверх";
- `outset` — рисует трехмерную "ступеньку вниз".

Поддерживается IE начиная с 4.0.

border-top-width

Задает толщину верхней границы элемента страницы.

```
border-top-width: medium|thin|thick|{Толщина};
```

Толщина может быть задана числовым значением. Также доступны три предопределенных значения: `thin`, `medium` и `thick`, рисующие тонкую, среднюю и толстую линию соответственно. Значение по умолчанию `medium`.

Поддерживается IE и N начиная с 4.0.

border-width

Задает толщину всех границ элемента страницы. Заменяет атрибуты `border-top-width`, `border-right-width`, `border-bottom-width` и `border-left-width`.

```
border-width: {border-top-width} [{border-right-width}]
↳[{border-bottom-width}] [{border-left-width}];
```

Может быть задано от одного до четырех значений. Если задано одно значение, оно применяется ко всем четырем границам. Если задано два значения, первое относится к верхней и нижней границам, а второе — к левой и правой. Если задано три значения, то первое применяется к верхней границе, второе — к левой и правой, третье — к нижней.

Толщина может быть задана числовым значением. Также доступны три предопределенных значения: `thin`, `medium` и `thick`, рисующие тонкую, среднюю и толстую линию соответственно. Значение по умолчанию `medium`.

Поддерживается IE и N начиная с 4.0.

bottom

Задает вертикальную позицию нижней границы свободно позиционированного элемента относительно родителя.

```
bottom: auto|{Y}|{Y}%;
```

Координата может быть задана как абсолютной величиной, так и процентом от высоты родителя. Предопределенное значение `auto` заставляет Web-обозреватель позиционировать элемент самостоятельно. Значение по умолчанию `auto`.

Поддерживается IE начиная с 4.0.

clear

Задает поведение текста при "обтекании" им некоторых элементов страницы, таких как изображения. Атрибут задается для текста, а не для элемента страницы, который он будет "обтекать".

```
clear: none|left|right|both;
```

Доступны четыре значения:

- `none` (значение по умолчанию) — разрешает тексту "обтекать" элемент страницы.
- `left` — запрещает тексту "обтекать" элемент страницы с левой стороны;
- `right` — с правой стороны;
- `all` — с обеих сторон.

Поддерживается IE и N начиная с 4.0.

clip

Задает прямоугольный фрагмент элемента страницы, который будет видим.

```
clip: auto|rect({Верхняя граница} {Правая граница} {Нижняя граница}
↳{Левая граница})
```

Предопределенное значение `auto` задает видимость всего элемента страницы, это же значение по умолчанию. Для того чтобы ограничить видимую часть элемента страницы прямоугольным фрагментом, задаются четыре координаты границ этого прямоугольника, разделенные пробелами.

Поддерживается IE и N начиная с 4.0.

color

Задает цвет текста.

```
color: {Цвет};
```

Поддерживается IE начиная с 3.02 для текстовых элементов страницы и начиная с 4.0 для нетекстовых. Поддерживается N начиная с 4.0.

cursor

Задает форму курсора мыши, которую он принимает при наведении на элемент страницы.

```
cursor: auto|crosshair|default|hand|move|*-resize|text|wait|help;
```

Доступны следующие значения:

- `auto` (значение по умолчанию) — заставляет Web-обозреватель самому определять нужную форму курсора мыши;
- `crosshair` — крестообразный курсор;
- `default` — курсор по умолчанию, обычно стрелка;
- `hand` — "указующий перст";
- `move` — стрелка, указывающая "на все четыре стороны";
- `*-resize` (где вместо `*` может стоять `n`, `ne`, `nw`, `s`, `se`, `sw`, `e` или `w`) — "стрелка компаса", указывающая направление;
- `text` — текстовый курсор;
- `wait` — "песочные часы", курсор ожидания;
- `help` — стрелка с вопросительным знаком.

Поддерживается IE начиная с 4.0.

direction

Задает порядок чтения текста: слева направо или справа налево.

```
direction: ltr|rtl|inherit;
```

Доступны три значения:

- `ltr` (значение по умолчанию) — задает порядок чтения слева направо;
- `rtl` — справа налево;
- `inherit` — заставляет наследовать порядок чтения у родителя.

Для документов, составленных на европейских языках, порядок чтения всегда слева направо (`ltr`).

Поддерживается IE начиная с 5.0.

display

Задает вид элемента страницы.

```
display: inline|block|none|inline-block|list-item|
↳table-header-group|table-footer-group;
```

Доступны семь значений:

- `inline` (значение по умолчанию) — заставляет элемент страницы вести себя как встроенный. При этом он располагается внутри текста, его позиция и размеры рассчитываются согласно позиции и размерам текста. Пример встроенного элемента — `<I>`;
- `block` — делает элемент страницы блочным. При этом его можно свободно позиционировать. Пример блочного элемента — `<DIV>`;
- `none` — делает элемент страницы невидимым. При этом страница отображается так, будто этого элемента вообще не существует;
- `inline-block` — аналогично `inline`, но содержимое элемента страницы ведет себя как блочный элемент (N не поддерживается);
- `list-item` — аналогично `block`, но при этом элемент страницы считается позицией списка;
- `table-header-group` — заставляет элемент страницы отображаться после верхнего заголовка таблицы и перед всеми строками данных (аналогично тегу `<THEAD>`) (N не поддерживается);
- `table-footer-group` — заставляет элемент страницы отображаться перед основанием таблицы и после всех строк данных (аналогично тегу `<TFOOT>`) (N не поддерживается).

Поддерживается IE и N начиная с 4.0.

filter

Задает набор визуальных фильтров и преобразований, применяемых к элементу страницы.

```
filter: {Список фильтров и преобразований с параметрами, разделенными
↳пробелами};
```

Поддерживается IE начиная с 4.0.

float

Задает принудительное выравнивание элемента страницы относительно родителя.

```
float: none|left|right;
```

Доступны три значения:

- `none` (значение по умолчанию) — заставляет элемент страницы появляться там, где он задан;
- `left` — принудительно выравнивает элемент страницы по левому краю;
- `right` — по правому краю.

Поддерживается IE начиная с 4.0 для кнопок и внедренных объектов и начиная с 5.0 — для остальных элементов страницы. Поддерживается N начиная с 4.0.

font

Задает параметры шрифта элемента страницы. Заменяет атрибуты `font-family`, `font-height`, `font-size`, `font-style`, `font-variant` и `font-weight`. Значения этих атрибутов могут располагаться в любом порядке.

```
font: {font-family} [{font-height}] [{font-size}] [{font-style}]
↳[{font-variant}] [{font-weight}];
```

Значение по умолчанию — `normal normal normal medium normal "Times New Roman"`.

Альтернативный формат:

```
font: caption|icon|menu|message-box|small-caption|status-bar;
```

В этом случае доступны шесть предопределенных значений, задающие один из стандартных шрифтов, используемых в элементах интерфейса Windows:

- `caption` — шрифт заголовков кнопок, текстовых меток и т. п.;
- `icon` — шрифт подписей под пиктограммами;
- `menu` — шрифт пунктов меню;
- `message-box` — шрифт содержимого стандартных окон-предупреждений;
- `small-caption` — мелкий шрифт заголовков;
- `status-bar` — шрифт содержимого строки состояния.

Поддерживается IE начиная с 4.0.

font-family

Задает имя шрифта, используемого в элементе страницы.

```
font-family: {Имя шрифта}|serif|san-serif|cursive|fantasy|monospace;
```

В качестве значения этого атрибута задается либо непосредственно имя нужного шрифта, либо одно из пяти предопределенных значений, задающих имя шрифтового семейства. Можно задавать одновременно несколько

шрифтов, разделив их имена запятыми; в этом случае Web-обозреватель сможет выбрать из них тот, который установлен на компьютере клиента. Если имя шрифта содержит пробелы, его следует взять в одинарные кавычки.

Поддерживается IE начиная с 3.02 для текстовых элементов и начиная с 4.0 для нетекстовых. Поддерживается N начиная с 4.0.

font-size

Задает размер шрифта, используемого в элементе страницы.

```
font-size: xx-small|x-small|small|medium|large|x-large|xx-large|
↳larger|smaller|{Абсолютный размер}|{Относительный размер}%
```

Возможно задание либо абсолютного размера шрифта в одной из поддерживаемых CSS единиц измерения, либо как процент от размера шрифта родителя. Также доступны девять предопределенных значений. Значения `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large` и `xx-large` задают один из семи размеров шрифтов, поддерживаемых HTML. Значения `smaller` и `larger` задают размер шрифта — по отношению к шрифту родителя на размер меньше или больше соответственно.

Поддерживается IE начиная с 3.02 для текстовых элементов и начиная с 4.0 для нетекстовых. Поддерживается N начиная с 4.0.

font-style

Задает вид шрифта, используемого в элементе страницы.

```
font-style: normal|italic|oblique;
```

Доступны три значения:

- `normal` (значение по умолчанию) — задает обычный вид шрифта;
- `italic` — задает курсивное начертание;
- `oblique` — задает наклонное начертание (IE отображает его как курсив, а N вообще не поддерживает).

Поддерживается IE начиная с 3.02 для текстовых элементов и начиная с 4.0 для нетекстовых. Поддерживается N начиная с 4.0.

font-variant

Задает вид малых букв шрифта, используемого в элементе страницы.

```
font-variant: normal|small-caps;
```

Доступны два значения:

- `normal` (значение по умолчанию) — задает обычный вид малых букв шрифта;
- `small-caps` — делает их такими же, как большие буквы, только меньшего размера (капитель).

Поддерживается IE начиная с 4.0.

font-weight

Задает "жирность" шрифта, используемого в элементе страницы.

```
font-weight: normal|bold|bolder|lighter|100|200|300|400|500|600|700|800|
↳900;
```

"Жирность" может быть задана тремя способами. Во-первых, предопределенными значениями `normal` и `bold`, задающими обычное и жирное начертание соответственно. Во-вторых, относительными значениями `bolder` и `lighter`, делающими шрифт элемента страницы жирнее и светлее шрифта родителя. И, в-третьих, одним из девяти значений от `100` до `900`; здесь нормальному начертанию соответствует значение `400`, а жирному — `700`. Значение по умолчанию `normal`.

Поддерживается IE начиная с 3.02 для текстовых элементов и начиная с 4.0 для нетекстовых. При этом IE 4.0 и более старые версии распознавали только значения `normal` и `bold` этого атрибута. Поддерживается N начиная с 4.0.

height

Задает высоту свободно позиционированного элемента.

```
height: auto|{Y}|{Y}%;
```

Высота может быть задана как абсолютной величиной, так и процентом от высоты родителя. Предопределенное значение `auto` заставляет Web-обозреватель устанавливать высоту элемента самостоятельно. Значение по умолчанию `auto`.

Поддерживается IE и N начиная с 4.0.

ime-mode

Задает состояние IME (Input Method Editor — редактор способа ввода), с помощью которого вводятся иероглифические тексты на китайском, корейском и японском языках. Этот атрибут применяется только для полей ввода.

```
ime-mode: auto|active|inactive|disabled;
```

Доступны четыре значения:

- `auto` (значение по умолчанию) — передает управление IME Web-обозревателю;
- `active` — активизирует IME. Пользователь может его деактивизировать;
- `inactive` — деактивизирует IME. Пользователь может его активизировать;
- `disabled` — отключает IME.

Поддерживается IE начиная с 5.0.

layout-flow

Задает направление написания текста: по горизонтали или по вертикали.

```
layout-flow: horizontal|vertical-ideographic;
```

Доступны два значения:

- `horizontal` (значение по умолчанию) — задает горизонтальное направление написания текста;
- `vertical-ideographic` — вертикальное направление написания текста.

Поддерживается IE начиная с 5.5. В настоящее время признан устаревшим; вместо него рекомендуется использовать атрибут `writing-mode`.

layout-grid

Задает параметры разметки элемента страницы, используемой для вывода иероглифических текстов на китайском, корейском и японском языках. Заменяет атрибуты `layout-grid-char`, `layout-grid-line`, `layout-grid-mode` и `layout-grid-type`. Значения этих атрибутов могут располагаться в любом порядке.

```
layout-grid: [{layout-grid-char}] [{layout-grid-line}]
↳[{layout-grid-mode}] [{layout-grid-type}];
```

Значение по умолчанию — `both loose none none`.

Поддерживается IE начиная с 5.0.

layout-grid-char

Задает шаг горизонтальной разметки элемента страницы, используемой для вывода иероглифических текстов на китайском, корейском и японском языках.

```
layout-grid-char: none|auto|{Y}|{Y}%;
```

Шаг разметки может быть задан как абсолютной величиной, так и процентом от шага разметки родителя. Предопределенное значение `auto` заставляет Web-обозреватель устанавливать шаг разметки по максимальному символу текста. Другое предопределенное значение `none` вообще отключает разметку. Значение по умолчанию `none`.

Поддерживается IE начиная с 5.0.

layout-grid-line

Задает шаг вертикальной разметки элемента страницы, используемой для вывода иероглифических текстов на китайском, корейском и японском языках.

```
layout-grid-line: none|auto|{Y}|{Y}%;
```

Шаг разметки может быть задан как абсолютной величиной, так и процентом от шага разметки родителя. Предопределенное значение `auto` заставляет Web-обозреватель устанавливать шаг разметки по максимальному символу текста. Другое предопределенное значение `none` вообще отключает разметку. Значение по умолчанию `none`.

Поддерживается IE начиная с 5.0.

layout-grid-mode

Задает тип разметки элемента страницы, используемой для вывода иероглифических текстов на китайском, корейском и японском языках.

```
layout-grid-mode: both|none|char|line;
```

Доступны четыре значения:

- `both` (значение по умолчанию) — задает использование и горизонтальной, и вертикальной разметок;
- `none` — отключает разметку;
- `char` — задает использование горизонтальной разметки;
- `line` — задает использование вертикальной разметки.

Поддерживается IE начиная с 5.0.

layout-grid-type

Задает режим форматирования иероглифических текстов на китайском, корейском и японском языках с использованием разметки элемента страницы.

```
layout-grid-type: loose|strict|fixed;
```

Доступны три значения:

- `loose` (значение по умолчанию) — задает "гибкое" форматирование, используемое для корейских и японских текстов;
- `strict` — задает более "строгое" форматирование, используемое для китайских, корейских и японских текстов;
- `fixed` — задает самое "строгое" форматирование, когда символы жестко привязываются к разметке.

Поддерживается IE начиная с 5.0.

left

Задает горизонтальную позицию левой границы свободно позиционированного элемента относительно родителя.

```
left: auto|{X}|{X}%;
```

Координата может быть задана как абсолютной величиной, так и процентом от ширины родителя. Предопределенное значение `auto` заставляет Web-обозреватель позиционировать элемент самостоятельно. Значение по умолчанию `auto`.

Поддерживается IE и N начиная с 4.0.

letter-spacing

Задает дополнительное пространство, добавляемое между символами текста.

```
letter-spacing: normal|{Величина};
```

Значение этого атрибута может быть задано либо абсолютной величиной в одной из поддерживаемых CSS единиц измерения, либо предопределенным значением `normal`, задающим стандартную величину расстояния между символами. Значение по умолчанию `normal`.

Поддерживается IE начиная с 4.0.

line-break

Задает правила разрыва строк для текста на японском языке.

```
line-break: normal|strict;
```

Доступны два значения: `normal` (значение по умолчанию), задающее обычные правила разрыва японского текста, и `strict`, задающее строгие правила.

Поддерживается IE начиная с 5.0.

line-height

Задает вертикальное расстояние между строками текста.

```
line-height: normal|{Y}|{Y}%;
```

Высота может быть задана как абсолютной величиной, так и процентом от высоты родителя. Предопределенное значение `normal` задает стандартное расстояние. Значение по умолчанию `normal`.

Поддерживается IE и N начиная с 4.0.

list-style

Задает параметры маркера или номера позиции списка. Заменяет атрибуты `list-style-image`, `list-style-position` и `list-style-type`. Значения этих атрибутов могут располагаться в любом порядке.

```
list-style: [{list-style-image}] [{list-style-position}]
↳[{list-style-type}];
```

Значение по умолчанию `disc outside none`.

Поддерживается IE начиная с 4.0.

list-style-image

Задает графическое изображение, отображаемое в качестве маркера позиции списка. Имеет приоритет над атрибутом `list-style-type`.

```
list-style-image: none|url({Интернет-адрес файла изображения});
```

Если задано предопределенное значение `none`, то стиль маркера берется из установок атрибута `list-style-type`, если он задан, или отображается маркер по умолчанию. Если задан интернет-адрес файла изображения, то оно отображается в качестве маркера, перекрывая установки атрибута `list-style-type`. Значение по умолчанию `none`.

Поддерживается IE начиная с 4.0.

list-style-position

Задает местонахождение маркера позиции списка: в тексте позиции или вне его.

```
list-style-position: outside|inside;
```

Доступны два значения. Значение `outside` (по умолчанию) задает отображение маркера позиции списка вне текста позиции. Значение же `inside` за-

ставляет Web-обозреватель отобразить маркер позиции в ее тексте в качестве первого символа.

Поддерживается IE начиная с 4.0.

list-style-type

Задает тип маркера или номера позиции списка.

```
list-style-type: disc|circle|square|decimal|lower-roman|upper-roman|
↳lower-alpha|upper-alpha|none;
```

Доступны девять значений:

- `disc` (значение по умолчанию) — отображает сплошной кружок;
- `circle` — окружность;
- `square` — сплошной квадрат;
- `decimal` — нумерует позиции арабскими цифрами;
- `lower-roman` — малыми римскими;
- `upper-roman` — большими римскими;
- `lower-alpha` — помечает позиции малыми латинскими буквами;
- `upper-alpha` — большими латинскими;
- `none` — вообще убирает маркер или нумерацию.

Поддерживается IE и N начиная с 4.0.

margin

Задает ширины полей между элементом страницы и его соседями. Заменяет атрибуты `margin-top`, `margin-right`, `margin-bottom` и `margin-left`.

```
margin: {margin-top} [{margin-right}] [{margin-bottom}] [{margin-left}];
```

Может быть задано от одного до четырех значений. Если задано одно значение, оно применяется ко всем четырем полям. Если задано два значения, первое относится к верхнему и нижнему полю, а второе — к левому и правому. Если задано три значения, то первое применяется к верхнему полю, второе — к левому и правому, третье — к нижнему.

Поддерживается IE начиная с 3.02 для текстовых элементов и начиная с 4.0 для нетекстовых. Поддерживается N начиная с 4.0.

margin-bottom

Задает нижнее поле между элементом страницы и его соседями снизу.

```
margin-bottom: auto|{Y}|{Y}%;
```

Координата может быть задана как абсолютной величиной, так и процентом от высоты родителя. Предопределенное значение `auto` заставляет Web-обозреватель устанавливать поле самостоятельно. При этом если задан атрибут `margin-top`, значение нижнего поля устанавливается равным значению верхнего поля. Значение по умолчанию `auto`.

Поддерживается IE начиная с 3.02 для текстовых элементов и начиная с 4.0 для нетекстовых. Поддерживается N начиная с 4.0.

margin-left

Задает левое поле между элементом страницы и его соседями слева.

```
margin-left: auto|{X}|{X}%;
```

Координата может быть задана как абсолютной величиной, так и процентом от ширины родителя. Предопределенное значение `auto` заставляет Web-обозреватель устанавливать поле самостоятельно. При этом если задан атрибут `margin-right`, значение левого поля устанавливается равным значению правого поля. Значение по умолчанию `auto`.

Поддерживается IE начиная с 3.02 для текстовых элементов и начиная с 4.0 для нетекстовых. Поддерживается N начиная с 4.0.

margin-right

Задает правое поле между элементом страницы и его соседями справа.

```
margin-right: auto|{X}|{X}%;
```

Координата может быть задана как абсолютной величиной, так и процентом от ширины родителя. Предопределенное значение `auto` заставляет Web-обозреватель устанавливать поле самостоятельно. При этом если задан атрибут `margin-left`, значение правого поля устанавливается равным значению левого поля. Значение по умолчанию `auto`.

Поддерживается IE начиная с 3.02 для текстовых элементов и начиная с 4.0 для нетекстовых. Поддерживается N начиная с 4.0.

margin-top

Задает верхнее поле между элементом страницы и его соседями сверху.

```
margin-top: auto|{Y}|{Y}%;
```

Координата может быть задана как абсолютной величиной, так и процентом от высоты родителя. Предопределенное значение `auto` заставляет Web-обозреватель устанавливать поле самостоятельно. При этом если задан атрибут `margin-bottom`, значение верхнего поля устанавливается равным значению нижнего поля. Значение по умолчанию `auto`.

Поддерживается IE начиная с 3.02 для текстовых элементов и начиная с 4.0 для нетекстовых. Поддерживается N начиная с 4.0.

overflow

Задает поведение элемента страницы, если содержимое в нем не помещается.

```
overflow: visible|scroll|hidden|auto;
```

Доступно четыре значения:

- `visible` — заставляет элемент страницы расшириться так, чтобы все его содержимое было видно (значение по умолчанию);
- `scroll` — заставляет Web-обозреватель отобразить в элементе страницы полосы прокрутки, пользуясь которыми можно прокручивать его содержимое;
- `hidden` — скрывает все то, что выходит за пределы элемента страницы;
- `auto` — аналогично `scroll` за тем исключением, что полосы прокрутки отображаются только тогда, когда они реально необходимы (значение по умолчанию для <TEXTAREA>).

Поддерживается IE начиная с 4.0.

overflow-x

Задает поведение элемента страницы, если его ширина меньше ширины содержимого.

```
overflow-x: visible|scroll|hidden|auto;
```

Доступно четыре значения:

- `visible` — заставляет элемент страницы расшириться по горизонтали так, чтобы все его содержимое было видно (значение по умолчанию);
- `scroll` — заставляет Web-обозреватель отобразить в элементе страницы полосу прокрутки, пользуясь которой, можно прокручивать его содержимое;
- `hidden` — скрывает все то, что выходит за пределы элемента страницы (значение по умолчанию для <TEXTAREA>);
- `auto` — аналогично `scroll` за тем исключением, что полоса прокрутки отображается только тогда, когда она реально необходима.

Поддерживается IE начиная с 4.0.

overflow-y

Задает поведение элемента страницы, если его высота меньше высоты его содержимого.

```
overflow-y: visible|scroll|hidden|auto;
```

Доступно четыре значения:

- `visible` — заставляет элемент страницы расшириться по вертикали так, чтобы все его содержимое было видно (значение по умолчанию);
- `scroll` — заставляет Web-обозреватель отобразить в элементе страницы полосу прокрутки, пользуясь которой можно прокручивать его содержимое;
- `hidden` — скрывает все то, что выходит за пределы элемента страницы;
- `auto` — аналогично `scroll` за тем исключением, что полоса прокрутки отображается только тогда, когда она реально необходима (значение по умолчанию для `<TEXTAREA>`).

Поддерживается IE начиная с 4.0.

padding

Задает отступ между элементом страницы и различными границами. Заменяет атрибуты `padding-top`, `padding-right`, `padding-bottom` и `padding-left`.

```
padding: {padding-top} [{padding-right}] [{padding-bottom}]
↳[{padding-left}];
```

Может быть задано от одного до четырех значений. Если задано одно значение, оно применяется ко всем четырем границам элемента. Если задано два значения, первое относится к верхней и нижней границам, а второе — к левой и правой. Если задано три значения, то первое применяется к верхней границе, второе — к левой и правой, третье — к нижней. Значение по умолчанию `0`, для `<TD>` `1`.

Поддерживается IE и N начиная с 4.0.

padding-bottom

Задает отступ между элементом страницы и нижней границей.

```
padding-bottom: {Y}|{Y}%;
```

Координата может быть задана как абсолютной величиной, так и процентом от высоты родителя. Значение по умолчанию `0`, для тегов `<TD>` `1`.

Поддерживается IE и N начиная с 4.0.

padding-left

Задает отступ между элементом страницы и левой границей.

```
padding-left: {X}|{X}%;
```

Координата может быть задана как абсолютной величиной, так и процентом от ширины родителя. Значение по умолчанию 0, для тегов <TD> 1.

Поддерживается IE и N начиная с 4.0.

padding-right

Задает расстояние между элементом страницы и правой границей.

```
padding-right: {X}|{X}%;
```

Координата может быть задана как абсолютной величиной, так и процентом от ширины родителя. Значение по умолчанию 0, для тегов <TD> 1.

Поддерживается IE и N начиная с 4.0.

padding-top

Задает расстояние между элементом страницы и верхней границей.

```
padding-top: {Y}|{Y}%;
```

Координата может быть задана как абсолютной величиной, так и процентом от высоты родителя. Значение по умолчанию 0, для тегов <TD> 1.

Поддерживается IE и N начиная с 4.0.

page-break-after

Устанавливает, будет ли после текущего элемента при печати Web-страницы производиться прогон листа.

```
page-break-after: auto|always|empty string;
```

Доступны три значения:

- auto (значение по умолчанию) — передает управление размещением информации на листе операционной системе;
- always — заставляет принтер прогнать лист после печати текущего элемента страницы;
- empty string — запрещает принтеру делать это в любом случае.

Поддерживается IE начиная с 4.0.

page-break-before

Устанавливает, будет ли перед текущим элементом при печати Web-страницы производиться прогон листа.

```
page-break-before: auto|always|empty string;
```

Доступны три значения:

- `auto` (значение по умолчанию) — передает управление размещением информации на листе операционной системе;
- `always` — заставляет принтер прогнать лист перед печатью текущего элемента страницы;
- `empty string` — запрещает принтеру делать это в любом случае.

Поддерживается IE начиная с 4.0.

position

Задает тип позиционирования текущего элемента страницы.

```
position: static|absolute|relative;
```

Доступны три значения:

- `static` (значение по умолчанию) — задает статическое позиционирование, при котором элемент страницы отображается внутри общего "потока" текста, т. е. не свободно. Значения атрибутов `bottom`, `left`, `right` и `top` при этом не принимаются Web-обозревателем во внимание;
- `absolute` — задает абсолютное свободное позиционирование. Значения атрибутов `bottom`, `left`, `right` и `top` при этом задают абсолютные координаты элемента страницы относительно родителя;
- `relative` — задает относительное свободное позиционирование. Значения атрибутов `bottom`, `left`, `right` и `top` при этом задают смещения координат элемента страницы от точки, в которой бы он был отображен, будь атрибут `position` установлен в `static`.

Поддерживается IE и N начиная с 4.0.

right

Задает горизонтальную позицию правой границы свободно позиционированного элемента относительно родителя.

```
right: auto|{X}|{X}%;
```

Координата может быть задана как абсолютной величиной, так и процентом от ширины родителя. Предопределенное значение `auto` заставляет Web-обозреватель позиционировать элемент самостоятельно. Значение по умолчанию `auto`.

Поддерживается IE начиная с 4.0.

ruby-align

Задает способ выравнивания текста аннотации, заданного тегом <RT>. Используется только для тега <RUBY>.

```
ruby-align: auto|left|center|right|distribute-letter|distribute-space|
↳line-edge;
```

Доступно семь значений:

- auto (значение по умолчанию) — заставляет Web-обозреватель самому выбрать способ выравнивания текста аннотации;
- left — выравнивает текст аннотации по левому краю;
- center — по центру;
- right — по правому краю;
- distribute-letter — "распределяет" текст аннотации по всему доступному пространству;
- distribute-space — то же самое, что distribute-letter, только с добавлением дополнительного пространства между словами;
- line-edge — "прижимает" текст аннотации к близлежащей границе элемента страницы или центрирует, если таковых поблизости нет.

Поддерживается IE начиная с 5.0.

ruby-overhang

Устанавливает способ перекрытия текстом аннотации смежного текста. Доступен только для тега <RUBY>.

```
ruby-overhang: auto|whitespace|none;
```

Доступны три значения:

- auto (значение по умолчанию) — задает перекрытие любого смежного текста;
- whitespace — заставляет текст аннотации перекрывать только свободное пространство;
- none — заставляет текст аннотации перекрывать только смежный с ним текст.

Поддерживается IE начиная с 5.0.

ruby-position

Задает местоположение текста аннотации. Доступен только для тега <RUBY>.

```
ruby-position: above|inline;
```

Доступны два значения: `above` (значение по умолчанию) помещает текст аннотации над аннотируемым текстом, а `inline` — внутри его.

Поддерживается IE начиная с 5.0.

scrollbar-3dlight-color

Задает цвет верхней и левой границ полосы прокрутки, ее бегунка и стрелок.

```
scrollbar-3dlight-color: {Цвет};
```

Поддерживается IE начиная с 5.5.

scrollbar-arrow-color

Задает цвет стрелок на кнопках полосы прокрутки.

```
scrollbar-arrow-color: {Цвет};
```

Поддерживается IE начиная с 5.5.

scrollbar-base-color

Задает цвет бегунка и кнопок-стрелок полосы прокрутки.

```
scrollbar-base-color: {Цвет};
```

Поддерживается IE начиная с 5.5.

scrollbar-darkshadow-color

Задает цвет "тени", "отбрасываемой" бегунком и кнопками прокрутки полосы прокрутки (цвет правых и нижних их граней).

```
scrollbar-darkshadow-color: {Цвет};
```

Поддерживается IE начиная с 5.5.

scrollbar-face-color

Задает цвет бегунка и кнопок прокрутки полосы прокрутки.

```
scrollbar-face-color: {Цвет};
```

Поддерживается IE начиная с 5.5.

scrollbar-highlight-color

Задает цвет "освещенной" части бегунка и кнопок прокрутки полосы прокрутки (цвет левых и верхних их граней).

```
scrollbar-highlight-color: {Цвет};
```

Поддерживается IE начиная с 5.5.

scrollbar-shadow-color

Задает цвет "неосвещенной" части бегунка и кнопок прокрутки полосы прокрутки (цвет правых и нижних их граней). Не путать с цветом "тени", задаваемым атрибутом `scrollbar-darkshadow-color`.

```
scrollbar-shadow-color: {Цвет};
```

Поддерживается IE начиная с 5.5.

scrollbar-track-color

Задает цвет рабочей части полосы прокрутки, т. е. той ее части, по которой перемещается бегунок.

```
scrollbar-track-color: {Цвет};
```

Поддерживается IE начиная с 5.5.

table-layout

Позволяет "зафиксировать" значения ширины ячеек таблицы. Это может сильно ускорить вывод больших таблиц на экран. Применяется только для тега `<TABLE>`.

```
table-layout: auto|fixed;
```

Доступны два значения: `auto` (значение по умолчанию) устанавливает ширину ячейки по ширине ее содержимого, а `fixed` использует для установки ширин ячеек значения атрибутов `WIDTH` или, если они не заданы, просто дает ячейкам равную ширину. Другими словами, задание значения `auto` позволит точно "подогнать" значения ширины ячеек, но таблица при этом будет выводиться очень долго. Значение `fixed` этого атрибута позволит Web-обозревателю вывести таблицу значительно быстрее, но Web-дизайнер должен будет сам задать значения ширины ячеек.

Поддерживается IE начиная с 5.0.

text-align

Задает горизонтальное выравнивание текста.

```
text-align: left|right|center|justify;
```

Доступны четыре значения:

- `left` (значение по умолчанию) — выравнивает текст по левому краю;
- `right` — по правому краю;
- `center` — по центру;
- `justify` — по обоим краям (по ширине).

Поддерживается IE начиная с 3.02; значение `justify` поддерживается начиная с 4.0. Поддерживается N начиная с 4.0.

text-align-last

Задает горизонтальное выравнивание последней строки абзаца.

```
text-align-last: auto|inherit|left|right|center|justify;
```

Доступны шесть значений:

- `auto` (значение по умолчанию) — выравнивает последнюю строку абзаца так же, как остальные строки (основываясь на значении атрибута `text-align`);
- `inherit` — так же, как выровнен текст родителя;
- `left` — по левому краю;
- `right` — по правому краю;
- `center` — по центру;
- `justify` — по обоим краям (по ширине).

Поддерживается IE начиная с 5.5.

text-autospace

Позволяет установить, будет ли добавлять дополнительное пространство между фрагментами текста, написанными на разных языках.

```
text-autospace: none|ideograph-alpha|ideograph-numeric|
↳ideograph-parenthesis|ideograph-space;
```

Доступно пять значений:

- `none` (значение по умолчанию) — запрещает добавлять дополнительное пространство между фрагментами текста;
- `ideograph-alpha` — добавляет дополнительное пространство между иероглифическими и неиероглифическими (латинскими, кириллическими, греческими и т. д.) фрагментами текста;
- `ideograph-numeric` — добавляет дополнительное пространство между иероглифическим текстом и цифрами;
- `ideograph-parenthesis` — добавляет дополнительное пространство между иероглифическим текстом и круглыми скобками;
- `ideograph-space` — увеличивает ширину пробелов, граничащих с иероглифическим текстом.

Поддерживается IE начиная с 5.0.

text-decoration

Задает специальное оформление текста: подчеркнутый, зачеркнутый и т. п.

```
text-decoration: none|underline|overline|line-through|blink;
```

Доступно пять значений:

- none — отменяет любое специальное оформление (значение по умолчанию для большинства тегов);
- underline — подчеркивает текст (значение по умолчанию для тегов <A>, <INS> и <U>);
- overline — "надчеркивает" текст;
- line-through — зачеркивает текст (значение по умолчанию для тегов <DEL>, <S> и <STRIKE>);
- blink — заставляет текст мерцать (IE не поддерживается).

Поддерживается IE начиная с 3.02 для текстовых элементов страницы и начиная с 4.0 для нетекстовых. Поддерживается N начиная с 4.0.

text-indent

Задает отступ красной строки.

```
text-indent: {Отступ}|{Отступ}%;
```

Отступ может быть задан как абсолютной величиной, так и процентом от ширины родителя. Значение по умолчанию 0.

Поддерживается IE и N начиная с 4.0.

text-justify

Задает тип текста по ширине. Значение атрибута text-align при этом должно быть равно justify.

```
text-justify: auto|newspaper|distribute|distribute-all-lines|
↳distribute-center-last|inter-word|inter-ideograph|inter-cluster|
↳kashida;
```

Доступны девять значений:

- auto (значение по умолчанию) — отдает управление выравниванием по ширине на усмотрение Web-обозревателя;
- newspaper — выравнивает строки, изменяя расстояния между словами и между символами;
- distribute — аналогично newspaper и предназначено для азиатских языков (тайский и пр.);

- distribute-all-lines — аналогично distribute за тем исключением, что последняя строка абзаца также подвергается полному выравниванию. Предназначено для иероглифических языков;
- distribute-center-last — не реализовано;
- inter-word — выравнивает строки, изменяя только расстояния между словами;
- inter-ideograph — выравнивает строки иероглифического текста, изменяя расстояния между словами и между иероглифами;
- inter-cluster — выравнивает строки текста на азиатских языках, не содержащих пробелов между словами;
- kashida — выравнивает строки текста на арабском языке, изменяя ширину самих символов.

Поддерживается IE начиная с 5.0.

text-kashida-space

Задает процент, на который будут расширяться символы арабского языка при выравнивании текста по ширине. Можно использовать только, если атрибут text-justify равен auto, distribute, kashida или newspaper.

```
text-kashida-space: {Расширение}%|inherit;
```

Величина отступа может быть задана как процент свободного пространства между символами, на которое они могут расширяться. Значение 0% (используется по умолчанию) означает, что расширение символов недопустимо, а вместо них будет расширяться свободное пространство; значение 100% — что допустимо расширение только символов, но не свободного пространства.

Поддерживается IE начиная с 5.5.

text-transform

Задает преобразование регистра символов текста.

```
text-transform: none|capitalize|uppercase|lowercase;
```

Доступны четыре значения:

- none (значение по умолчанию) — отключает любые преобразования регистра символов;
- capitalize — преобразует первую букву каждого слова текста в верхний регистр;
- uppercase — преобразует все символы текста в верхний регистр;
- lowercase — в нижний регистр.

Поддерживается IE и N начиная с 4.0.

text-underline-position

Задает местонахождение линии подчеркивания: выше или ниже текста. Имеет смысл, если атрибут `text-decoration` равен `underline` или `overline`.

```
text-underline-position: below|above;
```

Доступны два значения: `below` (значение по умолчанию) помещает линию подчеркивания под текстом, а `above` — над текстом ("надчеркивание").

Поддерживается IE начиная с 5.5.

top

Задает вертикальную позицию верхней границы свободно позиционированного элемента относительно родителя.

```
top: auto|{Y}|{Y}%;
```

Координата может быть задана как абсолютной величиной, так и процентом от высоты родителя. Предопределенное значение `auto` заставляет Web-обозреватель позиционировать элемент самостоятельно. Значение по умолчанию `auto`.

Поддерживается IE и N начиная с 4.0.

unicode-bidi

Задает поведение встроенных элементов при изменении направления письма с помощью атрибута `direction`.

```
unicode-bidi: normal|embed|bidi-override;
```

Доступны три значения:

- `normal` (значение по умолчанию) — меняет направление письма и у родителя;
- `embed` — меняет направление письма только у встроенного элемента;
- `bidi-override` — аналогично `embed` за тем исключением, что направление письма меняется согласно значению атрибута `direction`, независимо от локальных установок Web-обозревателя.

Поддерживается IE начиная с 5.0.

vertical-align

Задает вертикальное выравнивание элемента страницы внутри родителя.

```
vertical-align: auto|baseline|sub|super|top|text-top|middle|bottom|
↳text-bottom;
```

Доступно девять значений:

- `auto` — выравнивает элемент страницы согласно значению атрибута `layout-flow`;
- `baseline` (значение по умолчанию) — задает выравнивание базовой линии элемента страницы по базовой линии родителя;
- `sub` — превращает текст в нижний индекс;
- `super` — превращает текст в верхний индекс;
- `top` — выравнивает верх элемента страницы по самому верху родителя;
- `text-top` — выравнивает верх текста элемента страницы по верху текста родителя;
- `middle` — выравнивает центр элемента страницы по центру родителя;
- `bottom` — выравнивает низ элемента страницы по низу родителя;
- `text-bottom` — выравнивает низ текста элемента страницы по низу текста родителя.

Поддерживается IE начиная с 4.0.

visibility

Позволяет сделать элемент страницы видимым или невидимым.

```
visibility: inherit|visible|hidden;
```

Доступно три значения:

- `inherit` (значение по умолчанию) — заставляет элемент страницы вести себя так же, как родитель ("наследовать" "видимость");
- `visible` — делает элемент страницы видимым;
- `hidden` — невидимым.

Поддерживается IE и N начиная с 4.0.

white-space

Задает, будут ли строки текста, содержащегося в элементе страницы, автоматически переноситься, если они не помещаются в нем по ширине.

```
white-space: normal|nowrap|pre;
```

Доступны три значения:

- `normal` (значение по умолчанию) — включает автоматической перенос длинных строк;

- `nowrap` — отключает автоматической перенос строк. Чтобы "разорвать" строку вручную, вставьте в нужном месте тег `<BR>` (N не поддерживается);
- `pre` — не поддерживается.

Поддерживается IE начиная с 5.5 и N начиная с 4.0.

width

Задает ширину свободно позиционированного элемента.

```
width: auto|{X}|{X}%;
```

Ширина может быть задана как абсолютной величиной, так и процентом от ширины родителя. Предопределенное значение `auto` заставляет Web-обозреватель устанавливать ширину элемента самостоятельно. Значение по умолчанию `auto`.

Поддерживается IE и N начиная с 4.0.

word-break

Включает или отключает поддержку переноса строк по словам (а не только по пробелам) для текстов, содержащих фрагменты на разных языках.

```
work-break: normal|break-all|keep-all;
```

Доступны три значения:

- `normal` (значение по умолчанию) — разрешает строкам "разрываться" по слову;
- `break-all` — предназначено для текстов на азиатских языках с небольшими иноязычными фрагментами;
- `keep-all` — предназначено для текстов, включающих фрагменты на иероглифических языках.

Поддерживается IE начиная с 5.0.

word-spacing

Задает дополнительное пространство, добавляемое между словами в тексте.

```
word-spacing: normal|{Величина};
```

Значение этого атрибута может быть задано либо абсолютной величиной в одной из поддерживаемых CSS единиц измерения, либо предопределенным значением `normal`, задающим стандартную величину расстояния между символами. Значение по умолчанию `normal`.

Поддерживается IE начиная с 4.01.

word-wrap

Устанавливает, будет ли строка, выходящая за границы элемента страницы и не содержащая пробелов, переноситься по словам.

```
word-wrap: normal|break-word;
```

Доступны два значения: `normal` (значение по умолчанию) запрещает переносить строки по словам, а `break-word` разрешает.

Поддерживается IE начиная с 5.5.

writing-mode

Задает направление строк текста: горизонтальное или вертикальное.

```
writing-mode: lr-tb|tb-rl;
```

Доступны два значения. `lr-tb` (значение по умолчанию) задает обычное горизонтальное расположение строк текста; текст пишется слева направо и сверху вниз. `tb-rl` поворачивает текст на 90° по часовой стрелке; при этом он будет писаться сверху вниз и справа налево.

Поддерживается IE начиная с 5.5.

z-index

Задает порядок перекрытия свободно позиционированными объектами друг друга.

```
z-index: auto|{Порядок перекрытия};
```

Порядок перекрытия задается положительным или отрицательным целым числом. При этом элементы с большим значением этого атрибута будут перекрывать элементы с меньшим значением. Предопределенное значение `auto` задает порядок перекрытия по умолчанию, когда элементы, определенные в HTML-коде раньше, перекрываются заданными позже. Значение по умолчанию `auto`.

Поддерживается IE и N начиная с 4.0.

zoom

Задает масштаб отображения элемента страницы.

```
zoom: normal|{Масштаб}|{Масштаб}%;
```

Масштаб может быть задан как числом с плавающей точкой, обозначающим степень увеличения или уменьшения, так и процентной величиной. Предопределенное значение `normal` задает масштаб 1.0 или 100%. Значение по умолчанию `normal`.

Поддерживается IE начиная с 5.5.

Псевдостили гиперссылок

Псевдостили применяются к гиперссылкам <A> в особых случаях.

active

Применяется к активным гиперссылкам, т. е. гиперссылкам, на которых находится фокус ввода пользователя. Аналогичен атрибуту ALINK тега <BODY>.

```
{Задание стиля гиперссылки}:active {Определение стиля}
```

По умолчанию в IE активные гиперссылки выделяются красным цветом.

Поддерживается IE начиная с 4.0.

hover

Применяется к гиперссылкам, когда пользователь помещает над ними курсор мыши.

```
{Задание стиля гиперссылки}:hover {Определение стиля}
```

По умолчанию в IE гиперссылки подчеркиваются, когда пользователь помещает над ними курсор мыши.

Поддерживается IE начиная с 4.0.

link

Применяется к не посещенным еще пользователем гиперссылкам. Аналогичен атрибуту LINK тега <BODY>.

```
{Задание стиля гиперссылки}:link {Определение стиля}
```

По умолчанию в IE непосещенные гиперссылки отображаются синим цветом.

Поддерживается IE начиная с 3.02.

visited

Применяется к уже посещенным пользователем гиперссылкам. Аналогичен атрибуту VLINK тега <BODY>.

```
{Задание стиля гиперссылки}:visited {Определение стиля}
```

По умолчанию в IE посещенные гиперссылки отображаются бордовым цветом.

Поддерживается IE начиная с 3.02.

Пример использования псевдостилей гиперссылок

Давайте рассмотрим небольшой пример таблицы стилей, переопределяющей стили гиперссылок в какой-либо Web-странице. Разберем подробно каждую строчку этой таблицы стилей.

```
<STYLE>
BODY {font-size: 12pt; color: blue; }
```

Здесь мы переопределяем размер шрифта и цвет текста тела Web-страницы. Он будет отображаться двенадцатипунктовым кеглем и синим цветом.

```
A:link { color: black; }
```

А для гиперссылок мы выберем красный цвет. Заметьте, что установки псевдостиля `active` перекроют установки текста тега `<BODY>`.

```
A:active { color: lime; }
```

Активная гиперссылка будет ярко-зеленой.

```
A:visited { color: indigo; }
```

Уже посещенная — цвета индиго.

```
A:hover { color: lime; text-decoration: none }
```

А та, на которую пользователь "укажет" мышью — тоже ярко-зеленой и неподчеркнутой.

```
</STYLE>
```

Вот и все. Можете создать небольшую примерную Web-страничку и проверить эту таблицу стилей на практике.

Псевдостили текста

Псевдостили применяются к некоторым элемента текстовых абзацев, например, к первой строке абзаца или к первой букве первой строки.

first-letter

Применяется к первой букве первой строки абзаца. Может использоваться для создания буквиц.

```
{Задание стиля абзаца}:first-letter {Определение стиля}
```

По умолчанию в IE первые буквы первых строк абзацев никак не выделяются.

Поддерживается IE начиная с 5.5.

first-line

Применяется к первой строке абзаца.

```
{Задание стиля абзаца}:first-line {Определение стиля}
```

По умолчанию в IE первые строки абзацев никак не выделяются.

Поддерживается IE начиная с 5.5.

Правила

Правила используются в таблицах стилей для особых нужд.

charset

Задает текстовую кодировку для внешней таблицы стилей. Может использоваться только во внешних таблицах стилей; должна быть первой строкой в файле. Коды текстовых кодировок указаны в табл. П1.6.

```
@charset {Кодировка};
```

Пример использования.

```
@charset "windows-1251";
```

Поддерживается IE начиная с 4.0.

font-face

Задает файл загружаемого шрифта для использования в Web-странице.

```
@font-face {Определение загружаемого шрифта};
```

Определение загружаемого шрифта имеет следующий формат:

```
{font-family: {Имя шрифта}
src: url({Интернет-адрес файла шрифта}) }
```

Пример использования.

```
@font-face {
  font-family: comic;
  src: url(http://valid_url/comic_font_file.eot); }
```

Поддерживается IE начиная с 4.0.

import

Импортирует внешнюю таблицу стилей.

```
@import url("{Интернет-адрес файла таблицы стилей}");
```

Пример использования.

```
@import url("http://www.somesite.ru/someStyleSheet.css");
```

Поддерживается IE начиная с 4.0.

page

Используется при задании размеров, ориентации и полей печатной страницы в таблице стилей.

```
@page {Селектор страницы} { {Правила} }
```

Селектор страницы может принимать одно из трех значений:

- `:first` — первая печатная страница;
- `:left` — четная страница;
- `:right` — нечетная страница.

Пример использования правила `@page`.

```
@page :first { margin-top: 2cm; margin-bottom: 2cm }
```

Здесь мы задали для первой распечатанной страницы поля по два сантиметра сверху и снизу.

Поддерживается IE начиная с 5.5.

Прочее

Объявление *important*

Используется для задания неперекрываемых установок стиля.

```
{Установки стиля}!important
```

Пример использования.

```
<STYLE>
   P { color: red !important }
</STYLE>
<P STYLE="color: green">Этот текст останется красным.</P>
```

Здесь установки, сделанные для текстового абзаца `<P>`, не будут перекрыты в дальнейшем.

Поддерживается IE начиная с 4.0.

Единицы измерения

Все единицы измерения, поддерживаемые CSS, перечислены в табл. П2.1.

Таблица П2.1. *Единицы измерения, поддерживаемые CSS*

Единица измерения	Обозначение
Высота буквы M текущего шрифта	em
Высота буквы x текущего шрифта	ex
Пикселы	px
Пункты	pt
Пики	pc
Дюймы	in
Миллиметры	mm
Сантиметры	cm

Коды цветов

Все коды цветов, поддерживаемые CSS, перечислены в табл. П2.2.

Таблица П2.2. *Коды цветов, поддерживаемые CSS*

Обозначение	Цвет	Код
aliceBlue	Блекло-голубой	F0F8FF
antuqueWhite	Античный белый	FAEBD7
aqua	Синий	00FFFF
aquamarine	Аквамарин	7FFFD4
azure	Лазурь	F0FFFF
beige	Бежевый	F5F5DC
bisque	Бисквитный	FFE4C4
black	Черный	000000
blanchedalmond	Светло-кремовый	FFEBCD
blue	Голубой	0000FF
blueviolet	Светло-фиолетовый	8A2BE2

Таблица П2.2 (продолжение)

Обозначение	Цвет	Код
brown	Коричневый	A52A2A
burlywood	Старого дерева	DEB887
cadetblue	Блеклый серо-голубой	5F9EA0
chartreuse	Фисташковый	7FFF00
chocolate	Шоколадный	D2691E
coral	Коралловый	FF7F50
cornflowerblue	Васильковый	6495ED
cornsilk	Темно-зеленый	FFF8DC
crimson	Малиновый	DC143C
cyan	Циан	00FFFF
darkblue	Темно-голубой	00008B
darkcyan	Темный циан	008B8B
darkgoldenrod	Темный красно-золотой	B8860B
darkgray	Темно-серый	A9A9A9
darkgreen	Темно-зеленый	006400
darkkhaki	Темный хаки	BDB76B
darkmagenta	Темный фуксин	8B008B
darkolivegreen	Темно-оливковый	556B2F
darkorange	Темно-оранжевый	FF8C00
darkorchid	Темно-орхидейный	9932CC
darkred	Темно-красный	8B0000
darksalmon	Темный оранжево-розовый	E9967A
darkseagreen	Темный морской волны	8FBC8F
darkslateblue	Темный серовато-синий	483D8B
darkslategray	Темный синевато-серый	2F4F4F
darkturquoise	Темно-бирюзовый	00CED1
darkviolet	Темно-фиолетовый	9400D3
deeppink	Темно-розовый	FF1493

Таблица П2.2 *(продолжение)*

Обозначение	Цвет	Код
deepskyblue	Темный небесно-синий	00BFFF
dimgray	Тускло-серый	696969
dodgerblue	Тускло-васильковый	1E90FF
firebrick	Огнеупорного кирпича	B22222
floralwhite	Цветочно-белый	FFFAF0
forestgreen	Лесной зеленый	228B22
fuchsia	Фуксия	FF00FF
fainsboro	Светлый серо-фиолетовый	DCDCDC
ghostwhite	Туманно-белый	F8F8FF
gold	Золотой	FFD700
goldenrod	Красного золота	DAA520
gray	Серый	808080
green	Зеленый	008000
greenyellow	Желто-зеленый	ADFF2F
honeydew	Свежего меда	F0FFF0
hotpink	Ярко-розовый	FF69B4
indianred	Ярко-красный	CD5C5C
indigo	Индиго	4B0082
ivory	Слоновой кости	FFFFF0
khaki	Хаки	F0E68C
lavender	Бледно-лиловый	E6E6FA
lavenderblush	Бледный розово-лиловый	FFF0F5
lawngreen	Зеленой травы	7CFC00
lemonchiffon	Лимонный	FFFACD
lightblue	Светло-голубой	ADD8E6
lightcoral	Светло-коралловый	F08080
lightcyan	Светлый циан	E0FFFF
lightgray	Светло-серый	D3D3D3

Таблица П2.2 (продолжение)

Обозначение	Цвет	Код
lightgreen	Светло-зеленый	9CEE90
lightpink	Светло-розовый	FFB6C1
lightsalmon	Светлый оранжево-розовый	FFA07A
lightseagreen	Светлый морской волны	20B2AA
lightskyblue	Светлый небесно-синий	87CEFA
lightslategray	Светлый синевато-серый	778899
lightsteelblue	Светло-стальной	B0C4DE
lightyellow	Светло-желтый	FFFFE0
lime	Известковый	00FF00
limegreen	Зеленовато-известковый	32CD32
linen	Льняной	FAF0E6
magenta	Фуксин	FF00FF
maroon	Оранжево-розовый	800000
mediumaquamarine	Умеренно-аквамариновый	66CDAA
mediumblue	Умеренно-голубой	0000CD
mediumorchid	Умеренно-орхидейный	BA55D3
mediumpurple	Умеренно-пурпурный	9370DB
mediumseagreen	Умеренный морской волны	3CB371
mediumslateblue	Умеренный серовато-синий	7B68EE
mediumspringgreen	Умеренный синевато-серый	00FA9A
mediumturquoise	Умеренно-бирюзовый	48D1CC
mediumvioletred	Умеренный красно-фиолетовый	C71585
midnightblue	Ночной синий	191970
mintcream	Мятно-кремовый	F5FFFA
mistyrose	Туманно-розовый	FFE4E1
moccasin	Болотный	FFE4B5
navajowhite	Грязно-серый	FFDEAD
navy	Темно-синий	000080

Таблица П2.2 (продолжение)

Обозначение	Цвет	Код
oldlace	Старого коньяка	FDF5E6
olive	Оливковый	808000
olivedrab	Тускло-коричневый	6B8E23
orange	Оранжевый	FFA500
orangered	Красно-оранжевый	FF4500
orchid	Орхидейный	DA70D6
palegoldenrod	Бледно-золотой	EEE8AA
palegreen	Бледно-зеленый	98FB98
paleturquoise	Бледно-бирюзовый	AFEEEE
plaevioletred	Бледный красно-фиолетовый	DB7093
papayawhip	Дыни	FFEFD5
peachpuff	Персиковый	FFDAB9
peru	Коричневый	CD853F
pink	Розовый	FFC0CB
plum	Сливовый	DDA0DD
powderblue	Туманно-голубой	B0E0E6
purple	Пурпурный	800080
rosybrown	Розово-коричневый	BC8F8F
royalblue	Королевский голубой	4169E1
saddlebrown	Старой кожи	8B4513
salmon	Оранжево-розовый	FA8072
sandybrown	Рыже-коричневый	F4A460
seagreen	Морской волны	2E8B57
seashell	Морской пены	FFF5EE
sienna	Охра	A0522D
silver	Серебристый	C0C0C0
skyblue	Небесно-голубой	87CEEB
slateblue	Серовато-синий	6A5ACD

***Таблица П2.2** (окончание)*

Обозначение	Цвет	Код
slategray	Синевато-серый	708090
snow	Снежный	FFFAFA
springgreen	Весенний зеленый	00FF7F
steelblue	Голубовато-стальной	4682B4
tan	Желтовато-коричневый	D2B48C
teal	Чайный	008080
thistle	Чертополоха	D8BFD8
tomato	Томатный	FF6347
turquoise	Бирюзовый	40E0D0
violet	Фиолетовый	EE82EE
wheat	Пшеничный	F5DEB3
white	Белый	FFFFFF
whitesmoke	Белый дымчатый	F5F5F5
yellow	Желтый	FFFF00
yellowgreen	Желто-зеленый	9ACD32

ПРИЛОЖЕНИЕ 3

Объектная модель документа

В этом приложении описываются все объекты, свойства, методы и события, поддерживаемые двумя популярнейшими программами Web-обозревателей.

Свойства

Свойства перечислены в алфавитном порядке. Поясняется семантика свойства, приводится формальное описание формата, дается информация о поддержке основными Web-обозревателями.

above

Возвращает ссылку на слой, находящийся выше текущего, или на окно Web-обозревателя, если текущий слой — самый верхний.

```
{Объект слоя}.above;
```

Доступно только для чтения.

Поддерживается N начиная с 4.0.

accelerator

Позволяет указать, содержит ли текст элемента страницы клавишу-ускоритель. Клавиша-ускоритель — это особая клавиша на клавиатуре, при нажатии которой вместе с клавишей <Alt> происходит переход к данному элементу страницы.

```
{Объект стиля}.accelerator[ = "true|false"];
```

Доступны два значения: `true` указывает, что текст содержит клавишу-ускоритель, а `false` — что не содержит. Значения по умолчанию нет. Свой-

ство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.0.

accessKey

Задает или возвращает клавишу-ускоритель для быстрого доступа к элементу страницы. Например, можно присвоить клавишу-ускоритель полю ввода, и при нажатии ее вместе с клавишей <Alt> поле ввода будет активизироваться.

```
{Объект}.accessKey[ = "{Клавиша-ускоритель}"];
```

Поддерживается IE начиная с 4.0 для элементов управления и внедренных объектов и начиная с 5.0 для всех остальных элементов.

action

Задает или возвращает интернет-адрес серверной программы, которой форма передаст введенные пользователем данные.

```
{Объект формы}.action[ = "{Интернет-адрес серверной программы}"];
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

activeElement

Возвращает указатель на активный элемент страницы.

```
document.activeElement;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

align (<CAPTION> и <LEGEND>)

Задает или возвращает выравнивание элемента заголовка таблицы (`<CAPTION>`) или группы элементов управления (`<LEGEND>`).

```
{Объект}.align[ = "bottom|center|left|right|top"];
```

Доступны пять значений:

- `bottom` — выравнивание по нижнему краю и по центру;
- `center` — выравнивание по центру;
- `left` — выравнивание по левому краю;
- `right` — выравнивание по правому краю;
- `top` — выравнивание по верхнему краю и по центру.

Поддерживается IE начиная с 3.02 для <CAPTION> и начиная с 4.0 для <LEGEND>.

align (<DIV>, <Hn>, <HR>, <P>, элементы таблицы)

Задает или возвращает выравнивание содержимого элемента страницы.

```
{Объект}.align[ = "left|center|right|justify"];
```

Доступны четыре значения:

- left — выравнивание по левой границе (значение по умолчанию);
- center — выравнивание по центру (значение по умолчанию для <TH>);
- right — выравнивание по правой границе;
- justify — выравнивание по ширине (доступно только для <DIV>, <Hn> и <P>).

Поддерживается IE начиная с 3.02 для <COL>, <COLGROUP>, <DIV>, <HR>, <P>, <TD>, <TH>, <TR> и начиная с 4.0 для <Hn>, <TBODY>, <TFOOT>, <THEAD>.

align (<IFRAME> и <TABLE>)

Задает или возвращает выравнивание "плавающего" фрейма (<IFRAME>) или таблицы (<TABLE>).

```
{Объект}.align[ = "left|center|right"];
```

Доступны три значения:

- left — выравнивание по левому краю доступного пространства (значение по умолчанию);
- center — выравнивание по центру доступного пространства;
- right — выравнивание по правому краю доступного пространства.

Поддерживается IE начиная с 3.02 для <TABLE> и начиная с 4.0 для <IFRAME>.

align (<IMG>, элементы управления, внедренные элементы)

Задает или возвращает местоположение рисунка, элемента управления или внедренного элемента относительно "обтекающего" его текста.

```
{Объект}.align[ = "left|right|top|texttop|middle|absmiddle|baseline|
↳bottom|absbottom"];
```

Доступны следующие значения:

- left — элемент смещается влево, а текст обтекает его справа (значение по умолчанию);

- `right` — элемент смещается вправо, а текст обтекает его слева;
- `top` — вершина элемента выравнивается по верхней границе строки;
- `texttop` — вершина элемента выравнивается по вершине самого высокого символа строки;
- `middle` — элемент центрируется в строке;
- `absmiddle` — центр элемента выравнивается точно по центру строки;
- `baseline` — низ элемента выравнивается по базовой линии строки;
- `bottom` — низ элемента выравнивается по низу строки;
- `absbottom` — низ изображения выравнивается по низу самого низко сидящего символа строки.

Поддерживается IE начиная с 3.02 для `<APPLET>`, `<IMG>`, `<INPUT>`, `<OBJECT>` и начиная с 4.0 для `<EMBED>`, `<FIELDSET>`, `<SELECT>`.

align (CSS)

Задает или возвращает принудительное выравнивание элемента страницы относительно родителя.

```
{Объект стиля}.align[ = "none|left|right"];
```

Доступны три значения:

- `none` (значение по умолчанию) — заставляет элемент страницы появляться там, где он задан;
- `left` — принудительно выравнивает элемент страницы по левому краю;
- `right` — по правому краю.

Поддерживается N начиная с 4.0.

aLink

Задает или возвращает цвет активных гиперссылок в документе.

```
body.aLink[ = "{Цвет}"];
```

Поддерживается IE начиная с 4.0.

alinkColor

Задает или возвращает цвет активных гиперссылок в документе.

```
document.alinkColor[ = "{Цвет}"];
```

Значение по умолчанию `#0000FF`.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

allowTransparency

Задает или возвращает прозрачность фрейма, обычного или "плавающего".

```
{Объект фрейма}.allowTransparency[ = false|true];
```

По умолчанию (значение `false` свойства) фон фрейма определяется отображаемой в нем Web-страницей. Если же свойству присвоить значение `true`, то фон фрейма можно задать каким угодно; если же он не задан, используется цвет фона Web-страницы, где определен фрейм.

Поддерживается IE начиная с 5.5.

alt

Задает или возвращает "альтернативный" текст, отображаемый на месте изображения, если оно еще не загружено.

```
{Объект}.alt[ = "{"Альтернативный" текст}"];
```

Поддерживается IE начиная с 3.02 для `<IMG>` и `<INPUT TYPE="image">` и начиная с 4.0 для `<AREA>`.

altHTML

Задает "альтернативный" HTML-код, подставляемый вместо внедренного объекта при ошибке его загрузки.

```
{Внедренный объект}.altHTML = "{"Альтернативный" HTML-код}";
```

Доступно только для записи.

Поддерживается IE начиная с 4.0.

altKey

Задает или возвращает состояние клавиши <Alt>.

```
event.altKey[ = false|true];
```

Доступно два значения: `false` говорит о том, что клавиша <Alt> не нажата, а `true` — что нажата.

Поддерживается IE начиная с 4.0 только для чтения и начиная с 5.0 для чтения и записи.

altLeft

Задает или возвращает состояние левой клавиши <Alt>.

```
event.altLeft[ = false|true];
```

Доступно два значения: `false` говорит о том, что левая клавиша <Alt> не нажата, а `true` — что нажата.

Поддерживается IE начиная с 5.5 и только на операционных системах Windows NT 4.0 и Windows 2000.

appCodeName

Возвращает имя кода программы Web-обозревателя.

```
navigator.appCodeName;
```

Доступно только для чтения. IE и N возвращают строку "Mozilla".

Поддерживается IE начиная с 3.02 для `navigator` и начиная с 4.0 для `clientInformation`. Поддерживается N начиная с 2.0.

appMinorVersion

Возвращает младший номер версии программы Web-обозревателя.

```
navigator.appMinorVersion;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

appName

Возвращает имя программы Web-обозревателя.

```
navigator.appName;
```

Доступно только для чтения.

Поддерживается IE начиная с 3.02 для `navigator` и начиная с 4.0 для `clientInformation`. Поддерживается N начиная с 2.0.

appVersion

Возвращает версию программы Web-обозревателя и сведения о компьютерной платформе, на которой она работает.

```
navigator.appVersion;
```

Доступно только для чтения. IE возвращает строку вида:

```
5.0 (compatible; MSIE 5.5; Windows 98; Win 9x 4.90)
```

N возвращает строку

```
4.05 (Win32; I)
```

Поддерживается IE начиная с 3.02 для `navigator` и начиная с 4.0 для `clientInformation`. Поддерживается N начиная с 2.0.

autocomplete

Задает или возвращает возможность "автозаполнения" форм и полей ввода текста.

```
{Объект}.autocomplete[ = "on|off"];
```

Доступны два значения: `on` включает автозаполнение, а `off` отключает. Значения по умолчанию нет, точнее, оно зависит от настроек пользователя.

Поддерживается IE начиная с 5.0.

availHeight

Возвращает высоту доступной области экрана компьютера, исключая Панель задач Windows.

```
screen.availHeight;
```

Доступно только для чтения.

Поддерживается IE и N начиная с 4.0.

availLeft

Возвращает горизонтальную координату первого пиксела, не занятого Панелью задач и аналогичными ей элементами интерфейса Windows.

```
screen.availLeft;
```

Доступно только для чтения.

Поддерживается N начиная с 4.0.

availTop

Возвращает вертикальную координату первого пиксела, не занятого Панелью задач и аналогичными ей элементами интерфейса Windows.

```
screen.availTop;
```

Доступно только для чтения.

Поддерживается N начиная с 4.0.

availWidth

Возвращает ширину доступной области экрана компьютера, исключая Панель задач Windows.

```
screen.availWidth;
```

Доступно только для чтения.

Поддерживается IE и N начиная с 4.0.

background

Задает или возвращает интернет-адрес файла фонового рисунка для документа или таблицы.

```
{Объект}.background[ = "{Интернет-адрес файла изображения}"];
```

Поддерживается IE начиная с 3.02 и N начиная с 4.0.

background (CSS)

Комбинированное свойство, заменяющее свойства backgroundAttachment, backgroundColor, backgroundImage, backgroundPosition и backgroundRepeat. Задает или возвращает все свойства фона элемента страницы в один прием. Значения этих свойств могут располагаться в любом порядке.

```
{Объект стиля}.background[ = "[{background-color}] [{background-image}]
↳[{background-repeat}] [{background-attachment}]
↳[{background-position}]"];
```

Значение по умолчанию "transparent none repeat scroll 0% 0%".

Поддерживается IE начиная с 3.02; задание значений backgroundPosition и backgroundRepeat поддерживается начиная с 4.0.

backgroundAttachment

Позволяет "зафиксировать" фоновый рисунок так, чтобы он не прокручивался вместе с содержимым Web-страницы.

```
{Объект стиля}.backgroundAttachment[ = "scroll|fixed"];
```

Доступны два значения: scroll (значение по умолчанию) заставляет фоновый рисунок прокручиваться вместе с содержимым страницы, а fixed "фиксирует" его. Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 4.0, в составе атрибута background — с 3.02.

backgroundColor

Задает или возвращает фоновый цвет Web-страницы или ее элемента.

```
{Объект стиля}.backgroundColor[ = "{Цвет}|transparent"];
```

Предопределенное значение transparent задает "прозрачный" фон. Оно же является значением по умолчанию. Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 4.0, в составе атрибута background — с 3.02. Поддерживается N начиная с 4.0.

backgroundImage

Задает или возвращает фоновый рисунок Web-страницы или ее элемента.

```
{Объект стиля}.backgroundColor[ = "url({Интернет-адрес файла рисунка})|
↳none"];
```

Предопределенное значение none отключает фоновый рисунок. Оно же является значением по умолчанию. Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 4.0, в составе атрибута background — с 3.02. Поддерживается N начиная с 4.0.

backgroundPosition

Комбинированное свойство, заменяющее свойства backgroundPositionX и backgroundPositionY. Задает или возвращает местонахождение фонового рисунка.

```
{Объект стиля}.backgroundPosition[ = "[{backgroundPositionX}]
↳[{backgroundPositionY}]"];
```

Значение по умолчанию 0% 0%. Если задана только одна координата, то она считается горизонтальной, а для вертикальной координаты принимается значение 50%.

Поддерживается IE начиная с 4.0.

backgroundPositionX

Задает или возвращает горизонтальную координату фонового рисунка.

```
{Объект стиля}.backgroundPositionX[ = "{X}|{X}%|left|center|right"];
```

Координата может быть задана целым числом (абсолютная координата), процентом от соответствующего размера фонового рисунка (относительная координата) или предопределенным значением. Доступны три предопределенных значения: left, center и right, задающие выравнивание фонового рисунка на странице по левому краю, по центру и по правому краю соответственно. Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения. Значение по умолчанию 0%.

Поддерживается IE начиная с 4.0.

backgroundPositionY

Задает или возвращает вертикальную координату фонового рисунка.

```
{Объект стиля}.backgroundPositionY[ = "{Y}|{Y}%|top|center|bottom"];
```

Координата может быть задана целым числом (абсолютная координата), процентом от соответствующего размера фонового рисунка (относительная координата) или предопределенным значением. Доступны три предопределенных значения: `top`, `center` и `bottom`, задающие выравнивание по верху, по центру и по низу страницы соответственно. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения. Значение по умолчанию `0%`.

Поддерживается IE начиная с 4.0.

backgroundRepeat

Задает или возвращает порядок повторения фонового рисунка на Web-странице или ее элементе.

```
{Объект стиля}.backgroundRepeat[ = "repeat|no-repeat|repeat-x|repeat-y"];
```

Доступны четыре значения:

- `repeat` (значение по умолчанию) — заставляет фоновый рисунок повторяться по горизонтали и вертикали;
- `no-repeat` — запрещает фоновому рисунку повторяться;
- `repeat-x` — заставляет фоновый рисунок повторяться только по горизонтали;
- `repeat-y` — заставляет фоновый рисунок повторяться только по вертикали.

Поддерживается IE начиная с 4.0.

balance

Задает или возвращает значение стереобаланса при воспроизведении фонового звука, заданного тегом `<BGSOUND>`.

```
{Объект фонового звука}.balance[ = "{Значение баланса}"];
```

Может принимать значения от −10 000 (крайняя левая позиция) до 10 000 (крайняя правая позиция). Значение по умолчанию 0 (центр).

Поддерживается IE начиная с 4.0.

BaseHref

Возвращает интернет-адрес Web-страницы, в которую встроен объект `<OBJECT>`.

```
{Объект}.BaseHref;
```

Доступно только для чтения.

Поддерживается IE начиная с 3.02.

behavior

Задает или возвращает поведение прокручиваемого текста в элементе <MARQUEE>.

```
{Объект}.behavior[ = "scroll|alternate|slide"];
```

Доступны три значения:

- `scroll` (значение по умолчанию) — задает обычную прокрутку, т. е. текст уходит в одну сторону и появляется с противоположной;
- `alternate` — заставляет текст двигаться взад-вперед;
- `slide` — заставляет текст прокрутиться до конца и остановиться.

Поддерживается IE начиная с 3.02.

behavior (CSS)

Задает или возвращает поведение DHTML для элемента страницы.

```
{Объект стиля}.behavior[ = "url({Интернет-адрес файла поведения})|
↪url(#{Имя элемента ActiveX, реализующего это поведение})|
↪url(#default#{Имя встроенного поведения})"];
```

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.0.

behaviorCookie

Возвращает cookie, идентифицирующий поведение отображения, на котором произошло текущее событие.

```
event.behaviorCookie;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.5.

behaviorPart

Возвращает значение, идентифицирующее поведение отображения, на котором произошло текущее событие.

```
event.behaviorPart;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.5.

below

Возвращает ссылку на слой, находящийся ниже текущего, или `null`, если текущий слой — самый нижний.

```
{Объект слоя}.below;
```

Доступно только для чтения.

Поддерживается N начиная с 4.0.

bgColor

Задает или возвращает цвет фона.

```
{Объект}.bgColor[ = "{Цвет}"];
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

bgProperties

Задает или возвращает поведение фонового рисунка документа: будет ли он перемещаться при прокрутке содержимого или нет.

```
body.bgProperties[ = ""|"fixed"];
```

Если в качестве значения этого атрибута указана пустая строка (значение по умолчанию), фоновый рисунок будет перемещаться при прокрутке содержимого документа. Если указано значение `fixed`, то фон прокручиваться не будет.

Поддерживается IE начиная с 3.02.

blockDirection

Возвращает направление написания текста: слева направо или справа налево.

```
{Объект}.blockDirection;
```

Могут возвращаться два значения: `ltr`, обозначающее направление написания текста слева направо, и `rtl` — справа налево.

Поддерживается IE начиная с 5.0.

border (<FRAMESET> и <IFRAME>)

Задает или возвращает значение толщины границы между фреймами в пикселах. Значение 0 убирает границу совсем.

```
{Объект фрейма}.border[ = "{Толщина границы в пикселах}"];
```

Значение по умолчанию 5.

Поддерживается IE начиная с 4.0.

border (<IMG>, LayoutRect и <TABLE>)

Задает или возвращает значение толщины в пикселах границы, отображаемой вокруг рисунка, или границы таблицы. Значение 0 убирает границу совсем.

```
{Объект}.border[ = "{Толщина границы в пикселах}"];
```

В N доступно только для чтения.

Поддерживается IE начиная с 3.02 и N начиная с 3.0.

border (CSS)

Задает или возвращает все свойства границы элемента страницы в один прием. Заменяет свойства `borderColor`, `borderStyle` и `borderWidth`. Значения этих свойств могут располагаться в любом порядке.

```
{Объект стиля}.border[ = "[{borderColor}] [{borderStyle}]
↳[{borderWidth}]"];
```

Значение по умолчанию `medium none`.

Поддерживается IE начиная с 4.0.

borderBottom

Задает или возвращает все свойства нижней границы элемента страницы в один прием. Заменяет свойства `borderBottomColor`, `borderBottomStyle` и `borderBottomWidth`. Значения этих свойств могут располагаться в любом порядке.

```
{Объект стиля}.borderBottom[ = "[{borderBottomColor}]
↳[{borderBottomStyle}] [{borderBottomWidth}]"];
```

Значение по умолчанию `medium none`.

Поддерживается IE начиная с 4.0.

borderBottomColor

Задает цвет нижней границы элемента страницы.

```
{Объект стиля}.borderBottomColor[ = "{Цвет}"];
```

Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 4.0.

borderBottomStyle

Задает или возвращает тип нижней границы элемента страницы.

```
{Объект стиля}.borderBottomStyle[ = "none|dotted|dashed|solid|double|
↳groove|ridge|inset|outset"];
```

Доступны девять значений:

- none (значение по умолчанию) — запрещает рисование границы;
- dotted — рисует точечную линию;
- dashed — рисует штриховую линию;
- solid — рисует сплошную линию;
- double — рисует двойную сплошную линию;
- groove — рисует трехмерную вдавленную границу;
- ridge — рисует трехмерную выпуклую границу;
- inset — рисует трехмерную "ступеньку вверх";
- outset — рисует трехмерную "ступеньку вниз".

Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 4.0.

borderBottomWidth

Задает или возвращает значение толщины нижней границы элемента страницы.

```
{Объект стиля}.borderBottomWidth[ = "medium|thin|thick|{Толщина}"];
```

Толщина может быть задана числовым значением. Также доступны три предопределенных значения: thin, medium и thick, рисующие тонкую, среднюю и толстую линию соответственно. Значение по умолчанию medium. Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE и N начиная с 4.0.

borderCollapse

Задает, будут ли границы ячеек и общая граница таблицы сливаться в одну или нет, или возвращает этот признак. Применяется только для тега <TABLE>.

```
{Объект таблицы}.style.borderCollapse[ = "separate|collapse"];
```

Доступны два значения: separate (значение по умолчанию) разделяет рамку таблицы и границы ее ячеек, а collapse по возможности объединяет их. Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 5.0.

borderColor

Задает или возвращает цвет границы.

```
{Объект}.borderColor[ = "{Цвет}"];
```

Поддерживается IE начиная с 3.02 для таблиц и их элементов и начиная с 4.0 для фреймов.

borderColor (CSS) (IE)

Задает или возвращает цвет всех границ элемента страницы. Заменяет свойства borderTopColor, borderRightColor, borderBottomColor и borderLeftColor.

```
{Объект стиля}.borderColor[ = "{borderTopColor} [{borderRightColor}]
↳[{borderBottomColor}] [{borderLeftColor}]"];
```

Может быть задано от одного до четырех значений. Если задано одно значение, оно применяется ко всем четырем границам. Если задано два значения, первое относится к верхней и нижней границе, а второе — к левой и правой. Если задано три значения, то первое применяется к верхней границе, второе — к левой и правой, третье — к нижней. Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 4.0.

borderColor (CSS) (N)

Задает цвет границ элемента страницы.

```
{Объект стиля}.borderColor[ = "none|{Цвет}"];
```

Может быть задано либо цветовое значение, либо none, обозначающее отсутствие границ.

Поддерживается N начиная с 4.0.

borderColorDark

Задает или возвращает цвет "затененной" части трехмерной границы таблицы или ее элемента.

```
{Объект таблицы}.borderColorDark[ = "{Цвет}"];
```

Поддерживается IE начиная с 3.02.

borderColorLight

Задает или возвращает цвет "освещенной" части трехмерной границы таблицы или ее элемента.

```
{Объект таблицы}.borderColorLight[ = "{Цвет}"];
```

Поддерживается IE начиная с 3.02.

borderLeft

Задает или возвращает все свойства левой границы элемента страницы в один прием. Заменяет свойства `borderLeftColor`, `borderLeftStyle` и `borderLeftWidth`. Значения этих свойств могут располагаться в любом порядке.

```
{Объект стиля}.borderLeft[ = "[{borderLeftColor}] [{borderLeftStyle}]
↳[{borderLeftWidth}]"];
```

Значение по умолчанию `medium none`.

Поддерживается IE начиная с 4.0.

borderLeftColor

Задает или возвращает цвет левой границы элемента страницы.

```
{Объект стиля}.borderLeftColor[ = "{Цвет}"];
```

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

borderLeftStyle

Задает или возвращает тип левой границы элемента страницы.

```
{Объект стиля}.borderLeftStyle[ = "none|dotted|dashed|solid|double|
↳groove|ridge|inset|outset"];
```

Доступны девять значений:

- none (значение по умолчанию) — запрещает рисование границы;
- dotted — рисует точечную линию;
- dashed — рисует штриховую линию;
- solid — рисует сплошную линию;
- double — рисует двойную сплошную линию;
- groove — рисует трехмерную вдавленную границу;
- ridge — рисует трехмерную выпуклую границу;
- inset — рисует трехмерную "ступеньку вверх";
- outset — рисует трехмерную "ступеньку вниз".

Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 4.0.

borderLeftWidth

Задает или возвращает значение толщины левой границы элемента страницы.

```
{Объект стиля}.borderLeftWidth[ = "medium|thin|thick|{Толщина}"];
```

Толщина может быть задана числовым значением. Также доступны три предопределенных значения: thin, medium и thick, рисующие тонкую, среднюю и толстую линию соответственно. Значение по умолчанию medium. Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE и N начиная с 4.0.

borderRight

Задает или возвращает все свойства правой границы элемента страницы в один прием. Заменяет свойства borderRightColor, borderRightStyle и borderRightWidth. Значения этих свойств могут располагаться в любом порядке.

```
{Объект стиля}.borderRight[ = "[{borderRightColor}] [{borderRightStyle}]
↳[{borderRightWidth}]"];
```

Значение по умолчанию medium none.

Поддерживается IE начиная с 4.0.

borderRightColor

Задает или возвращает цвет правой границы элемента страницы.

```
{Объект стиля}.borderRightColor[ = "{Цвет}"];
```

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

borderRightStyle

Задает или возвращает тип правой границы элемента страницы.

```
{Объект стиля}.borderRightStyle[ = "none|dotted|dashed|solid|double|
↳groove|ridge|inset|outset"];
```

Доступны девять значений:

- `none` (значение по умолчанию) — запрещает рисование границы;
- `dotted` — рисует точечную линию;
- `dashed` — рисует штриховую линию;
- `solid` — рисует сплошную линию;
- `double` — рисует двойную сплошную линию;
- `groove` — рисует трехмерную вдавленную границу;
- `ridge` — рисует трехмерную выпуклую границу;
- `inset` — рисует трехмерную "ступеньку вверх";
- `outset` — рисует трехмерную "ступеньку вниз".

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

borderRightWidth

Задает или возвращает значение толщины правой границы элемента страницы.

```
{Объект стиля}.borderRightWidth[ = "medium|thin|thick|{Толщина}"];
```

Толщина может быть задана числовым значением. Также доступны три предопределенных значения: `thin`, `medium` и `thick`, рисующие тонкую, среднюю и толстую линию соответственно. Значение по умолчанию `medium`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE и N начиная с 4.0.

borderStyle

Задает или возвращает тип сразу всех границ элемента страницы в один прием. Заменяет свойства borderTopStyle, borderRightStyle, borderBottomStyle и borderLeftStyle.

```
{Объект стиля}.borderStyle[ = "none|dotted|dashed|solid|double|groove|
↳ridge|inset|outset"];
```

Доступны девять значений:

- none (значение по умолчанию) — запрещает рисование границы;
- dotted — рисует точечную линию (N не поддерживается);
- dashed — рисует штриховую линию (N не поддерживается);
- solid — рисует сплошную линию;
- double — рисует двойную сплошную линию;
- groove — рисует трехмерную вдавленную границу;
- ridge — рисует трехмерную выпуклую границу;
- inset — рисует трехмерную "ступеньку вверх";
- outset — рисует трехмерную "ступеньку вниз".

Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE и N начиная с 4.0.

borderTop

Задает или возвращает все свойства верхней границы элемента страницы в один прием. Заменяет свойства borderTopColor, borderTopStyle и borderTopWidth. Значения этих свойств могут располагаться в любом порядке.

```
{Объект стиля}.borderTop[ = "[{borderTopColor}] [{borderTopStyle}]
↳[{borderTopWidth}]"];
```

Значение по умолчанию medium none.

Поддерживается IE начиная с 4.0.

borderTopColor

Задает или возвращает цвет верхней границы элемента страницы.

```
{Объект стиля}.borderTopColor[ = "{Цвет}"];
```

Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 4.0.

borderTopStyle

Задает или возвращает тип верхней границы элемента страницы.

```
{Объект стиля}.borderTopStyle[ = "none|dotted|dashed|solid|double|
↳groove|ridge|inset|outset"];
```

Доступны девять значений:

- none (значение по умолчанию) — запрещает рисование границы;
- dotted — рисует точечную линию;
- dashed — рисует штриховую линию;
- solid — рисует сплошную линию;
- double — рисует двойную сплошную линию;
- groove — рисует трехмерную вдавленную границу;
- ridge — рисует трехмерную выпуклую границу;
- inset — рисует трехмерную "ступеньку вверх";
- outset — рисует трехмерную "ступеньку вниз".

Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 4.0.

borderTopWidth

Задает или возвращает значение толщины верхней границы элемента страницы.

```
{Объект стиля}.borderTopWidth[ = "medium|thin|thick|{Толщина}"];
```

Толщина может быть задана числовым значением. Также доступны три предопределенных значения: thin, medium и thick, рисующие тонкую, среднюю и толстую линию соответственно. Значение по умолчанию medium. Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE и N начиная с 4.0.

borderWidth

Задает или возвращает значения толщины всех границ элемента страницы. Заменяет свойства borderTopWidth, borderRightWidth, borderBottomWidth и borderLeftWidth.

```
{Объект стиля}.borderWidth[ = "{borderTopWidth} [{borderRightWidth}]
↳[{borderBottomWidth}] [{borderLeftWidth}]"];
```

Может быть задано от одного до четырех значений. Если задано одно значение, оно применяется ко всем четырем границам. Если задано два значения, первое относится к верхней и нижней границе, а второе — к левой и правой. Если задано три значения, то первое применяется к верхней границе, второе — к левой и правой, третье — к нижней.

Толщина может быть задана числовым значением. Также доступны три предопределенных значения: `thin`, `medium` и `thick`, рисующие тонкую, среднюю и толстую линию соответственно. Значение по умолчанию `medium`.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

bottom (TextRectangle)

Задает или возвращает вертикальную координату нижней границы объекта `TextRectangle` в пикселах.

```
{Объект TextRectangle}.bottom[ = {Y}];
```

Поддерживается IE начиная с 5.0.

bottom (CSS)

Задает или возвращает вертикальную позицию нижней границы свободно позиционированного элемента относительно родителя.

```
{Объект стиля}.bottom[ = "auto|{Y}|{Y}%"];
```

Координата может быть задана как абсолютной величиной, так и процентом от высоты родителя. Предопределенное значение `auto` заставляет Web-обозреватель позиционировать элемент самостоятельно. Значение по умолчанию `auto`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

bottomMargin

Задает или возвращает расстояние от нижней границы окна Web-обозревателя до нижней границы содержимого Web-страницы в пикселах.

```
body.bottomMargin[ = "{Расстояние}"];
```

Значение по умолчанию 15.

Поддерживается IE начиная с 4.0.

boundingHeight

Возвращает высоту содержимого объекта TextRange в пикселах.

```
{Объект TextRange}.boundingHeight;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.01.

boundingLeft

Возвращает горизонтальную координату левой границы содержимого объекта TextRange относительно родителя в пикселах.

```
{Объект TextRange}.boundingLeft;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.01.

boundingTop

Возвращает вертикальную координату верхней границы содержимого объекта TextRange относительно родителя в пикселах.

```
{Объект TextRange}.boundingTop;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.01.

boundingWidth

Возвращает ширину содержимого объекта TextRange в пикселах.

```
{Объект TextRange}.boundingWidth;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.01.

browserLanguage

Возвращает язык программы Web-обозревателя или операционной системы.

```
navigator.browserLanguage;
```

В IE версии 4.x это свойство возвращает язык программы Web-обозревателя. В IE версий 5.x и более поздних возвращается язык операционной системы клиента.

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

bufferDepth

Задает или возвращает количество двоичных разрядов, используемых для представления цвета при буферизации графических изображений в памяти.

```
screen.bufferDepth[ = 0|-1|1|4|8|15|16|24|32];
```

Доступны следующие значения:

- 0 — отключает буферизацию изображений в памяти (значение по умолчанию);
- -1 — для буферизации в памяти используется такое же представление цвета, как и на экране;
- 1, 4, 8, 15, 16, 24, 32 — задают количество двоичных разрядов для представления цвета при буферизации в памяти.

Поддерживается IE начиная с 4.0.

button

Задает или возвращает номер кнопки мыши, нажатой пользователем.

```
event.button[ = 0|1|2|3|4|5|6|7];
```

Доступны следующие значения:

- 0 — ни одна из кнопок мыши не была нажата (значение по умолчанию);
- 1 — была нажата левая кнопка;
- 2 — правая;
- 3 — левая и правая одновременно;
- 4 — средняя;
- 5 — левая и средняя одновременно;
- 6 — правая и средняя одновременно;
- 7 — все кнопки одновременно.

В IE версий 4.x доступно только для чтения.

Поддерживается IE начиная с 4.0.

cancelBubble

Задает или возвращает признак прохождения событий через иерархию объектов страницы.

```
event.cancelBubble[ = false|true];
```

По умолчанию (значение false) события последовательно проходят от потомков к родителям через всю иерархию объектов страницы. Присвоив этому свойству значение true, можно прервать это прохождение.

Поддерживается IE начиная с 4.0.

canHaveChildren

Возвращает признак, может ли элемент страницы иметь потомков.

```
{Объект}.canHaveChildren;
```

Возвращается значение true (элемент страницы может иметь потомков) либо false (не может). Доступно только для чтения.

Поддерживается IE начиная с 5.0.

canHaveHTML

Задает или возвращает признак, может ли элемент страницы содержать HTML-текст.

```
{Объект}.canHaveHTML[ = true|false];
```

Доступны значения true (элемент страницы может содержать HTML-текст) либо false (не может). Доступно только для чтения для всех объектов, за исключением defaults.

Поддерживается IE начиная с 5.5.

caption

Возвращает указатель на заголовок <CAPTION> таблицы <TABLE> или null, если такового нет.

```
{Объект таблицы}.caption;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

cellIndex

Возвращает позицию ячейки <TD> в коллекции cells.

```
{Объект ячейки таблицы}.cellIndex;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

cellPadding

Задает или возвращает расстояние между границей и содержимым ячейки таблицы в пикселах либо как процент от доступного пространства.

```
{Объект таблицы}.cellPadding[ = "{Расстояние}"];
```

Значение по умолчанию 1.

Поддерживается IE начиная с 3.02.

cellSpacing

Задает или возвращает расстояние между ячейками таблицы в пикселах либо как процент от доступного пространства.

```
{Объект таблицы}.cellSpacing[ = "{Расстояние}"];
```

Значение по умолчанию 2.

Поддерживается IE начиная с 3.02.

checked

Задает или возвращает включенное состояние флажка или радиокнопки.

```
{Объект флажка или радиокнопки}.checked[ = true|false];
```

По умолчанию все флажки и радиокнопки выключены (значение `false`). Чтобы включить флажок или радиокнопку, присвойте этому свойству значение `true`.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

class

Задает или возвращает стилевой класс элемента страницы.

```
{Объект}.class[ = "{Стилевой класс, определенный в таблице стилей}"];
```

Поддерживается IE начиная с 4.0.

classid

Задает или возвращает уникальный идентификатор класса элемента ActiveX.

```
{Внедренный объект}.classid[ = "{Уникальный идентификатор класса}"];
```

Поддерживается IE начиная с 3.02.

clear

Задает или возвращает поведение разорванного тегом
 текста при "обтекании" им некоторых элементов страницы, таких как изображения.

```
{Объект разрыва}.clear[ = "all|left|right|none"];
```

Доступны четыре значения:

- all — запрещает тексту "обтекать" элемент страницы с обеих сторон;
- left — с левой стороны;
- right — с правой стороны;
- none — разрешает тексту "обтекать" элемент страницы.

Поддерживается IE начиная с 3.02.

clear (CSS)

Задает или возвращает поведение текста при "обтекании" им некоторых элементов страницы, таких как изображения. Атрибут задается для текста, а не для элемента страницы, который он будет "обтекать".

```
{Объект стиля}.clear[ = "none|left|right|both"];
```

Доступны четыре значения:

- none (значение по умолчанию) — разрешает тексту "обтекать" элемент страницы;
- left — запрещает тексту "обтекать" элемент страницы с левой стороны;
- right — с правой стороны;
- all — с обеих сторон.

Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE и N начиная с 4.0.

clientHeight

Возвращает высоту клиентской области элемента страницы без учета всех отступов, границ и полос прокрутки.

```
{Объект}.clientHeight;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0 для внедренных элементов, элементов управления и таблиц и начиная с 5.0 для остальных элементов.

clientLeft

Возвращает горизонтальную координату левой границы клиентской области элемента страницы без учета всех отступов, границ и полос прокрутки.

```
{Объект}.clientLeft;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0 для внедренных элементов, элементов управления и таблиц и начиная с 5.0 для остальных элементов.

clientTop

Возвращает вертикальную координату верхней границы клиентской области элемента страницы без учета всех отступов, границ и полос прокрутки.

```
{Объект}.clientTop;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0 для внедренных элементов, элементов управления и таблиц и начиная с 5.0 для остальных элементов.

clientWidth

Возвращает ширину клиентской области элемента страницы без учета всех отступов, границ и полос прокрутки.

```
{Объект}.clientWidth;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0 для внедренных элементов, элементов управления и таблиц и начиная с 5.0 для остальных элементов.

clientX

Задает или возвращает горизонтальную координату курсора мыши относительно клиентской области окна Web-обозревателя.

```
event.clientX[ = {X}];
```

В IE версий 4.x доступно только для чтения.

Поддерживается IE начиная с 4.0.

clientY

Задает или возвращает вертикальную координату курсора мыши относительно клиентской области окна Web-обозревателя.

```
event.clientY[ = {Y}];
```

В IE версий 4.x доступно только для чтения.

Поддерживается IE начиная с 4.0.

clip

Задает или возвращает прямоугольный фрагмент элемента страницы, который будет видим.

```
{Объект стиля}.clip[ = "auto|rect({Верхняя граница} {Правая граница}
↳{Нижняя граница} {Левая граница})"];
```

Предопределенное значение `auto` задает видимость всего элемента страницы, это же значение по умолчанию. Для того чтобы ограничить видимую часть элемента страницы прямоугольным фрагментом, задаются координаты границ этого прямоугольника, разделенные пробелами, всего четыре.

Поддерживается IE начиная с 4.0.

clip.bottom

Задает или возвращает вертикальную координату нижней границы видимого прямоугольного фрагмента слоя.

```
{Объект слоя}.clip.bottom[ = "auto|{Y}"];
```

Поддерживается N начиная с 4.0.

clip.height

Задает или возвращает высоту видимого прямоугольного фрагмента слоя.

```
{Объект слоя}.clip.height[ = {Y}];
```

Поддерживается N начиная с 4.0.

clip.left

Задает или возвращает горизонтальную координату левой границы видимого прямоугольного фрагмента слоя.

```
{Объект слоя}.clip.left[ = {X}];
```

Поддерживается N начиная с 4.0.

clip.right

Задает или возвращает горизонтальную координату правой границы видимого прямоугольного фрагмента слоя.

```
{Объект слоя}.clip.right[ = {X}];
```

Поддерживается N начиная с 4.0.

clip.top

Задает или возвращает вертикальную координату верхней границы видимого прямоугольного фрагмента слоя.

```
{Объект слоя}.clip.top[ = {Y}];
```

Поддерживается N начиная с 4.0.

clip.width

Задает или возвращает ширину видимого прямоугольного фрагмента слоя.

```
{Объект слоя}.clip.width[ = {X}];
```

Поддерживается N начиная с 4.0.

clipBottom

Возвращает вертикальную координату нижней границы видимого прямоугольного фрагмента элемента страницы.

```
{Объект}.currentStyle.clipBottom;
```

Предопределенное значение `auto` говорит о том, что элемент страницы полностью видим. Свойство доступно только для чтения.

Поддерживается IE начиная с 5.0.

clipLeft

Возвращает горизонтальную координату левой границы видимого прямоугольного фрагмента элемента страницы.

```
{Объект}.currentStyle.clipLeft;
```

Предопределенное значение `auto` говорит о том, что элемент страницы полностью видим. Свойство доступно только для чтения.

Поддерживается IE начиная с 5.0.

clipRight

Возвращает горизонтальную координату правой границы видимого прямоугольного фрагмента элемента страницы.

```
{Объект}.currentStyle.clipRight;
```

Предопределенное значение `auto` говорит о том, что элемент страницы полностью видим. Свойство доступно только для чтения.

Поддерживается IE начиная с 5.0.

clipTop

Возвращает вертикальную координату верхней границы видимого прямоугольного фрагмента элемента страницы.

```
{Объект}.currentStyle.clipTop;
```

Предопределенное значение `auto` говорит о том, что элемент страницы полностью видим. Свойство доступно только для чтения.

Поддерживается IE начиная с 5.0.

closed

Возвращает `true`, если окно Web-обозревателя закрыто, и `false` — в противном случае.

```
window.closed;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

code

Задает или возвращает интернет-адрес файла, содержащего откомпилированный класс Java.

```
{Внедренный объект}.code[ = "{Интернет-адрес файла с классом Java}"];
```

Поддерживается IE начиная с 4.0.

codeBase

Задает или возвращает интернет-адрес папки, где содержится дистрибутивный файл элемента ActiveX, Java-класса или расширения Web-обозревателя.

```
{Внедренный объект}.codeBase[ = "{Интернет-адрес дистрибутивного
↳файла}"];
```

Поддерживается IE начиная с 3.02.

codeType

Задает или возвращает тип данных MIME элемента ActiveX.

```
{Внедренный объект}.codeType[ = "{Тип данных MIME}"];
```

Поддерживается IE начиная с 3.02.

color

Задает или возвращает цвет шрифта или горизонтальной линии.

```
{Объект}.color[ = "{Цвет}"];
```

Поддерживается IE начиная с 3.02.

color (CSS)

Задает или возвращает цвет текста.

```
{Объект стиля}.color[ = "{Цвет}"];
```

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 3.02 для текстовых элементов страницы и начиная с 4.0 для нетекстовых. Поддерживается N начиная с 4.0.

colorDepth

Задает или возвращает количество двоичных разрядов, используемых для представления цвета на экране.

```
screen.colorDepth[ = 1|4|8|15|16|24|32];
```

Доступны значения `1`, `4`, `8`, `15`, `16`, `24`, `32`, задающие количество двоичных разрядов для представления цвета на экране.

Поддерживается IE и N начиная с 4.0.

cols (<FRAMESET>)

Задает или возвращает количество и значения ширины всех фреймов-колонок в наборе фреймов.

```
{Объект набора фреймов}.cols[ = "Список значений ширины фреймов-колонок,
↳разделенных запятыми"];
```

Значения ширины фреймов-колонок могут быть заданы как в пикселах, так и в процентах от ширины всего набора фреймов. Также может быть исполь-

зовано значение *, предписывающее Web-обозревателю отвести соответствующему фрейму все оставшееся место.

```
someFrameset.cols = "100, 20%, *";
```

Поддерживается IE начиная с 3.02.

cols (<TABLE>)

Задает или возвращает количество колонок в таблице.

```
{Объект таблицы}.cols[ = "{Количество колонок}"];
```

Поддерживается IE начиная с 3.02.

cols (<TEXTAREA>)

Задает или возвращает ширину области редактирования в символах текста.

```
{Объект области редактирования}.cols[ = "{Ширина в символах}"];
```

Значение по умолчанию 20.

Поддерживается IE начиная с 3.02.

colSpan

Задает или возвращает количество ячеек, которые должны быть объединены в одну.

```
{Объект ячейки}.colSpan[ = "{Количество ячеек, объединяемых в одну}"];
```

Значение этого свойства может быть изменено только после загрузки страницы.

Поддерживается IE начиная с 3.02.

compact

Задает или возвращает признак компактного отображения позиций списков.

```
{Объект списка}.compact[ = true|false];
```

По умолчанию все списки отображаются в обычном режиме (значение `false`). Чтобы включить компактное отображение (т. е. без дополнительных пробелов), присвойте этому свойству значение `true`.

Поддерживается IE начиная с 4.0.

complete

Возвращает `true`, если загрузка графического изображения завершена, и `false` — в противном случае.

```
{Объект рисунка}.complete;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

content

Задает или возвращает метаданные в теге <META>. Тип метаданных зависит от значений свойств httpEquiv или name.

```
{Объект метаданных}.content[ = "{Метаданные}"];
```

Поддерживается IE начиная с 3.02.

contentEditable

Задает или возвращает признак, дающий пользователю возможность редактирования содержимого Web-страницы или ее элемента.

```
{Объект}.contentEditable[ = "inherit|true|false"];
```

Доступны три значения:

- inherit (значение по умолчанию) — заставляет элемент страницы "наследовать" эту возможность от родителя;
- true — разрешает пользователю редактировать содержимое элемента страницы;
- false — запрещает редактирование.

Поддерживается IE начиная с 5.5.

contentOverflow

Возвращает true, если для форматирования страницы нужен новый объект LayoutRect, и false — в противном случае.

```
event.contentOverflow;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.5.

contentWindow

Возвращает ссылку на объект window фрейма <FRAME> или <IFRAME>.

```
{Объект фрейма}.contentWindow;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.5.

cookie

Задает или возвращает значение cookie, связанное с Web-страницей.

```
document.cookie[ = "[expires={Дата}][; ][domain={Доменный адрес}][; ]
[path={Путь}][; ][secure][; ][{Параметр}={Значение}]"];
```

В качестве значения присваивается или возвращается список пар вида {Параметр}={Значение}, разделенных точками с запятой. Также могут указываться следующие специальные параметры;

- expires={Дата} — задает дату "устаревания" cookie. Если указанная дата находится в прошлом, cookie удаляется немедленно, если это текущая дата — после закрытия Web-обозревателя;
- domain={Доменный адрес} — задает доменный адрес, страницы с которого могут получать доступ к этому cookie;
- path={Путь} — задает путь на текущем сайте, страницы с которого могут получать доступ к этому cookie;
- secure — разрешает доступ к cookie только по защищенному протоколу.

Пример значения cookie.

```
userid=3453973320985; expires=Fri, 31 Dec 2001 23:59:59 GMT;"
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

cookieEnabled

Возвращает true, если пользователь разрешил Web-обозревателю сохранять на жестком диске компьютера cookie, и false — в противном случае.

```
navigator.cookieEnabled;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

coords

Задает или возвращает координаты "горячей" области в карте-изображении. Зависит от значения атрибута SHAPE.

```
{Объект карты-изображения}.coords[ = "{Координаты "горячей" области}"];
```

Доступны три типа "горячих" областей и, соответственно, типа значения свойства coords:

- rect — прямоугольник, coords="{X1}, {Y1}, {X2}, {Y2}", где X1 и Y1 — координаты верхнего левого, а X2 и Y2 — правого нижнего угла прямоугольника;

- `circle` — круг, `coords="{X центра}, {Y центра}, {Радиус}"`;
- `poly` — многоугольник, `coords="{X1}, {Y1}, {X2}, {Y2}, {X3}, {Y3}..."`, где `Xn` и `Yn` — координаты соответствующей точки.

Поддерживается IE начиная с 3.02.

cpuClass

Возвращает тип процессора клиентского компьютера.

```
navigator.cpuClass;
```

Возвращается одно из пяти значений:

- `x86` — Intel-совместимый процессор;
- `68K` — процессор Motorola 68xxx;
- `Alpha` — процессор Digital Alpha;
- `PPC` — процессор Motorola PowerPC;
- `Other` — другой процессор.

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

current

Возвращает интернет-адрес текущей позиции в списке "истории".

```
history.current;
```

Доступно только для чтения.

Поддерживается N начиная с 3.0.

cssText

Задает или возвращает исходный текст определения стиля или таблицы стилей.

```
style.cssText[ = "{Исходный текст определения стиля или таблицы
↳стилей}"];
```

Поддерживается IE начиная с 5.0.

ctrlKey

Задает или возвращает состояние клавиши <Ctrl>.

```
event.ctrlKey[ = false|true];
```

Доступны два значения: `false` говорит о том, что клавиша <Ctrl> не нажата, а `true` — что нажата.

Поддерживается IE начиная с 4.0 только для чтения и начиная с 5.0 для чтения и записи.

ctrlLeft

Задает или возвращает состояние левой клавиши <Ctrl>.

```
event.ctrlLeft[ = false|true];
```

Доступны два значения: `false` говорит о том, что левая клавиша <Ctrl> не нажата, а `true` — что нажата.

Поддерживается IE начиная с 5.5 и только на операционных системах Windows NT 4.0 и Windows 2000.

cursor

Задает или возвращает форму курсора мыши, которую он принимает при наведении на элемент страницы.

```
{Объект стиля}.cursor[ = "auto|crosshair|default|hand|move|*-resize|text|
↳wait|help"];
```

Доступны следующие значения:

- `auto` (значение по умолчанию) — заставляет Web-обозреватель самому определять нужную форму курсора мыши;
- `crosshair` — крестообразный курсор;
- `default` — курсор по умолчанию, обычно стрелка;
- `hand` — "указующий перст";
- `move` — стрелка, указывающая "на все четыре стороны";
- `*-resize` (где вместо `*` может стоять `n`, `ne`, `nw`, `s`, `se`, `sw`, `e` или `w`) — "стрелка компаса", указывающая направление;
- `text` — текстовый курсор;
- `wait` — "песочные часы", курсор ожидания;
- `help` — стрелка с вопросительным знаком.

Поддерживается IE начиная с 4.0.

data (event)

Задает или возвращает массив строк, содержащий интернет-адреса ярлыков, "брошенных" в окно Web-обозревателя.

```
event.data[ = ({Массив строк})];
```

Поддерживается N начиная с 4.0.

data (<OBJECT>)

Задает или возвращает интернет-адрес файла данных, требуемых элементом ActiveX.

```
{Внедренный объект}.data[ = "{Интернет-адрес файла данных}"];
```

Поддерживается IE начиная с 3.02.

data (TextNode)

Задает или возвращает значение объекта `TextNode`.

```
{Объект}.data;
```

Поддерживается IE начиная с 5.0.

dataFld

Задает или возвращает поле таблицы базы данных, значение которого будет отображаться в данном элементе страницы.

```
{Объект}.dataFld[ = "{Имя поля таблицы базы данных}"];
```

Поддерживается IE начиная с 4.0.

dataFld (event)

Задает или возвращает имя поля таблицы базы данных, значение которого изменилось. Доступно при возникновении события `oncellchange`.

```
event.dataFld[ = "{Имя поля таблицы базы данных}"];
```

Поддерживается IE начиная с 5.0.

dataFormatAs

Задает или возвращает представление, используемое для отображения данных из полей таблицы.

```
{Объект}.dataFormatAs[ = "text|html|localized-text"];
```

Доступны три значения:

- `text` (значение по умолчанию) — задает текстовое представление;
- `html` — заставляет Web-обозреватель учитывать все теги HTML, встречающиеся в значениях полей;
- `localized-text` — работает аналогично `text`, за тем исключением, что при этом используются региональные настройки Windows.

Поддерживается IE начиная с 4.0.

dataPageSize

Задает или возвращает размер страницы данных, выводимых в один прием, в записях. Требуется, если нужно привязать таблицу к данным.

```
{Объект таблицы}.dataPageSize[ = "{Размер страницы данных}"];
```

Поддерживается IE начиная с 4.0.

dataSrc

Задает или возвращает объект-источник данных.

```
{Объект}.dataSrc[ = "#{Имя объекта-источника данных}"];
```

Поддерживается IE начиная с 4.0.

defaultCharset

Возвращает текстовую кодировку, заданную для Web-страницы по умолчанию.

```
document.defaultCharset;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

defaultChecked

Задает или возвращает состояние флажка или радиокнопки по умолчанию. Аналогично атрибуту CHECKED.

```
{Объект флажка или радиокнопки}.defaultChecked[ = true|false];
```

По умолчанию все флажки и радиокнопки изначально (сразу после загрузки Web-страницы или сброса формы) включены (значение `true`). Чтобы изначально отключить их, присвойте этому свойству значение `false`.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

defaultSelected

Задает или возвращает состояние позиции списка <OPTION> по умолчанию. Аналогично атрибуту SELECTED.

```
{Объект позиции списка}.defaultSelected[ = true|false];
```

По умолчанию все позиции списка изначально (сразу после загрузки Web-страницы или сброса формы) включены (значение true). Чтобы отключить их, присвойте этому свойству значение false.

Поддерживается IE начиная с 3.02 и N начиная с 3.0.

defaultStatus

Задает или возвращает значение, отображаемое Web-обозревателем в строке состояния окна по умолчанию, если не задано свойство status.

```
window.defaultStatus[ = "{Текст строки состояния}"];
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

defaultValue

Задает или возвращает содержимое элемента управления по умолчанию. Аналогично атрибуту VALUE.

```
{Объект}.defaultValue[ = "{Значение по умолчанию}"];
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

defer

Задает или возвращает состояние скрипта: отложено ли его выполнение или нет.

```
{Объект скрипта}.defer[ = true|false];
```

По умолчанию скрипты выполняются при загрузке Web-страницы (значение false). Если вы хотите отложить их выполнение до полной загрузки страницы, дайте этому свойству значение true.

Поддерживается IE начиная с 4.0.

description

Возвращает описание типа MIME или расширения Web-обозревателя.

```
{Объект}.description;
```

Доступно только для чтения.

Поддерживается N начиная с 3.0.

designMode

Задает или возвращает состояние документа: может ли он редактироваться пользователем.

```
document.designMode[ = "on|off"];
```

По умолчанию Web-страницу редактировать нельзя (значение `off`). Если вы хотите перевести Web-обозреватель в состояние, при котором пользователь сможет ее редактировать, присвойте этому свойству значение `on`.

Поддерживается IE начиная с 5.0.

dialogArguments

Возвращает значение или массив значений, переданных в диалоговое Web-окно.

```
window.dialogArguments;
```

Доступно исключительно для чтения и только для модальных или немодальных диалоговых Web-окон.

Поддерживается IE начиная с 4.0.

dialogHeight

Задает или возвращает высоту диалогового Web-окна в одной из единиц измерения, поддерживаемых CSS.

```
window.dialogHeight[ = "{Высота}"];
```

Доступно только для модальных или немодальных диалоговых Web-окон.

Поддерживается IE начиная с 4.0.

dialogLeft

Задает или возвращает горизонтальную координату левой стороны диалогового Web-окна в одной из единиц измерения, поддерживаемых CSS.

```
window.dialogLeft[ = "{X}"];
```

Доступно только для модальных или немодальных диалоговых Web-окон.

Поддерживается IE начиная с 4.0.

dialogTop

Задает или возвращает вертикальную координату верхней стороны диалогового Web-окна в одной из единиц измерения, поддерживаемых CSS.

```
window.dialogTop[ = "{Y}"];
```

Доступно только для модальных или немодальных диалоговых Web-окон.

Поддерживается IE начиная с 4.0.

dialogWidth

Задает или возвращает ширину диалогового Web-окна в одной из единиц измерения, поддерживаемых CSS.

```
window.dialogWidth[ = "{Ширина}"];
```

Доступно только для модальных или немодальных диалоговых Web-окон.

Поддерживается IE начиная с 4.0.

dir

Задает или возвращает порядок чтения текста.

```
{Объект}.dir[ = "ltr|rtl"];
```

По умолчанию текст читается слева направо (`ltr`). Значение `rtl` позволяет задать порядок чтения справа налево. Для документов, составленных на европейских языках, порядок чтения всегда слева направо (`ltr`).

Поддерживается IE начиная с 5.0.

direction

Задает или возвращает направление прокрутки текста в элементе <MARQUEE>.

```
{Объект}.direction[ = "{left|right|up|down}"];
```

Текст может прокручиваться налево (`left`; значение по умолчанию), направо (`right`), вверх (`up`) и вниз (`down`).

Поддерживается IE начиная с 3.02.

direction (CSS)

Задает или возвращает порядок чтения текста: слева направо или справа налево.

```
{Объект стиля}.direction[ = "ltr|rtl|inherit"];
```

Доступны три значения:

- `ltr` (значение по умолчанию) — задает порядок чтения слева направо;
- `rtl` — справа налево;
- `inherit` — заставляет наследовать порядок чтения у родителя.

Для документов, составленных на европейских языках, порядок чтения всегда слева направо (`ltr`). Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.0.

disabled

Задает или возвращает состояние элемента страницы: разрешен он для доступа пользователя или запрещен.

```
{Объект}.disabled[ = true|false];
```

По умолчанию доступ к элементу страницы разрешен (значение `true`). При этом пользователь может редактировать текст в поле ввода, включать и отключать флажки и радиокнопки, выбирать пункты из списков, нажимать командные кнопки и редактировать содержимое остальных элементов страницы, если для них было задано соответствующее значение свойства `contentEditable`. Чтобы запретить доступ к элементу страницы, присвойте этому свойству значение `false`.

Поддерживается IE начиная с 4.0 для элементов управления и начиная с 5.5 для остальных элементов страницы.

disabled (таблицы стилей)

Задает или возвращает состояние таблицы стилей: применяется ли она к Web-странице или нет.

```
{Объект таблицы стиля}.disabled[ = true|false];
```

По умолчанию таблица стиля применяется к Web-странице (значение `true`). Чтобы отменить ее действие, присвойте этому свойству значение `false`.

Поддерживается IE начиная с 4.0.

display

Задает или возвращает вид элемента страницы: встроенный, блочный, позиция списка или вообще не видимый на странице.

```
{Объект стиля}.display[ = "inline|block|none|inline-block|list-item|
↳table-header-group|table-footer-group"];
```

Доступны семь значений:

- `inline` (значение по умолчанию) — заставляет элемент страницы вести себя как встроенный. При этом он располагается внутри текста, и его позиция и размеры рассчитываются согласно позиции и размерам текста. Пример встроенного элемента — `<I>`;

- `block` — делает элемент страницы блочным. При этом он может быть свободно позиционирован. Пример блочного элемента — `<DIV>`;
- `none` — делает элемент страницы невидимым. При этом страница отображается так, будто этого элемента вообще не существует;
- `inline-block` — аналогично `inline`, но содержимое элемента страницы ведет себя как блочный элемент (N не поддерживается);
- `list-item` — аналогично `block`, но при этом элемент страницы считается позицией списка;
- `table-header-group` — заставляет элемент страницы отображаться после верхнего заголовка таблицы и перед всеми строками данных аналогично тегу `<THEAD>` (N не поддерживается);
- `table-footer-group` — заставляет элемент страницы отображаться перед основанием таблицы и после всех строк данных аналогично тегу `<TFOOT>` (N не поддерживается).

Поддерживается IE и N начиная с 4.0.

document (popup)

Возвращает ссылку на Web-страницу, загруженную во всплывающее окно.

```
{Объект всплывающего окна}.document;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.5.

document (window)

Возвращает ссылку на Web-страницу, загруженную в текущее окно Web-обозревателя.

```
window.document;
```

Доступно только для чтения.

Поддерживается N начиная с 2.0.

document (слой)

Возвращает ссылку на Web-страницу, загруженную в слой.

```
{Объект слоя}.document;
```

Доступно только для чтения.

Поддерживается N начиная с 4.0.

documentElement

Возвращает ссылку на самый верхний в иерархии объект, содержащийся в текущей Web-странице. Как правило, это объект <HTML>.

```
document.documentElement;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.0.

domain

Задает или возвращает домен Web-сервера, с которого была загружена Web-страница.

```
document.domain[ = "{Домен}"];
```

Изначально возвращает доменное имя Web-сервера. Может использоваться, чтобы сделать несколько Web-страниц "дружественными", если они загружены с разных серверов. Это может понадобиться, чтобы обойти систему безопасности серверов. Например, если две страницы в наборе фреймов загружены с серверов http://home.microsoft.com и http://download.microsoft.com, то, присвоив этому свойству значение `microsoft.com`, можно "подружить" эти страницы.

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

dropEffect

Задает или возвращает тип операции drag-n-drop, разрешенной объектом-приемником.

```
dataTransfer.dropEffect[ = "none|move|copy|link"];
```

Доступны четыре значения:

- `none` — вообще отменяет любые операции drag-n-drop (значение по умолчанию);
- `move` — разрешает операцию переноса;
- `copy` — копирования;
- `link` — создания ссылки или ярлыка.

Поддерживается IE начиная с 5.0.

dynsrc

Задает или возвращает интернет-адрес динамического содержимого, отображаемого вместо статичного рисунка. Динамическим содержимым для рисунка может быть видеофильм.

```
{Объект рисунка}.dynsrc[ = "{Интернет-адрес динамичного содержимого}"];
```

Если задан этот атрибут, то вместо обычного статичного рисунка будет отображен видеофильм, содержащийся в файле, интернет-адрес которого был задан в этом атрибуте.

Поддерживается IE начиная с 3.02.

effectAllowed

Задает или возвращает тип операции drag-n-drop, разрешенной объектом-источником.

```
dataTransfer.effectAllowed[ = "uninitialized|none|move|copy|link|
↳copyLink|copyMove|linkMove|all"];
```

Доступны девять значений:

- `unitialized` — задает операцию drag-n-drop, выполняемую по умолчанию самим Web-обозревателем (значение по умолчанию);
- `none` — вообще отменяет любые операции drag-n-drop;
- `move` — разрешает операцию переноса;
- `copy` — копирования;
- `link` — создания ссылки или ярлыка;
- `copyLink` — копирования и создания ссылки или ярлыка;
- `copyMove` — копирования и перемещения;
- `linkMove` — перемещения и создания ссылки или ярлыка;
- `all` — разрешает все.

Поддерживается IE начиная с 5.0.

enabledPlugin

Возвращает ссылку на объект `plugin`, поддерживающий текущий тип данных MIME. Если такого нет, возвращается `null`.

```
{Объект типа MIME}.enabledPlugin;
```

Доступно только для чтения.

Поддерживается N начиная с 3.0.

encoding

Задает или возвращает метод кодирования данных, отправляемых формой, в виде MIME.

```
{Объект формы}.encoding[ = "{Метод кодирования данных}"];
```

Значение по умолчанию application/x-www-form-urlencoded. Может также использоваться значение multipart/form-data, если в форме используется элемент <INPUT TYPE="file">.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

event

Задает или возвращает событие, к которому привязан скрипт-обработчик.

```
{Объект формы}.event[ = "{Тип события}"];
```

Этот способ создания обработчиков событий специфичен для Internet Explorer.

Поддерживается IE начиная с 3.02.

expando

Задает или возвращает состояние объектной модели Web-обозревателя: можно или нельзя создавать произвольные свойства объектов.

```
document.expando[ = true|false];
```

По умолчанию объектная модель Web-обозревателя разрешает создавать у объектов произвольные свойства (значение true), например, для хранения каких-либо значений продолжительное время. Вы можете запретить это, присвоив данному свойству значение false.

Поддерживается IE начиная с 4.0.

face

Задает или возвращает имя шрифта.

```
{Объект}.face[ = "{Имя шрифта}"];
```

Поддерживается IE начиная с 3.02 для <FONT> и начиная с 4.0 для <BASEFONT>.

fgColor

Задает или возвращает цвет текста.

```
{Объект}.fgColor[ = "{Цвет}"];
```

Значение по умолчанию #000000.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

fileCreatedDate

Возвращает дату создания файла HTML-документа или графического изображения.

```
{Объект}.fileCreatedDate;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

filename

Возвращает имя файла расширения Web-обозревателя.

```
{Объект}.filename;
```

Доступно только для чтения.

Поддерживается N начиная с 3.0.

fileModifiedDate

Возвращает дату последнего изменения файла HTML-документа или графического изображения.

```
{Объект}.fileModifiedDate;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

fileSize

Возвращает размер файла HTML-документа или графического изображения.

```
{Объект}.fileSize;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

fileUpdatedDate

Возвращает дату последнего обновления файла графического изображения.

```
{Объект}.fileUpdatedDate;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

filter

Задает или возвращает набор визуальных фильтров и преобразований, применяемых к элементу страницы.

```
{Объект стиля}.filter[ = "{Список фильтров и преобразований
↳с параметрами, разделенных пробелами}"];
```

Поддерживается IE начиная с 4.0.

firstChild

Возвращает ссылку на первый элемент с коллекции `childNodes`.

```
{Объект}.firstChild;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.0.

font

Задает или возвращает параметры шрифта элемента страницы. Заменяет свойства `fontFamily`, `fontHeight`, `fontSize`, `fontStyle`, `fontVariant` и `fontWeight`. Значения этих свойств могут располагаться в любом порядке.

```
{Объект стиля}.font[ = "{fontFamily} [{fontHeight}] [{fontSize}]
↳[{fontStyle}] [{fontVariant}] [{fontWeight}]"];
```

Значение по умолчанию `normal normal normal medium normal "Times New Roman"`.

Альтернативный формат.

```
{Объект стиля}.font[ = "caption|icon|menu|message-box|small-caption|
↳status-bar"];
```

В этом случае доступны шесть предопределенных значений, задающих один из стандартных шрифтов, используемых в элементах интерфейса Windows:

- `caption` — шрифт заголовков кнопок, текстовых меток и т. п.;
- `icon` — шрифт подписей под пиктограммами;
- `menu` — шрифт пунктов меню;
- `message-box` — шрифт содержимого стандартных окон-предупреждений;
- `small-caption` — мелкий шрифт заголовков;
- `status-bar` — шрифт содержимого строки состояния.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

fontFamily

Задает или возвращает имя шрифта, используемого в элементе страницы.

```
{Объект стиля}.fontFamily[ = "{Имя шрифта}|serif|san-serif|cursive|
↳fantasy|monospace"];
```

В качестве значения этого свойства задается либо непосредственно имя нужного шрифта, либо одно из пяти предопределенных значений, задающих имя шрифтового семейства. Можно задавать одновременно несколько шрифтов, разделив их имена запятыми; в этом случае Web-обозреватель сможет выбрать из них тот, который установлен на компьютере клиента. Если имя шрифта содержит пробелы, его следует взять в одинарные кавычки. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 3.02 для текстовых элементов и начиная с 4.0 для нетекстовых. Поддерживается N начиная с 4.0.

fontSize

Задает или возвращает имя размера шрифта, используемого в элементе страницы.

```
font-size: xx-small|x-small|small|medium|large|x-large|xx-large|
↳larger|smaller|{Абсолютный размер}|{Относительный размер}%
```

Возможно задание либо абсолютного размера шрифта в одной из поддерживаемых CSS единиц измерения, либо как процент от размера шрифта родителя. Также доступны девять предопределенных значений. Значения `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large` и `xx-large` задают один из семи размеров шрифтов, поддерживаемых HTML. Значения `smaller` и `larger` задают размер шрифта относительно шрифта родителя — на размер меньше или больше соответственно. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 3.02 для текстовых элементов и начиная с 4.0 для нетекстовых. Поддерживается N начиная с 4.0.

fontSmoothingEnabled

Возвращает `true`, если пользователем включено сглаживание шрифтов при отображении их на экране, и `false` — в противном случае.

```
screen.fontSmoothingEnabled;
```

Доступно только для чтения.

Поддерживается IE начиная 4.0.

fontStyle

Задает или возвращает вид шрифта, используемого в элементе страницы.

```
{Объект стиля}.fontStyle[ = "normal|italic|oblique"];
```

Доступны три значения:

- `normal` (значение по умолчанию) — задает обычный вид шрифта;
- `italic` — задает курсивное начертание;
- `oblique` — задает наклонное начертание (IE отображает его как курсив).

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 3.02 для текстовых элементов и начиная с 4.0 для нетекстовых. Поддерживается N начиная с 4.0.

fontVariant

Задает или возвращает вид малых букв шрифта, используемого в элементе страницы.

```
{Объект стиля}.fontVariant[ = "normal|small-caps"];
```

Доступны два значения:

- `normal` (значение по умолчанию) — задает обычный вид малых букв шрифта;
- `small-caps` — делает их такими же, как большие буквы, только меньшего размера (капитель).

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

fontWeight

Задает или возвращает "жирность" шрифта, используемого в элементе страницы.

```
{Объект стиля}.fontWeight[ = "normal|bold|bolder|lighter|100|200|300|400|
↳500|600|700|800|900"];
```

"Жирность" может быть задана тремя способами. Во-первых, предопределенными значениями `normal` и `bold`, задающими обычное и жирное начер-

тание соответственно. Во-вторых, относительными значениями `bolder` и `lighter`, делающими шрифт элемента страницы жирнее и светлее шрифта родителя. И, в-третьих, одним из девяти значений от `100` до `900`; здесь нормальному начертанию соответствует значение `400`, а жирному — `700`. Значение по умолчанию `normal`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 3.02 для текстовых элементов и начиная с 4.0 для нетекстовых. При этом IE 4.0 и более старые версии поддерживали только значения `normal` и `bold` этого атрибута. Поддерживается N начиная с 4.0.

form

Возвращает ссылку на форму, к которой привязан элемент управления.

```
{Объект элемента управления}.form;
```

Доступно только для чтения.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

frame

Задает или возвращает значение, предписывающее отобразить или скрыть внешние границы таблицы.

```
{Объект таблицы}.frame[ = "void|above|below|border|hsides|lhs|rhs|vsides|
↳box"];
```

Доступны девять значений:

- `void` — внешней границы нет вообще (значение по умолчанию);
- `above` — рисуется только верхняя линия внешней границы;
- `below` — рисуется только нижняя линия внешней границы;
- `border` — рисуются все линии внешней границы;
- `hsides` — рисуются только горизонтальные линии внешней границы, т. е. верхняя и нижняя;
- `lhs` — рисуется только левая линия внешней границы;
- `rhs` — рисуется только правая линия внешней границы;
- `vsides` — рисуются только вертикальные линии внешней границы, т. е. левая и правая;
- `box` — рисуются все линии внешней границы.

Поддерживается IE начиная с 3.02.

frameBorder (фреймы)

Задает или возвращает признак отображения границ между фреймами.

```
{Объект фрейма}.frameBorder[ = "1|0|yes|no"];
```

По умолчанию границы между фреймами отображаются (значения `1` или `yes`). Значения `0` или `no` отключают отображение границ.

Поддерживается IE начиная с 3.02.

frameElement

Возвращает ссылку на объект фрейма.

```
window.frameElement;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.5.

frameSpacing

Задает или возвращает интервал между фреймами в пикселах.

```
{Объект фрейма}.frameSpacing[ = "{Интервал между фреймами}"];
```

Значение по умолчанию 2.

Поддерживается IE начиная с 3.02.

fromElement

Задает или возвращает ссылку на элемент управления, с которого "ушел" курсор мыши. Имеет смысл только при возникновении событий `onmouseover` и `onmouseout`.

```
event.fromElement[ = {Элемент страницы}];
```

В IE 4.x доступно только для чтения.

Поддерживается IE начиная с 4.0.

hash

Задает или возвращает часть интернет-адреса текущей Web-страницы или гиперссылки, следующей за символом "решетки" #. Обычно это имя "якоря".

```
location.hash[ = "{Часть адреса, следующая за #}"];
```

Поддерживается IE начиная с 3.02 для `<A>` и `location` и начиная с 4.0 для `<AREA>`. Поддерживается N начиная с 2.0.

hasLayout

Возвращает `true`, если элемент страницы абсолютно позиционирован, и его ширина и/или высота заданы. В противном случае возвращается `false`.

```
{Объект}.hasLayout;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.5.

height

Задает или возвращает высоту элемента страницы в пикселах или как процент от доступного пространства.

```
{Объект}.height[ = "{Высота}"];
```

В N доступно только для чтения.

Поддерживается IE начиная с 3.02 для `<EMBED>`, `<IMG>`, `<MARQUEE>`, `<OBJECT>`, `<TD>` и `<TH>`, начиная с 4.0 для `<TABLE>`, начиная с 5.0 для `<TR>` и начиная с 5.5 для `<FRAME>`. Поддерживается N начиная с 3.0.

height (document)

Задает или возвращает высоту документа в пикселах.

```
document.height;
```

Поддерживается N начиная с 4.0.

height (event)

Задает или возвращает высоту окна или фрейма в пикселах.

```
event.height[ = {Y}];
```

Поддерживается N начиная с 4.0.

height (screen)

Возвращает полную высоту экрана компьютера в пикселах.

```
screen.height;
```

Доступно только для чтения.

Поддерживается IE и N начиная с 4.0.

height (CSS)

Задает или возвращает высоту свободно позиционированного элемента.

```
{Объект стиля}.height[ = "auto|{Y}|{Y}%"];
```

Высота может быть задана как абсолютной величиной, так и процентом от высоты родителя. Предопределенное значение `auto` заставляет Web-обозреватель устанавливать высоту элемента самостоятельно. Значение по умолчанию `auto`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

hidden

Задает или возвращает признак внедренного элемента `<EMBED>`: будет ли тот виден на Web-странице.

```
{Внедренный объект}.hidden[ = true|false];
```

По умолчанию внедренный элемент `<EMBED>` видим (значение `false`). Чтобы скрыть его, присвойте этому свойству значение `true`.

Поддерживается IE начиная с 3.02.

hideFocus

Задает или возвращает признак: показывать фокус ввода (текстовый курсор) в элементе страницы или нет.

```
{Объект}.hideFocus[ = true|false];
```

По умолчанию фокус ввода всегда показывается (значение `false`). Чтобы скрыть его, присвойте этому свойству значение `true`.

Поддерживается IE начиная с 5.5.

host

Задает или возвращает доменное имя сервера и номер порта, являющиеся частью интернет-адреса текущей Web-страницы или гиперссылки.

```
location.host[ = "{Доменное имя и номер порта}"];
```

Поддерживается IE начиная с 3.02 для `<A>` и `location` и начиная с 4.0 для `<AREA>`. Поддерживается N начиная с 2.0.

hostname

Задает или возвращает доменное имя сервера, являющееся частью интернет-адреса текущей Web-страницы или гиперссылки.

```
location.hostname[ = "{Доменное имя}"];
```

Поддерживается IE начиная с 3.02 для <A> и location и начиная с 4.0 для <AREA>. Поддерживается N начиная с 2.0.

href (<BASE>)

Задает или возвращает базовый интернет-адрес в теге <BASE>, относительно которого отсчитываются все ссылки.

```
{Объект}.href[ = "{Базовый интернет-адрес}"];
```

Поддерживается IE начиная с 3.02.

href (location)

Задает или возвращает полный интернет-адрес текущей Web-страницы.

```
location.host[ = "{Полный интернет-адрес}"];
```

Поддерживается IE начиная с 3.02.

href (styleSheet)

Задает или возвращает интернет-адрес файла таблицы стилей.

```
{Объект таблицы стилей}.href[ = "{Интернет-адрес файла таблицы стилей}"];
```

Поддерживается IE начиная с 4.0.

href (гиперссылки)

Задает или возвращает интернет-адрес назначения, куда пользователь попадет после активизации гиперссылки.

```
{Объект}.href[ = "{Интернет-адрес назначения}"];
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

hspace

Задает или возвращает расстояние по горизонтали от элемента страницы до окружающего его текста.

```
{Объект}.hspace[ = "{Горизонтальный отступ}"];
```

Значение по умолчанию 0.

В N доступно только для чтения.

Поддерживается IE начиная с 3.02 для <APPLET>, <IMG>, <INPUT TYPE="image">, <MARQUEE> и <OBJECT> и начиная с 4.0 для <IFRAME>. Поддерживается N начиная с 3.0.

htmlFor (<LABEL>)

Задает или возвращает элемент управления, к которому привязана метка <LABEL>.

```
{Объект метки}.htmlFor[ = "{Имя элемента управления}"];
```

Поддерживается IE начиная с 4.0.

htmlFor (<SCRIPT>)

Задает или возвращает элемент страницы, к которому привязан скрипт.

```
{Объект скрипта}.htmlFor[ = "{Имя элемента страницы}"];
```

Этот способ создания обработчиков событий специфичен для IE.

Поддерживается IE начиная с 3.02.

htmlText

Возвращает HTML-код текстового фрагмента TextRange.

```
{Объект}.htmlText;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

httpEquiv

Задает или возвращает тип метаданных, задаваемых в теге <META>.

```
{Объект метатега}.httpEquiv[ = "description|refresh|url|mimetype|
↳charset"];
```

Доступны пять предопределенных значений:

- description — задает описание Web-страницы;
- refresh — задает интервал обновления Web-страницы, отображенной Web-обозревателем, в секундах;
- url — задает адрес, откуда будет загружаться Web-страница при обновлении. Требует наличия тега <META>, задающего интервал обновления (refresh);
- mimetype — задает тип данных MIME Web-страницы (всегда text/html);
- charset — задает кодовую страницу, которая использовалась при написании текста.

Также могут поддерживаться другие, нестандартные типы метаданных.

Поддерживается IE начиная с 3.02.

id

Возвращает уникальное имя элемента страницы, заданное атрибутом `ID`.

```
{Объект}.id[ = "{Имя элемента страницы}"];
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

imeMode

Задает или возвращает состояние IME (Input Method Editor — редактор способа ввода), с помощью которого вводятся иероглифические тексты на китайском, корейском и японском языках.

```
{Объект поля ввода}.style.imeMode[ = "auto|active|inactive|disabled"];
```

Доступны четыре значения:

- `auto` (значение по умолчанию) — передает управление IME Web-обозревателю;
- `active` — активизирует IME. Пользователь может его деактивизировать;
- `inactive` — деактивизирует IME. Пользователь может его активизировать;
- `disabled` — отключает IME.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.0.

indeterminate

Задает или возвращает признак: находится ли флажок в неопределенном состоянии. Неопределенным состоянием флажка называется такое состояние, когда он закрашен серым. При этом пользователь все еще может щелкать по флажку, меняя тем самым его состояние.

```
{Объект флажка}.indeterminate[ = true|false];
```

По умолчанию флажок находится в обычном — включенном либо выключенном — состоянии (значение `false`). Чтобы ввести его в неопределенное состояние, присвойте этому свойству значение `true`.

Поддерживается IE начиная с 4.0.

index

Задает или возвращает порядковую позицию пункта списка <OPTION> в списке выбора <SELECT>.

```
{Объект позиции списка}.index[ = {Номер позиции}];
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

innerHeight

Задает или возвращает высоту клиентской области окна Web-обозревателя.

```
window.innerHeight[ = {Y}];
```

Поддерживается N начиная с 4.0.

innerHTML

Задает или возвращает фрагмент HTML-кода, заключенного между открывающим и закрывающим тегами текущего элемента страницы.

```
{Объект}.innerHTML[ = "{HTML-код}"];
```

Доступно для чтения и записи, для тегов <COL>, <COLGROUP>, <FRAMESET>, <HEAD>, <HTML>, <STYLE>, <TABLE>, <TBODY>, <TFOOT>, <THEAD>, <TITLE> и <TR> — только для чтения.

Поддерживается IE начиная с 4.0.

innerText

Задает или возвращает фрагмент текста, заключенного между открывающим и закрывающим тегами текущего элемента страницы, без HTML-тегов. При присвоении этому свойству некоего текста все HTML-теги также игнорируются.

```
{Объект}.innerText[ = "{Текст}"];
```

Доступно для чтения и записи, для тегов <HTML>, <TABLE>, <TBODY>, <TFOOT>, <THEAD> и <TR> — только для чтения.

Поддерживается IE начиная с 4.0.

innerWidth

Задает или возвращает ширину клиентской области окна Web-обозревателя.

```
window.innerWidth[ = {X}];
```

Поддерживается N начиная с 4.0.

isContentEditable

Возвращает `true`, если пользователь может редактировать содержимое Web-страницы или ее элемента, и `false`, если не может.

```
{Объект}.isContentEditable;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.5.

isDisabled

Возвращает `true`, если пользователь может взаимодействовать с элементом страницы, и `false`, если не может.

```
{Объект}.isDisabled;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.5.

isMap

Задает или возвращает признак рисунка `<IMG>`: будет ли он вести себя как карта-изображение.

```
{Объект рисунка}.isMap[ = true|false];
```

По умолчанию рисунок не является картой-изображением (значение `false`). Чтобы сделать его таковым, присвойте этому свойству значение `true`.

Поддерживается IE начиная с 3.02.

isOpen

Возвращает `true`, если всплывающее окно открыто, и `false`, если закрыто.

```
{Объект окна}.isOpen;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.5.

isTextEdit

Возвращает `true`, если объект `TextRange` может быть создан с использованием текущего элемента страницы, и `false`, если не может быть создан.

```
{Объект}.isTextEdit;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

keyCode

Задает или возвращает в формате Unicode код символа, связанного с нажатой клавишей. Если ни одна клавиша не была нажата, возвращается 0.

```
event.keyCode[ = {Код символа Unicode}"];
```

Поддерживается IE начиная с 4.0.

lang

Задает или возвращает язык, на котором набран текст.

```
{Объект}.lang[ = "{Код языка}"];
```

Список доступных кодов языка приведен в приложении 1.

Поддерживается IE начиная с 4.0.

language

Задает или возвращает название языка программирования, на котором написан скрипт, привязанный к элементу страницы.

```
{Объект}.language[ = "JScript|javascript|vbs|vbscript|XML|{Код языка}"];
```

Доступны пять предопределенных значений:

- JScript;
- javascript;
- vbs;
- vbscript;
- XML.

Могут поддерживаться и другие, нестандартные языки.

Поддерживается IE начиная с 4.0.

language (navigator)

Возвращает код языка программы Web-обозревателя.

```
navigator.language;
```

Доступно только для чтения.

Поддерживается N начиная с 4.0.

lastChild

Возвращает ссылку на последний элемент коллекции `childNodes`.

```
{Объект}.lastChild;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.0.

lastModified

Возвращает дату последнего изменения файла Web-страницы.

```
document.lastModified;
```

Доступно только для чтения.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

layerX

Задает или возвращает либо ширину элемента страницы (при наступлении события изменения размера), либо горизонтальную координату курсора мыши относительно слоя в пикселах.

```
event.layerX[ = {X}];
```

Поддерживается N начиная с 4.0.

layerY

Задает или возвращает либо высоту элемента страницы (при наступлении события изменения размера), либо вертикальную координату курсора мыши относительно слоя в пикселах.

```
event.layerY[ = {Y}];
```

Поддерживается N начиная с 4.0.

layoutFlow

Задает или возвращает направление написания текста: по горизонтали или по вертикали.

```
{Объект стиля}.layoutFlow[ = "horizontal|vertical-ideographic"];
```

Доступны два значения:

- `horizontal` (значение по умолчанию) — задает горизонтальное направление написания текста;
- `vertical-ideographic` — вертикальное направление написания текста.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.5. В настоящее время признано устаревшим; вместо него рекомендуется использовать свойство `writingMode`.

layoutGrid

Задает или возвращает параметры разметки элемента страницы, используемой для вывода иероглифических текстов на китайском, корейском и японском языках. Заменяет свойства layoutGridChar, layoutGridLine, layoutGridMode и layoutGridType. Значения этих свойств могут располагаться в любом порядке.

```
{Объект стиля}.layoutGrid[ = ["{layoutGridChar}] [{layoutGridLine}]
↳[{layoutGridMode}] [{layoutGridType}]"];
```

Значение по умолчанию both loose none none.

Поддерживается IE начиная с 5.0.

layoutGridChar

Задает или возвращает шаг горизонтальной разметки элемента страницы, используемой для вывода иероглифических текстов на китайском, корейском и японском языках.

```
{Объект стиля}.layoutGridChar[ = "none|auto|{Y}|{Y}%"];
```

Шаг разметки может быть задан как абсолютной величиной, так и процентом от шага разметки родителя. Предопределенное значение auto заставляет Web-обозреватель устанавливать шаг разметки по максимальному символу текста. Другое предопределенное значение none вообще отключает разметку. Значение по умолчанию none. Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 5.0.

layoutGridLine

Задает или возвращает шаг вертикальной разметки элемента страницы, используемой для вывода иероглифических текстов на китайском, корейском и японском языках.

```
{Объект стиля}.layoutGridLine[ = "none|auto|{Y}|{Y}%"];
```

Шаг разметки может быть задан как абсолютной величиной, так и процентом от шага разметки родителя. Предопределенное значение auto заставляет Web-обозреватель устанавливать шаг разметки по максимальному символу текста. Другое предопределенное значение none вообще отключает разметку. Значение по умолчанию none. Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 5.0.

layoutGridMode

Задает или возвращает тип разметки элемента страницы, используемой для вывода иероглифических текстов на китайском, корейском и японском языках.

```
{Объект стиля}.layoutGridMode[ = "both|none|char|line"];
```

Доступны четыре значения:

- `both` (значение по умолчанию) — задает использование и горизонтальной, и вертикальной разметок;
- `none` — отключает разметку;
- `char` — задает использование горизонтальной разметки;
- `line` — задает использование вертикальной разметки.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.0.

layoutGridType

Задает или возвращает режим форматирования текстов на китайском, корейском и японском языках с использованием разметки элемента страницы.

```
{Объект стиля}.layoutGridType[ = "loose|strict|fixed"];
```

Доступны три значения:

- `loose` (значение по умолчанию) — задает "гибкое" форматирование, используемое для корейских и японских текстов;
- `strict` — задает более "строгое" форматирование, используемое для китайских, корейских и японских текстов;
- `fixed` — задает самое "строгое" форматирование, когда символы жестко привязываются к разметке.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.0.

left (TextRectangle)

Задает или возвращает горизонтальную координату левой границы объекта `TextRectangle` в пикселах.

```
{Объект TextRectangle}.left[ = {X}];
```

Поддерживается IE начиная с 5.0.

left (слой)

Задает или возвращает горизонтальную координату левой границы слоя в пикселах.

```
{Объект слоя}.left[ = {X}];
```

Поддерживается N начиная с 4.0.

left (CSS)

Задает или возвращает горизонтальную позицию левой границы свободно позиционированного элемента относительно родителя.

```
{Объект стиля}.left[ = "auto|{X}|{X}%"];
```

Координата может быть задана как абсолютной величиной, так и процентом от ширины родителя. Предопределенное значение `auto` заставляет Web-обозреватель позиционировать элемент самостоятельно. Значение по умолчанию `auto`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

leftMargin

Задает или возвращает расстояние от левой границы окна Web-обозревателя до левой границы содержимого Web-страницы в пикселах.

```
body.leftMargin[ = "{Расстояние}"];
```

Значение по умолчанию 10.

Поддерживается IE начиная с 4.0.

length (<FORM>)

Возвращает количество элементов управления в форме.

```
{Объект формы}.length;
```

Доступно только для чтения.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

length (history)

Возвращает количество позиций в списке "истории" Web-обозревателя.

```
history.length;
```

Доступно только для чтения.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

length (<OPTION>)

Возвращает количество пунктов <OPTION> в списке <SELECT>.

```
{Объект пункта списка}.length;
```

Доступно только для чтения.

Поддерживается N начиная с 2.0.

length (plugin)

Возвращает количество элементов коллекции navigator.plugins.

```
{Объект расширения Web-обозревателя}.length;
```

Доступно только для чтения.

Поддерживается N начиная с 3.0.

length (<SELECT>)

Возвращает количество пунктов <OPTION> в списке <SELECT>.

```
{Объект списка}.length;
```

Доступно только для чтения.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

length (TextNode)

Возвращает длину содержимого объекта TextNode в символах текста.

```
{Объект TextNode}.length;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.0.

length (window)

Возвращает количество фреймов в окне Web-обозревателя.

```
wiindow.length;
```

Доступно только для чтения.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

length (коллекция)

Возвращает количество элементов коллекции.

```
{Коллекция}.length;
```

Доступно только для чтения.

Поддерживается IE начиная с 3.02.

letterSpacing

Задает или возвращает интервал, добавляемый между символами текста.

```
{Объект стиля}.letterSpacing[ = "normal|{Величина}"];
```

Значение этого свойства может быть задано либо абсолютной величиной в одной из поддерживаемых CSS единиц измерения, либо предопределенным значением `normal`, задающим стандартную величину расстояния между символами. Значение по умолчанию `normal`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

lineBreak

Задает или возвращает правила разрыва строк для текста на японском языке.

```
{Объект стиля}.lineBreak[ = "normal|strict"];
```

Доступны два значения: `normal` (значение по умолчанию), задающее обычные правила разрыва японского текста, и `strict`, задающее строгие правила. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.0.

lineHeight

Задает или возвращает расстояние по вертикали между строками текста.

```
{Объект стиля}.lineHeight[ = "normal|{Y}|{Y}%"];
```

Высота может быть задана как абсолютной величиной, так и процентом от высоты родителя. Предопределенное значение `normal` задает стандартное расстояние. Значение по умолчанию `normal`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE и N начиная с 4.0.

link

Задает или возвращает цвет гиперссылок в документе.

```
body.link[ = "{Цвет}"];
```

Поддерживается IE начиная с 3.02.

linkColor

Задает или возвращает цвет гиперссылок в документе.

```
document.linkColor[ = "{Цвет}"];
```

Значение по умолчанию `#0000FF`.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

listStyle

Задает или возвращает параметры маркера или номера позиции списка. Заменяет свойства `listStyleImage`, `listStylePosition` и `listStyleType`. Значения этих свойств могут располагаться в любом порядке.

```
{Объект стиля}.listStyle[ = "[{listStyleImage}] [{listStylePosition}]
↳[{listStyleType}]"];
```

Значение по умолчанию `disc outside none`.

Поддерживается IE начиная с 4.0.

listStyleImage

Задает или возвращает интернет-адрес файла графического изображения, отображаемого в качестве маркера позиции списка. Имеет приоритет над свойством `listStyleType`.

```
{Объект стиля}.listStyleImage[ = "none|url({Интернет-адрес файла
↳изображения})"];
```

Если задано предопределенное значение `none`, то стиль маркера берется из установок свойства `listStyleType`. Если задан интернет-адрес файла изображения, то это изображение отображается в качестве маркера, перекрывая установки свойства `listStyleType`. Значение по умолчанию `none`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

listStylePosition

Задает или возвращает местонахождение маркера позиции списка.

```
{Объект стиля}.listStylePosition[ = "outside|inside"];
```

Доступны два значения. Значение `outside` (по умолчанию) задает отображение маркера позиции списка вне текста позиции. Значение же `inside` заставляет Web-обозреватель отобразить маркер позиции в ее тексте в качестве

первого символа. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

listStyleType

Задает или возвращает тип маркера или номера позиции списка.

```
{Объект стиля}.listStyleType[ = "disc|circle|square|decimal|lower-roman|
↳upper-roman|lower-alpha|upper-alpha|none"];
```

Доступны девять значений:

- `disc` (значение по умолчанию) — отображает сплошной кружок;
- `circle` — окружность;
- `square` — сплошной квадрат;
- `decimal` нумерует позиции арабскими цифрами;
- `lower-roman` — малыми римскими;
- `upper-roman` — большими римскими;
- `lower-alpha` — помечает позиции малыми латинскими буквами;
- `upper-alpha` — большими латинскими;
- `none` — вообще убирает маркер или нумерацию.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE и N начиная с 4.0.

locationbar

Задает или возвращает состояние панели адреса: показана ли она на экране или скрыта.

```
window.locationbar[ = true|false];
```

Значение `true` показывает панель адреса на экране, значение `false` скрывает.

Поддерживается N начиная с 4.0.

loop (<BGSOUND>, <IMG>, <INPUT TYPE="image">)

Задает или возвращает количество повторений фонового звука или видеофильма.

```
{Объект}.loop[ = "-1|0|{Количество}"];
```

Значение -1 или 0 задает бесконечное повторение. Значение по умолчанию 1.

Поддерживается IE начиная с 3.02.

loop (<MARQUEE>)

Задает или возвращает количество "прокруток" текста в элементе <MARQUEE>.

```
{Объект}.loop[ = "-1|0|{Количество}"];
```

Значение -1 или 0 задает бесконечное повторение. Значение по умолчанию -1.

Поддерживается IE начиная с 3.02.

lowsrc

Задает или возвращает интернет-адрес файла с изображением низкого разрешения.

```
{Объект изображения}.lowsrc[ = "{Интернет-адрес файла изображения}"];
```

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

margin

Задает или возвращает значения ширины полей между элементом страницы и его соседями. Заменяет свойства marginTop, marginRight, marginBottom и marginLeft.

```
{Объект стиля}.margin[ = "{marginTop} [{marginRight}] [{marginBottom}]
↳[{marginLeft}]"];
```

Может быть задано от одного до четырех значений. Если задано одно значение, оно применяется ко всем четырем границам. Если задано два значения, первое относится к верхнему и нижнему полю, а второе — к левому и правому. Если задано три значения, то первое применяется к верхнему полю, второе — к левому и правому, третье — к нижнему. Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 3.02 для текстовых элементов и начиная с 4.0 для нетекстовых.

marginBottom

Задает или возвращает нижнее поле между элементом страницы и его соседями снизу.

```
{Объект стиля}.marginBottom[ = "auto|{Y}|{Y}%"];
```

Координата может быть задана как абсолютной величиной, так и процентом от высоты родителя. Предопределенное значение `auto` заставляет Web-обозреватель устанавливать поле самостоятельно. При этом если задано свойство `marginTop`, значение нижнего поля устанавливается равным значению верхнего поля. Значение по умолчанию `auto`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 3.02 для текстовых элементов и начиная с 4.0 для нетекстовых. Поддерживается N начиная с 4.0.

marginHeight

Задает или возвращает отступ по вертикали между границей фрейма и его содержимым.

```
{Объект фрейма}.marginHeight[ = "{Отступ}"];
```

Поддерживается IE начиная с 3.02.

marginLeft

Задает или возвращает левое поле между элементом страницы и его соседями слева.

```
{Объект стиля}.marginLeft[ = "auto|{X}|{X}%"];
```

Координата может быть задана как абсолютной величиной, так и процентом от ширины родителя. Предопределенное значение `auto` заставляет Web-обозреватель устанавливать поле самостоятельно. При этом если задано свойство `marginRight`, значение левого поля устанавливается равным значению правого поля. Значение по умолчанию `auto`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 3.02 для текстовых элементов и начиная с 4.0 для нетекстовых. Поддерживается N начиная с 4.0.

marginRight

Задает или возвращает правое поле между элементом страницы и его соседями справа.

```
{Объект стиля}.marginRight[ = "auto|{X}|{X}%"];
```

Координата может быть задана как абсолютной величиной, так и процентом от ширины родителя. Предопределенное значение `auto` заставляет Web-обозреватель устанавливать поле самостоятельно. При этом если задано свойство `marginLeft`, значение правого поля устанавливается равным значению левого поля. Значение по умолчанию `auto`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 3.02 для текстовых элементов и начиная с 4.0 для нетекстовых. Поддерживается N начиная с 4.0.

marginTop

Задает или возвращает верхнее поле между элементом страницы и его соседями сверху.

```
{Объект стиля}.marginTop[ = "auto|{Y}|{Y}%"];
```

Координата может быть задана как абсолютной величиной, так и процентом от высоты родителя. Предопределенное значение `auto` заставляет Web-обозреватель устанавливать поле самостоятельно. При этом если задано свойство `marginBottom`, значение верхнего поля устанавливается равным значению нижнего поля. Значение по умолчанию `auto`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 3.02 для текстовых элементов и начиная с 4.0 для нетекстовых. Поддерживается N начиная с 4.0.

marginWidth

Задает или возвращает отступ по горизонтали между границей фрейма и его содержимым.

```
{Объект фрейма}.marginWidth[ = "{Отступ}"];
```

Поддерживается IE начиная с 3.02.

maxLength

Задает или возвращает максимальное количество символов, которое пользователь может ввести в поле ввода.

```
{Объект поля ввода}.maxLength[ = "{Максимальное количество символов}"];
```

Поддерживается IE начиная с 4.0.

media (document)

Задает или возвращает вид использования Web-страницы: только для просмотра на экране компьютера, только для печати на принтере или и для того, и для другого.

```
document.media[ = "print|{Любое другое значение}"];
```

Доступно одно-единственное предопределенное значение `print`, говорящее о том, что эта Web-страница предназначена и должна форматироваться для

печати на принтере. Любое другое значение задает использование Web-страницы для просмотра на экране.

Поддерживается IE начиная с 5.5.

media (таблицы стилей)

Задает или возвращает вид использования таблицы стилей: только для просмотра Web-страницы на экране компьютера, только для печати на принтере или и для того, и для другого.

```
{Объект таблицы стилей}.media[ = "screen|print|all"];
```

Доступны три значения:

- `screen` — заставляет Web-обозреватель использовать таблицу стилей только для просмотра Web-страницы на экране;
- `print` — только для печати на принтере;
- `all` — и для просмотра, и для печати (значение по умолчанию).

Поддерживается IE начиная с 4.0 для `<LINK>` и начиная с 5.0 для `<STYLE>`.

menuArguments

Возвращает ссылку на окно, в котором было открыто контекстное меню.

```
external.menuArguments;
```

Это свойство можно использовать в скриптах, зарегистрированных в Реестре Windows для новых пунктов контекстного меню Web-страницы. Запросив его, программист получит ссылку на окно, где была открыта Web-страница, на элементе которой пользователь щелкнул правой клавишей мыши. Свойство доступно только для чтения.

Поддерживается IE начиная с 4.0.

menubar

Задает или возвращает состояние строки меню: показана ли она на экране или скрыта.

```
window.menubar[ = true|false];
```

Значение `true` показывает строку меню на экране, значение `false` скрывает.

Поддерживается N начиная с 4.0.

method

Задает или возвращает метод передачи данных формой: get или post.

```
{Объект формы}.method[ = "get|post"];
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

Methods

Задает или возвращает список методов протокола HTTP, поддерживаемых гиперссылкой.

```
{Объект гиперссылки}.Methods[ = "{Список методов HTTP, разделенных
↪запятыми}"];
```

Поддерживается IE начиная с 4.0.

modifiers

Задает или возвращает строку, содержащую список модификаторов. Эти модификаторы обозначают нажатие одной из функциональных клавиш: <Alt>, <Ctrl> и <Shift>.

```
event.modifiers[ = "{Список модификаторов}"];
```

Всего модификаторов три: ALT_MASK, CONTROL_MASK и SHIFT_MASK.

Поддерживается N начиная с 4.0.

multiple

Задает или возвращает признак списка <SELECT>: может ли пользователь выбрать в нем несколько значений.

```
{Объект списка}.multiple[ = true|false];
```

По умолчанию в списке может быть выбрано только один элемент (значение false). Чтобы дать возможность пользователю выбирать одновременно несколько элементов списка, дайте этому свойству значение true.

Поддерживается IE начиная с 3.02.

name (<A>)

Задает или возвращает имя "якоря".

```
{Объект гиперссылки}.name[ = "{Имя "якоря"}"];
```

Поддерживается IE начиная с 3.02 и N начиная с 4.0.

name (<IMG>, элементы управления, внедренные объекты)

Задает или возвращает имя элемента страницы, используемое в скриптах.

```
{Объект}.name[ = "{Имя}"];
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

name (<META>)

Задает или возвращает тип метаданных в теге <META>. Сами метаданные передаются значением свойства content.

```
{Объект метаданных}.name[ = "Author|Description|Generator|Keywords|
↳ProgID|Robots|Template|{Тип}"];
```

Для IE доступны семь предопределенных значений:

- Author — сообщает информацию об авторе Web-страницы;
- Description — обозначает описание Web-страницы;
- Generator — задает имя приложения, создавшего Web-страницу;
- Keywords — задает список ключевых слов, разделенных запятыми. Эти ключевые слова будут использоваться поисковыми машинами;
- ProgID — идентифицирует программу для редактирования Web-страниц;
- Robots — разрешает или запрещает индексацию Web-страницы поисковыми машинами. Доступны два значения атрибута CONTENT: all и noindex, разрешающее и запрещающее индексацию соответственно;
- Template — задает имя файла шаблона, использовавшегося для создания Web-страницы. Используется вместе с ProgID.

Также допускаются любые другие, не предопределенные типы метаданных.

Поддерживается IE начиная с 3.02.

name (namespace)

Возвращает имя пространства имен namespace.

```
namespace.name;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.5.

name (plugin)

Возвращает название расширения Web-обозревателя.

```
{Объект}.name;
```

Доступно только для чтения.

Поддерживается N начиная с 3.0.

name (window и фреймы)

Задает или возвращает имя окна или фрейма.

```
{Объект}.name[ = "{Имя фрейма}"];
```

Поддерживается IE начиная с 3.02. Поддерживается N начиная с 2.0, начиная с 3.0 поддерживает чтение и запись.

nameProp

Возвращает имя файла, на который ссылается гиперссылка или тег <IMG>.

```
{Объект}.nameProp;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.0.

next

Возвращает интернет-адрес следующей позиции в списке "истории".

```
history.next;
```

Доступно только для чтения.

Поддерживается N начиная с 3.0.

nextPage

Возвращает позицию, которую займет следующая печатаемая на принтере страница Web-документа. Применяется только в шаблонах печати.

```
event.nextPage;
```

Возвращается одно из трех значений:

- "left" — говорит о том, что новая страница будет четной ("левая");
- "right" — нечетной ("правая");
- пустая строка "" — правило @page не было задано.

Доступно только для чтения.

Поддерживается IE начиная с 5.5.

nextSibling

Возвращает ссылку на следующего потомка родителя текущего элемента страницы.

```
{Объект}.nextSibling;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.0.

nodeName

Возвращает имя тега, атрибута или объекта `TextNode`.

```
{Объект}.nodeName;
```

Возвращает имя тега, если `{Объект}` — элемент страницы, имя атрибута, если `{Объект}` — атрибут страницы (объект `Attribute`), или `#text`, если `{Объект}` — объект `TextNode`. Свойство доступно только для чтения.

Поддерживается IE начиная с 5.0.

nodeType

Возвращает тип объекта: тег, атрибут или объект `TextNode`.

```
{Объект}.nodeType;
```

Возвращает `1`, если `{Объект}` — элемент страницы, `null`, если `{Объект}` — атрибут страницы (объект `Attribute`), или `3`, если `{Объект}` — объект `TextNode`. Свойство доступно только для чтения.

Поддерживается IE начиная с 5.0.

nodeValue

Возвращает значение атрибута или объекта `TextNode`.

```
{Объект}.nodeValue;
```

Возвращает `null`, если `{Объект}` — элемент страницы; значение атрибута, если `{Объект}` — атрибут страницы (объект `Attribute`), или текстовое содержимое, если `{Объект}` — объект `TextNode`. Свойство доступно только для чтения.

Поддерживается IE начиная с 5.0.

noHref

Задает или возвращает признак области на карте-изображении: является ли она "пустой", не связанной ни с одной ссылкой.

```
{Объект области}.noHref[ = true|false];
```

По умолчанию область карты-изображения "горяча", т. е. привязана к какой-то гиперссылке (значение `false`). Чтобы сделать "пустую" область, дайте этому свойству значение `true`.

Поддерживается IE начиная с 3.02.

noResize

Задает или возвращает признак фрейма: может ли пользователь изменить его размер.

```
{Объект фрейма}.noResize[ = true|false];
```

По умолчанию пользователь может изменять размер фрейма, перетаскивая мышью его границу (значение `false`). Чтобы запретить эту возможность, дайте этому свойству значение `true`.

Поддерживается IE начиная с 4.0.

noShade

Задает или возвращает признак горизонтальной линейки <HR>: будет ли она иметь тень.

```
{Объект горизонтальной линейки}.noShade[ = true|false];
```

По умолчанию горизонтальная линейка имеет трехмерный вид (значение `false`). Чтобы сделать ее обычной, "плоской", дайте этому свойству значение `true`.

Поддерживается IE начиная с 3.02.

noWrap

Задает или возвращает признак элемента страницы: будет ли находящийся в нем текст автоматически разбиваться на строки, если элемент страницы слишком узок, чтобы вывести текст в одну строку.

```
{Объект}.noWrap[ = true|false];
```

По умолчанию слова в тексте переносятся на новую строку автоматически, если ширина элемента страницы недостаточна для отображения строки целиком (значение `false`). Чтобы отключить автоматический перенос, дайте этому свойству значение `true`.

Поддерживается IE начиная с 3.02 для <TD> и <TH> и начиная с 4.0 для <BODY>, <DD>, <DIV> и <DT>.

object

Возвращает ссылку на внедренный объект тега <OBJECT>.

```
{Внедренный объект}.object;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

offscreenBuffering

Задает или возвращает признак окна Web-обозревателя: будет ли при отображении страницы использоваться внеэкранная буферизация. При внеэкранной буферизации содержимое Web-страницы сначала отображается в памяти компьютера, а уже потом — выводится на экран.

```
window.offscreenBuffering[ = "auto|VARIANT_TRUE|VARIANT_FALSE"];
```

По умолчанию Web-обозреватель сам решает, использовать ли внеэкранную буферизацию или нет (значение auto). Чтобы включить ее, дайте этому свойству значение VARIANT_TRUE, чтобы выключить — значение VARIANT_FALSE.

Использование внеэкранной буферизации позволяет увеличить скорость вывода содержимого Web-страницы при прокрутке, но требует много оперативной памяти. Отключив внеэкранную буферизацию, можно сэкономить память компьютера для других задач.

Поддерживается IE и N начиная с 4.0.

offsetHeight

Возвращает высоту элемента страницы относительно системы координат родителя.

```
{Объект}.offsetHeight;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

offsetLeft

Возвращает горизонтальную координату левой границы элемента страницы относительно системы координат родителя.

```
{Объект}.offsetLeft;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

offsetParent

Возвращает ссылку на родителя элемента страницы, относительно системы координат которого вычисляются значения свойств `offsetHeight`, `offsetLeft`, `offsetTop` и `offsetWidth`. В большинстве случаев это элемент `<BODY>`.

```
{Объект}.offsetParent;
```

В IE 4.x для элемента `<TD>` возвращается ссылка на `<TR>`, в 5.x — на `<TABLE>`. Свойство доступно только для чтения.

Поддерживается IE начиная с 4.0.

offsetTop

Возвращает вертикальную координату верхней границы элемента страницы относительно системы координат родителя.

```
{Объект}.offsetTop;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

offsetWidth

Возвращает ширину элемента страницы относительно системы координат родителя.

```
{Объект}.offsetWidth;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

offsetX

Задает или возвращает горизонтальную координату курсора мыши относительно системы координаты элемента страницы, сгенерировавшего событие.

```
event.offsetX[ = "{X}"];
```

В IE 4.x это свойство было доступно только для чтения.

Поддерживается IE начиная с 4.0.

offsetY

Задает или возвращает вертикальную координату курсора мыши относительно системы координаты элемента страницы, сгенерировавшего событие.

```
event.offsetY[ = "{Y}"];
```

В IE 4.x это свойство было доступно только для чтения.

Поддерживается IE начиная с 4.0.

onLine

Возвращает `true`, если Web-обозреватель находится в состоянии "на линии" (on-line), т. е. подключен к Интернету, и `false`, если в состоянии "не на линии" (off-line).

```
navigator.onLine;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

opener

Задает или возвращает ссылку на окно Web-обозревателя, создавшее текущее окно.

```
window.opener[ = "{Окно}"];
```

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

outerHeight

Задает или возвращает полную высоту окна Web-обозревателя.

```
window.outerHeight[ = {Y}];
```

Поддерживается N начиная с 4.0.

outerHTML

Задает или возвращает фрагмент HTML-кода, заключенного между открывающим и закрывающим тегами текущего элемента страницы, включая и сами его теги.

```
{Объект}.outerHTML[ = "{HTML-код}"];
```

Доступно для чтения и записи, в случае тегов <CAPTION>, <COL>, <COLGROUP>, <FRAMESET>, <HEAD>, <HTML>, <STYLE>, <TBODY>, <TD>, <TFOOT>, <TH>, <THEAD> и <TR> — только для чтения.

Поддерживается IE начиная с 4.0.

outerText

Задает или возвращает фрагмент текста, заключенного между открывающим и закрывающим тегами текущего элемента страницы, включая и сами эти

теги. При этом все HTML-теги игнорируются, т. е. возвращается только собственно текст.

```
{Объект}.outerText[ = "{Текст}"];
```

Доступно для чтения и записи, в случае тегов <HTML>, <TBODY>, <TD>, <TFOOT>, <TH>, <THEAD> и <TR> — только для чтения.

Поддерживается IE начиная с 4.0.

outerWidth

Задает или возвращает полную ширину окна Web-обозревателя.

```
window.outerWidth[ = {X}];
```

Поддерживается N начиная с 4.0.

overflow

Задает или возвращает поведение элемента страницы, если его содержимое в нем не помещается.

```
{Объект стиля}.overflow[ = "visible|scroll|hidden|auto"];
```

Доступны четыре значения:

- visible — заставляет элемент страницы расшириться так, чтобы все его содержимое было видно (значение по умолчанию);
- scroll — заставляет Web-обозреватель отобразить в элементе страницы полосы прокрутки, пользуясь которыми можно прокручивать его содержимое;
- hidden — скрывает все то, что выходит за пределы элемента страницы;
- auto — аналогично scroll за тем исключением, что полосы прокрутки отображаются, только если они реально необходимы (значение по умолчанию для <TEXTAREA>).

Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 4.0.

overflowX

Задает или возвращает поведение элемента страницы, если его ширина меньше ширины его содержимого.

```
{Объект стиля}.overflowX[ = "visible|scroll|hidden|auto"];
```

Доступны четыре значения:

- `visible` — заставляет элемент страницы расшириться по горизонтали так, чтобы все его содержимое было видно (значение по умолчанию);
- `scroll` — заставляет Web-обозреватель отобразить в элементе страницы полосу прокрутки, пользуясь которой можно прокручивать его содержимое;
- `hidden` — скрывает все то, что выходит за пределы элемента страницы (значение по умолчанию для <TEXTAREA>);
- `auto` аналогично `scroll` за тем исключением, что полоса прокрутки отображается, только если она реально необходима.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

overflowY

Задает или возвращает поведение элемента страницы, если его высота меньше высоты его содержимого.

```
{Объект стиля}.overflowY[ = "visible|scroll|hidden|auto"];
```

Доступны четыре значения:

- `visible` — заставляет элемент страницы расшириться по вертикали так, чтобы все его содержимое было видно (значение по умолчанию);
- `scroll` — заставляет Web-обозреватель отобразить в элементе страницы полосу прокрутки, пользуясь которой, можно просматривать его содержимое;
- `hidden` — скрывает все то, что выходит за пределы элемента страницы;
- `auto` — аналогично `scroll` за тем исключением, что полоса прокрутки отображается, только если она реально необходима (значение по умолчанию для <TEXTAREA>).

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

owningElement

Возвращает ссылку на следующий объект в объектной иерархии. Как правило, это элемент <STYLE> или <LINK>, определяющий таблицу стилей.

```
{Объект таблицы стилей}.owningElement;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

padding

Задает или возвращает расстояния между элементом страницы и различными границами. Заменяет свойства paddingTop, paddingRight, paddingBottom и paddingLeft.

```
{Объект стиля}.padding[ = "{paddingTop} [{paddingRight}]
↳[{paddingBottom}] [{paddingLeft}]"];
```

Может быть задано от одного до четырех значений. Если задано одно значение, оно применяется ко всем четырем границам элемента. Если задано два значения, первое относится к верхней и нижней границе, а второе — к левой и правой. Если задано три значения, то первое применяется к верхней границе, второе — к левой и правой, третье — к нижней. Значение по умолчанию 0, для <TD> 1. Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 4.0.

paddingBottom

Задает или возвращает расстояние между элементом страницы и нижней границей.

```
{Объект стиля}.paddingBottom[ = "{Y}|{Y}%"];
```

Расстояние может быть задано как абсолютной величиной, так и процентом от высоты родителя. Значение по умолчанию 0, для тегов <TD> 1. Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE и N начиная с 4.0.

paddingLeft

Задает или возвращает расстояние между элементом страницы и левой границей.

```
{Объект стиля}.paddingLeft[ = "{X}|{X}%"];
```

Расстояние может быть задано как абсолютной величиной, так и процентом от ширины родителя. Значение по умолчанию 0, для тегов <TD> 1. Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE и N начиная с 4.0.

paddingRight

Задает или возвращает расстояние между элементом страницы и правой границей.

```
{Объект стиля}.paddingRight[ = "{X}|{X}%"];
```

Расстояние может быть задано как абсолютной величиной, так и процентом от ширины родителя. Значение по умолчанию `0`, для тегов `<TD>` `1`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE и N начиная с 4.0.

paddingTop

Задает или возвращает расстояние между элементом страницы и верхней границей.

```
{Объект стиля}.paddingTop[ = "{Y}|{Y}%"];
```

Расстояние может быть задано как абсолютной величиной, так и процентом от высоты родителя. Значение по умолчанию `0`, для тегов `<TD>` `1`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE и N начиная с 4.0.

pageBreakAfter

Задает или возвращает признак: будет ли после текущего элемента при печати Web-страницы производиться прогон листа.

```
{Объект стиля}.pageBreakAfter[ = "auto|always|empty string"];
```

Доступны три значения:

- `auto` (значение по умолчанию) — передает управление размещением информации на листе операционной системе;
- `always` — заставляет принтер прогнать лист после печати текущего элемента страницы;
- `empty string` — запрещает принтеру делать это в любом случае.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

pageBreakBefore

Задает или возвращает признак: будет ли перед текущим элементом при печати Web-страницы производиться прогон листа.

```
{Объект стиля}.pageBreakBefore[ = "auto|always|empty string"];
```

Доступны три значения:

- `auto` (значение по умолчанию) — передает управление размещением информации на листе операционной системе;
- `always` — заставляет принтер прогнать лист перед печатью текущего элемента страницы;
- `empty string` — запрещает принтеру делать это в любом случае.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

pageX(event)

Задает или возвращает горизонтальную координату курсора мыши относительно страницы в пикселах.

```
event.pageX[ = {X}];
```

Поддерживается N начиная с 4.0.

pageX(слой)

Задает или возвращает горизонтальную координату левой стороны слоя относительно страницы в пикселах.

```
{Объект слоя}.pageX[ = {X}];
```

Поддерживается N начиная с 4.0.

pageXOffset

Возвращает положение горизонтальной полосы прокрутки.

```
window.pageXOffset;
```

Доступно только для чтения.

Поддерживается N начиная с 4.0.

pageY(event)

Задает или возвращает вертикальную координату курсора мыши относительно страницы в пикселах.

```
event.pageY[ = {Y}];
```

Поддерживается N начиная с 4.0.

pageY(слой)

Задает или возвращает вертикальную координату верхней стороны слоя относительно страницы в пикселах.

```
{Объект слоя}.pageY[ = {Y}];
```

Поддерживается N начиная с 4.0.

pageYOffset

Возвращает положение вертикальной полосы прокрутки.

```
window.pageYOffset;
```

Доступно только для чтения.

Поддерживается N начиная с 4.0.

palette

Возвращает цветовую палитру, используемую для внедренного объекта <EMBED>. Палитра возвращается в строковом виде.

```
{Внедренный объект}.palette;
```

Доступно только для чтения.

Поддерживается IE начиная с 3.02.

parent

Возвращает ссылку на родителя текущего окна или фрейма. Для фрейма родителем будет набор фреймов.

```
window.parent;
```

Доступно только для чтения.

Поддерживается IE начиная с 3.02.

parentElement

Возвращает ссылку на родителя текущего элемента страницы. Если элемент страницы не имеет родителей, возвращается `null`.

```
{Объект}.parentElement;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

parentLayer

Возвращает ссылку на родительский слой. Если такого нет, возвращается `null`.

```
{Объект слоя}.parentLayer;
```

Доступно только для чтения.

Поддерживается N начиная с 4.0.

parentNode

Возвращает ссылку на родительский элемент страницы. Если элемент страницы не имеет родителей, возвращается `null`.

```
{Объект}.parentNode;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.0.

parentStyleSheet

Возвращает имя файла импортированной таблицы стилей.

```
{Объект таблицы стилей}.parentStyleSheet;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

parentTextEdit

Возвращает ссылку на родительский элемент страницы, который может использоваться для создания объекта `TextRange`, содержащего текущий элемент страницы. Если элемент страницы не имеет родителей, возвращается `null`.

```
{Объект}.parentTextEdit;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

parentWindow

Возвращает ссылку на окно Web-обозревателя, где отображается текущая Web-страница.

```
document.parentWindow;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

pathname

Возвращает путь и имя файла, на который указывает гиперссылка.

```
{Объект гиперссылки}.pathname;
```

Доступно только для чтения.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

personalbar

Задает или возвращает состояние пользовательской панели: показана ли она на экране или скрыта.

```
window.personalbar[ = true|false];
```

Значение `true` показывает пользовательскую панель на экране, значение `false` скрывает.

Поддерживается N начиная с 4.0.

pixelBottom

Задает или возвращает вертикальную позицию нижней границы элемента страницы в пикселах.

```
{Объект стиля}.pixelBottom[ = {Y}];
```

Поддерживается IE начиная с 4.0.

pixelDepth

Возвращает цветовое разрешение видеоподсистемы компьютера — количество двоичных разрядов на пиксел.

```
screen.pixelDepth;
```

Доступно только для чтения.

Поддерживается N начиная с 4.0.

pixelHeight

Задает или возвращает высоту элемента страницы в пикселах.

```
{Объект стиля}.pixelHeight[ = {Y}];
```

Поддерживается IE начиная с 4.0.

pixelLeft

Задает или возвращает горизонтальную позицию левой границы элемента страницы в пикселах.

```
{Объект стиля}.pixelLeft[ = {X}];
```

Поддерживается IE начиная с 4.0.

pixelRight

Задает или возвращает горизонтальную позицию правой границы элемента страницы в пикселах.

```
{Объект стиля}.pixelRight[ = {X}];
```

Поддерживается IE начиная с 4.0.

pixelTop

Задает или возвращает вертикальную позицию верхней границы элемента страницы в пикселах.

```
{Объект стиля}.pixelTop[ = {Y}];
```

Поддерживается IE начиная с 4.0.

pixelWidth

Задает или возвращает ширину элемента страницы в пикселах.

```
{Объект стиля}.pixelWidth[ = {X}];
```

Поддерживается IE начиная с 4.0.

platform

Возвращает наименование платформы клиентского компьютера.

```
navigator.platform;
```

Возвращается одно из семи значений:

- `HP-UX` — Unix-совместимые компьютеры Hewlett-Packard;
- `MacPPC` — компьютеры Apple Macintosh с процессором Motorola PowerPC;

- Mac68K — компьютеры Apple Macintosh с процессором Motorola 68K;
- SunOS — компьютеры Sun Solaris;
- Win32 — 32-разрядные операционные системы Microsoft Windows;
- Win16 — 16-разрядные операционные системы Microsoft Windows;
- WinCE — операционные системы Microsoft Windows для портативных компьютеров.

Доступно только для чтения.

Поддерживается IE и N начиная с 4.0.

pluginspage

Возвращает интернет-адрес Web-страницы, содержащей инструкции по инсталляции расширения Web-обозревателя.

```
{Объект}.pluginspage;
```

Доступно только для чтения.

Поддерживается IE начиная с 3.02.

port

Задает или возвращает номер порта, являющийся частью интернет-адреса текущей Web-страницы или гиперссылки. Если номер порта не задан, возвращается 0 или пустая строка.

```
location.port[ = "{Порт}"];
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

posBottom

Задает или возвращает вертикальную позицию нижней границы элемента страницы в единицах измерения, заданных свойством bottom.

```
{Объект стиля}.posBottom[ = {Y}];
```

Поддерживается IE начиная с 4.0.

posHeight

Задает или возвращает высоту элемента страницы в единицах измерения, заданных свойством height.

```
{Объект стиля}.posHeight[ = {Y}];
```

Поддерживается IE начиная с 4.0.

position

Задает или возвращает тип позиционирования текущего элемента страницы: свободное или статическое.

```
{Объект стиля}.position[ = "static|absolute|relative"];
```

Доступны три значения:

- `static` (значение по умолчанию) — задает статическое позиционирование, при котором элемент страницы отображается внутри общего "потока" текста, т. е. не свободно. Значения атрибутов `bottom`, `left`, `right` и `top` при этом не принимаются Web-обозревателем во внимание;
- `absolute` — задает абсолютное свободное позиционирование. Значения атрибутов `bottom`, `left`, `right` и `top` при этом задают абсолютные координаты элемента страницы относительно родителя;
- `relative` — задает относительное свободное позиционирование. Значения атрибутов `bottom`, `left`, `right` и `top` при этом задают смещения координат элемента страницы от точки, в которой бы он был отображен, будь атрибут `position` установлен в `static`.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

posLeft

Задает или возвращает горизонтальную позицию левой границы элемента страницы в единицах измерения, заданных свойством `left`.

```
{Объект стиля}.posLeft[ = {X}];
```

Поддерживается IE начиная с 4.0.

posRight

Задает или возвращает горизонтальную позицию правой границы элемента страницы в единицах измерения, заданных свойством `right`.

```
{Объект стиля}.posRight[ = {X}];
```

Поддерживается IE начиная с 4.0.

posTop

Задает или возвращает вертикальную позицию верхней границы элемента страницы в единицах измерения, заданных свойством top.

```
{Объект стиля}.posTop[ = {Y}];
```

Поддерживается IE начиная с 4.0.

posWidth

Задает или возвращает ширину элемента страницы в единицах измерения, заданных свойством width.

```
{Объект стиля}.posWidth[ = {X}];
```

Поддерживается IE начиная с 4.0.

previous

Возвращает интернет-адрес предыдущей позиции в списке "истории".

```
history.previous;
```

Доступно только для чтения.

Поддерживается N начиная с 3.0.

previousSibling

Возвращает ссылку на предыдущего потомка родителя текущего элемента страницы.

```
{Объект}.previousSibling;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.0.

propertyName

Задает или возвращает имя свойства, значение которого изменилось.

```
event.propertyName[ = "{Имя свойства}"];
```

Поддерживается IE начиная с 5.0.

protocol

Задает или возвращает наименование интернет-протокола, являющееся частью интернет-адреса текущей Web-страницы, рисунка или гиперссылки. Если номер протокола не задан, возвращается 0 или пустая строка.

```
location.protocol[ = "{Протокол}"];
```

Поддерживается IE начиная с 3.02 для <A>, <AREA>, document и location и начиная с 4.0 для <IMG>. Поддерживается N начиная с 2.0.

pseudoClass

Возвращает идентификатор псевдокласса печатаемой страницы, к которому будет применяться правило, заданное через @page.

```
page.pseudoClass;
```

Возвращается одно из трех значений:

- :first — первая печатаемая страница;
- :left — четная страницы;
- :right — нечетная страница.

Доступно только для чтения.

Поддерживается IE начиная с 5.5.

qualifier

Задает или возвращает имя именованного поднабора данных.

```
event.qualifier[ = "{Имя именованного поднабора данных}"];
```

В IE 4.x это свойство доступно только для чтения.

Пример использования свойства qualifier.

```
<OBJECT CLASSID="clsid:00000000-0000-0000-0000-000000000000"
ID="dsoSpreadSheet" ondatasetcomplete="dsComplete();">
```

Здесь мы определяем источник данных, привязанный к файлу Microsoft Excel.

```
<TABLE DATASRC="#dsoSpreadsheet.A1:A7">
   <TR><TD><SPAN DATAFLD="A"></SPAN></TD></TR>
</TABLE>
```

Эта таблица привязана к именованному поднабору данных A1:A7, представляющему собой диапазон ячеек таблицы Microsoft Excel.

```
<SCRIPT>
function dsComplete() { window.alert(oEvent.qualifier); }
</SCRIPT>
```

В этом скрипте мы можем получить доступ к имени поднабора данных.

Поддерживается IE начиная с 4.0.

readOnly (поля ввода)

Задает или возвращает признак поля ввода: может ли пользователь редактировать в нем текст.

```
{Объект поля ввода}.readOnly[ = true|false];
```

По умолчанию пользователь имеет возможность редактировать текст в поле ввода (значение `false`). Если вы хотите сделать этот текст нередактируемым, дайте этому свойству значение `true`. Обратите внимание: при этом поле ввода останется разрешенным для доступа; пользователь сможет выделять и копировать из него текст (`readOnly` не то же самое, что `disabled`).

Поддерживается IE начиная с 4.0.

readOnly (правила и таблицы стилей)

Возвращает `true`, если правило или таблица стилей определены на текущей странице, и `false`, если они импортированы.

```
{Объект}.readOnly;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

readyState

Возвращает текущее состояние элемента страницы.

```
{Объект}.readyState;
```

Возвращается одно из пяти значений:

- `uninitializes` — элемент страницы не инициализирован;
- `loading` — загружает данные;
- `loaded` — уже загрузил данные;
- `interactive` — инициализирован не полностью, но уже доступен пользователю;
- `complete` — полностью инициализирован.

Доступно только для чтения.

Поддерживается IE начиная с 4.0 для `document`, `<IMG>`, `<LINK>`, `<SCRIPT>`, начиная с 5.0 для подавляющего большинства остальных элементов страниц, кроме фреймов, и начиная с 5.5 для фреймов.

readyState (<OBJECT>)

Возвращает текущее состояние внедренного объекта.

```
{Внедренный объект}.readyState;
```

Возвращается одно из пяти значений:

- 0 — внедренный объект не инициализирован;
- 1 — загружает данные;
- 2 — уже загрузил данные;
- 3 — инициализирован не полностью, но уже доступен пользователю;
- 4 — полностью инициализирован.

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

reason

Задает или возвращает результат передачи данных источником данных.

```
event.reason;
```

Возвращается одно из трех значений:

- 0 — данные приняты благополучно;
- 1 — процесс получения данных был прерван;
- 2 — во время получения данных произошла ошибка.

В IE 4.x это свойство доступно только для чтения.

Поддерживается IE начиная с 4.0.

recordNumber

Возвращает порядковый номер записи в базе данных.

```
{Объект}.recordNumber;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

recordset

Задает или возвращает ссылку на объект источника данных.

```
event.recordset[ = {Источник данных}];
```

Поддерживается IE начиная с 4.0.

referrer

Возвращает интернет-адрес Web-страницы, с которой пользователь перешел на текущую страницу. Если же пользователь перешел на нее простым набором адреса в строке Web-обозревателя, возвращается пустая строка.

```
document.referrer;
```

Доступно только для чтения.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

rel

Задает или возвращает связь между текущей Web-страницей и той, на которую указывает гиперссылка.

```
{Объект гиперссылки}.rel[ = "Alternate|Appendix|Bookmark|Chapter|
↳Contents|Copyright|Glossary|Help|Index|Next|Offline|Prev|Section|
↳Shortcut Icon|Start|Stylesheet|Subsection">
```

Доступны семнадцать возможных значений:

- `Alternate` — гиперссылка указывает на страницу, заменяющую текущую;
- `Appendix` — на приложение к большому многостраничному документу;
- `Bookmark` — на закладку;
- `Chapter` — на отдельную часть большого многостраничного документа;
- `Contents` — на содержание большого многостраничного документа;
- `Copyright` — на соглашение об авторских правах;
- `Glossary` — на словарь терминов, используемых в большом многостраничном документе;
- `Help` — на справочную страницу;
- `Index` — на список всех страниц большого многостраничного документа;
- `Next` — на следующий документ в последовательности;
- `Offline` — на CDF-файл "канала";
- `Prev` — на предыдущий документ в последовательности;
- `Section` — на отдельный раздел большого многостраничного документа;
- `Shortcut Icon` — на файл иконки, используемой для обозначения ссылки на текущую страницу;
- `Start` — на первый документ в последовательности;
- `Stylesheet` — на таблицу стилей;

- Subsection — на отдельный подраздел большого многостраничного документа.

Поддерживается IE начиная с 3.02 для <A> и начиная с 4.0 для <LINK>.

repeat

Задает или возвращает признак события onkeydown: наступило ли оно повторно.

```
event.repeat;
```

Возвращает true, если событие onkeydown наступило повторно, и false — в противном случае.

Поддерживается IE начиная с 5.0.

returnValue (event)

Задает или возвращает значение, возвращаемое обработчиком события. Задав значение false, можно запретить действие, происходящее при наступлении события по умолчанию.

```
event.returnValue[ = true|false];
```

Значение по умолчанию true.

Поддерживается IE начиная с 4.0.

returnValue (window)

Задает или возвращает значение, возвращаемое модальным диалоговым Web-окном. Это значение может быть любого типа.

```
window.returnValue[ = {Возвращаемое значение}];
```

Поддерживается IE начиная с 4.0.

rev

Задает или возвращает связь между Web-страницей, на которую указывает гиперссылка, и текущей страницей.

```
{Объект гиперссылки}.rev[ = "Alternate|Appendix|Bookmark|Chapter|
↳Contents|Copyright|Glossary|Help|Index|Next|Prev|Section|Start|
↳Stylesheet|Subsection">
```

Доступны пятнадцать возможных значений:

- Alternate — страница, заменяющая другую;
- Appendix — приложение к большому многостраничному документу;

- `Bookmark` — закладка;
- `Chapter` — отдельная часть большого многостраничного документа;
- `Contents` — содержание большого многостраничного документа;
- `Copyright` — соглашение об авторских правах;
- `Glossary` — словарь терминов, используемых в большом многостраничном документе;
- `Help` — справочная страница;
- `Index` — список всех страниц большого многостраничного документа;
- `Next` — следующий документ в последовательности;
- `Prev` — предыдущий документ в последовательности;
- `Section` — отдельный раздел большого многостраничного документа;
- `Start` — первый документ в последовательности;
- `Stylesheet` — таблица стилей;
- `Subsection` — отдельный небольшой подраздел большого многостраничного документа.

Поддерживается IE начиная с 3.02 для `<A>` и начиная с 4.0 для `<LINK>`.

right (*TextRectangle*)

Задает или возвращает горизонтальную координату правой границы объекта `TextRectangle` в пикселах.

```
{Объект TextRectangle}.right[ = {X}];
```

Поддерживается IE начиная с 5.0.

right (CSS)

Задает или возвращает горизонтальную позицию правой границы свободно позиционированного элемента относительно родителя.

```
{Объект стиля}.right[ = "auto|{X}|{X}%"];
```

Координата может быть задана как абсолютной величиной, так и процентом от ширины родителя. Предопределенное значение `auto` заставляет Web-обозреватель позиционировать элемент самостоятельно. Значение по умолчанию `auto`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

rightMargin

Задает или возвращает расстояние от правой границы окна Web-обозревателя до правой границы содержимого Web-страницы в пикселах.

```
body.rightMargin[ = "{Расстояние}"];
```

Значение по умолчанию 10.

Поддерживается IE начиная с 4.0.

rowIndex

Возвращает позицию строки <TR> в коллекции rows таблицы <TABLE>.

```
{Объект строки таблицы}.rowIndex;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

rows (<FRAMESET>)

Задает или возвращает количество и высоту всех фреймов-строк в наборе фреймов.

```
{Объект набора фреймов}.rows[ = "Список высот фреймов-строк, разделенных
↳запятыми"];
```

Поддерживается IE начиная с 3.02.

rows (<TEXTAREA>)

Задает или возвращает высоту области редактирования в строках текста.

```
{Объект области редактирования}.rows[ = "{Высота в строках}"];
```

Поддерживается IE начиная с 3.02.

rowSpan

Задает или возвращает количество строк в таблице, которые должны быть объединены в одну.

```
{Объект ячейки}.rowSpan[ = "{Количество строк, объединяемых в одну}"];
```

Поддерживается IE начиная с 3.02.

rubyAlign

Задает или возвращает местонахождение текста аннотации, заданного тегом <RT>.

```
{Объект стиля}.rubyAlign[ = "auto|left|center|right|distribute-letter|
↳distribute-space|line-edge"];
```

Доступны семь значений:

- `auto` (значение по умолчанию) — заставляет Web-обозреватель самому выбрать способ выравнивания текста аннотации;
- `left` — выравнивает текст аннотации по левому краю;
- `center` — по центру;
- `right` — по правому краю;
- `distribute-letter` — "распределяет" текст аннотации по всему доступному пространству;
- `distribute-space` — то же самое, что `distribute-letter`, только с добавлением дополнительного пространства между словами;
- `line-edge` — "прижимает" текст аннотации к близлежащей границе элемента страницы или центрирует, если таковых поблизости нет.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.0.

rubyOverhang

Задает или возвращает способ перекрытия текстом аннотации смежного текста.

```
{Объект стиля}.rubyOverhang[ = "auto|whitespace|none"];
```

Доступны три значения:

- `auto` (значение по умолчанию) — задает перекрытие любого смежного текста;
- `whitespace` — заставляет текст аннотации перекрывать только свободное пространство;
- `none` — заставляет текст аннотации перекрывать только смежный с ним текст.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.0.

rubyPosition

Задает или возвращает местоположение текста аннотации.

```
{Объект стиля}.rubyPosition[ = "above|inline"];
```

Доступны два значения: above (значение по умолчанию) помещает текст аннотации над аннотируемым текстом, а inline — внутри его. Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 5.0.

rules

Задает или возвращает способ отображения внутренних границ таблицы.

```
{Объект таблицы}.rules[ = "none|rows|cols|groups|all"];
```

Доступны пять значений:

- none — внутренних границ нет совсем;
- rows — рисуются только горизонтальные линии (между строками);
- cols — рисуются только вертикальные линии (между столбцами);
- groups — рисуются вертикальные линии между группами колонок <COLGROUP> и горизонтальные линии между секциями <THEAD>, <TBODY> и <TFOOT>;
- all — рисуются все внутренние границы.

Поддерживается IE начиная с 4.0.

saveType

Возвращает формат данных, сохраненных в Буфере обмена Windows, при наступлении события oncontentsave.

```
event.saveType;
```

Возвращается одно из двух значений: TEXT (данные сохранены в текстовом формате) или HTML (в виде HTML-кода). Свойство доступно только для чтения.

Поддерживается IE начиная с 5.5.

scopeName

Возвращает строковое имя пространства имен, используемое в данной Web-странице.

```
{Объект}.scopeName;
```

Пространства имен предназначены для создания пользовательских HTML-тегов. Собственно, пространство имен содержит описания каждого набора таких пользовательских тегов, составленное на языке XML; имя используемого пространства имен указывается в атрибуте XMLNS тега <HTML>. Примене-

ние пространства имен позволит Web-обозревателю контролировать правильность HTML-кода, содержащего пользовательские теги.

Если Web-странице не было присвоено никакого пространства имен (т. е. HTML-код ее не содержит пользовательских тегов), возвращается значение по умолчанию HTML. В противном случае возвращается имя пространства имен. Свойство доступно только для чтения.

Поддерживается IE начиная с 5.0.

screenLeft

Возвращает горизонтальную координату левой стороны клиентской части окна Web-обозревателя (без учета границ, заголовка, строки меню и полос прокрутки) относительно экрана компьютера.

```
window.screenLeft;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.0.

screenTop

Возвращает вертикальную координату верхней стороны клиентской части окна Web-обозревателя (без учета границ, заголовка, строки меню и полос прокрутки) относительно экрана компьютера.

```
window.screenTop;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.0.

screenX

Задает или возвращает горизонтальную координату курсора мыши относительно экрана компьютера.

```
event.screenX[ = {X}];
```

В IE 4.x доступно только для чтения.

Поддерживается IE и N начиная с 4.0.

screenY

Задает или возвращает вертикальную координату курсора мыши относительно экрана компьютера.

```
event.screenY[ = {Y}];
```

В IE 4.x доступно только для чтения.

Поддерживается IE и N начиная с 4.0.

scroll

Задает или возвращает признак тела документа: будет ли окно Web-обозревателя иметь полосы прокрутки.

```
body.scroll[ = "yes|no|auto"];
```

Доступны три значения:

- `yes` — полосы прокрутки есть всегда, даже если они не нужны (значение по умолчанию);
- `no` — полос прокрутки нет, даже если содержимое Web-страницы не помещается в окно;
- `auto` — полосы прокрутки появляются, только если в них есть необходимость.

Поддерживается IE начиная с 4.0.

scrollAmount

Задает или возвращает шаг в пикселах, на который будет смещаться текст в `<MARQUEE>`.

```
{Объект}.scrollAmount[ = "{Шаг в пикселах}"];
```

Значение по умолчанию 6.

Поддерживается IE начиная с 3.02.

scrollbar3dLightColor

Задает или возвращает цвет верхней и левой границ полосы прокрутки, ее бегунка и стрелок.

```
{Объект стиля}.scrollbar3dLightColor[ = "{Цвет}"];
```

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.5.

scrollbarArrowColor

Задает или возвращает цвет стрелок на кнопках полосы прокрутки.

```
{Объект стиля}.scrollbarArrowColor[ = "{Цвет}"];
```

Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 5.5.

scrollbarBaseColor

Задает или возвращает цвет бегунка и кнопок-стрелок полосы прокрутки.

```
{Объект стиля}.scrollbarBaseColor[ = "{Цвет}"];
```

Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 5.5.

scrollbarDarkShadowColor

Задает или возвращает цвет "тени", "отбрасываемой" бегунком и кнопками прокрутки полосы прокрутки (цвет правых и нижних их граней).

```
{Объект стиля}.scrollbarDarkShadowColor[ = "{Цвет}"];
```

Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 5.5.

scrollbarFaceColor

Задает или возвращает цвет бегунка и кнопок прокрутки полосы прокрутки.

```
{Объект стиля}.scrollbarFaceColor[ = "{Цвет}"];
```

Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 5.5.

scrollbarHighlightColor

Задает или возвращает цвет "освещенной" части бегунка и кнопок прокрутки полосы прокрутки (цвет левых и верхних их граней).

```
{Объект стиля}.scrollbarHighlightColor[ = "{Цвет}"];
```

Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 5.5.

scrollbars

Задает или возвращает состояние полос прокрутки: показаны ли они на экране или скрыты.

```
window.scrollbars[ = true|false];
```

Значение `true` показывает полосы прокрутки на экране, значение `false` скрывает.

Поддерживается N начиная с 4.0.

scrollbarShadowColor

Задает или возвращает цвет "неосвещенной" части бегунка и кнопок прокрутки полосы прокрутки (цвет правых и нижних их граней). Не путать с цветом "тени", задаваемым свойством `scrollbarDarkShadowColor`.

```
{Объект стиля}.scrollbarShadowColor[ = "{Цвет}"];
```

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.5.

scrollbarTrackColor

Задает или возвращает цвет рабочей части полосы прокрутки, т. е. той ее части, по которой перемещается бегунок.

```
{Объект стиля}.scrollbarTrackColor[ = "{Цвет}"];
```

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.5.

scrollDelay

Задает или возвращает задержку в миллисекундах перед каждым шагом прокрутки текста в `<MARQUEE>`.

```
{Объект}.scrollDelay[ = "{Задержка в миллисекундах}"];
```

Значение по умолчанию 85.

Поддерживается IE начиная с 3.02.

scrollHeight

Возвращает полную высоту элемента страницы, включая и ту его часть, которая в данный момент не видна в окне Web-обозревателя.

```
{Объект}.scrollHeight;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0 для внедренных элементов, элементов управления и таблиц и начиная с 5.0 для остальных элементов.

scrolling

Задает или возвращает признак фрейма: будет ли его содержимое при необходимости прокручиваться.

```
{Объект фрейма}.scrolling[ = "yes|no|auto"];
```

Доступны три значения:

- `yes` — будет прокручиваться (появятся полосы прокрутки);
- `no` — не будет прокручиваться (полосы прокрутки не появятся, даже если в них возникнет необходимость);
- `auto` — полосы прокрутки появляются, только если в них есть необходимость (значение по умолчанию).

Поддерживается IE начиная с 3.02.

scrollLeft

Задает или возвращает положение горизонтальной полосы прокрутки элемента страницы. Другими словами, задает или возвращает расстояние между левой границей содержимого элемента страницы и левой границей его клиентской области.

```
{Объект}.scrollLeft[ = {X}];
```

Поддерживается IE начиная с 4.0 для внедренных элементов, элементов управления и таблиц и начиная с 5.0 для остальных элементов.

scrollTop

Задает или возвращает положение вертикальной полосы прокрутки элемента страницы. Другими словами, задает или возвращает расстояние между верхней границей содержимого элемента страницы и верхней границей его клиентской области.

```
{Объект}.scrollTop[ = {Y}];
```

Поддерживается IE начиная с 4.0 для внедренных элементов, элементов управления и таблиц и начиная с 5.0 для остальных элементов.

scrollWidth

Возвращает полную ширину элемента страницы, включая и ту его часть, что в данный момент не видна в окне Web-обозревателя.

```
{Объект}.scrollWidth;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0 для внедренных элементов, элементов управления и таблиц и начиная с 5.0 для остальных элементов.

search

Задает или возвращает параметры, переданные серверной программе как часть интернет-адреса текущей Web-страницы или гиперссылки. Строка параметров помещается в самом конце интернет-адреса после вопросительного знака ?.

```
location.search[ = "{Параметры}"];
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

sectionRowIndex

Возвращает позицию строки <TR> в коллекции rows секции <TBODY>, <TFOOT> или <THEAD>.

```
{Объект строки таблицы}.sectionRowIndex;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

selected

Задает или возвращает признак пункта списка <OPTION>: будет ли он выбран изначально.

```
{Объект пункта списка}.selected[ = true|false];
```

По умолчанию в списке будет выбран первый пункт (значение false). Если вы хотите, чтобы был выбран текущий пункт, дайте этому свойству значение true.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

selectedIndex

Задает или возвращает позицию выбранного пункта списка <SELECT>. Если ни один из пунктов не выбран, возвращается -1.

```
{Объект списка}.selectedIndex;
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

selector

Возвращает строку, обозначающую страницу, к которой будет применено правило, заданное в `@page`.

```
page.selector;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.5.

selectorText

Задает или возвращает строку, обозначающую элементы страницы, к которым будет применено текущее правило таблицы стилей.

```
{Объект}.selectorText[ = "{Элементы страницы}"];
```

Значения этому свойству задаются в виде `"H1"` или `"H1 B"`.

Поддерживается IE начиная с 5.0.

self

Возвращает ссылку на текущее окно или фрейм (объект `window`).

```
{Объект}.self;
```

Доступно только для чтения.

Поддерживается IE начиная с 3.02 для `window` и начиная с 4.0 для `<FRAME>`. Поддерживается N начиная с 2.0.

shape

Задает или возвращает форму "горячей" области на карте-изображении.

```
{Объект области}.shape[ = "circ|circle|poly|polygon|rect|rectangle"];
```

Доступны три значения:

- `circ` (или `circle`) — круг;
- `poly` (или `polygon`) — многоугольник;
- `rect` (или `rectangle`) — прямоугольник.

Координаты "горячей" области задаются свойством `coords`.

Поддерживается IE начиная с 3.02.

shiftKey

Задает или возвращает состояние клавиши <Shift>.

```
event.shiftKey[ = false|true];
```

Доступны два значения: `false` говорит о том, что клавиша <Shift> не нажата, а `true` — что нажата.

Поддерживается IE начиная с 4.0 только для чтения и начиная с 5.0 для чтения и записи.

shiftLeft

Задает или возвращает состояние левой клавиши <Shift>.

```
event.shiftLeft[ = false|true];
```

Доступны два значения: `false` говорит о том, что левая клавиша <Shift> не нажата, а `true` — что нажата.

Поддерживается IE начиная с 5.5 и только на операционных системах Windows NT 4.0 и Windows 2000.

siblingAbove

Возвращает ссылку на слой, находящийся выше текущего и помещающийся в том же родительском слое. Если текущий слой — самый высокий, возвращается ссылка на окно Web-обозревателя.

```
{Объект слоя}.siblingAbove;
```

Доступно только для чтения.

Поддерживается N начиная с 4.0.

siblingBelow

Возвращает ссылку на слой, находящийся ниже текущего и помещающийся в том же родительском слое. Если текущий слой — самый низкий, возвращается `null`.

```
{Объект слоя}.siblingBelow;
```

Доступно только для чтения.

Поддерживается N начиная с 4.0.

size (<BASEFONT> и <FONT>)

Задает или возвращает размер шрифта: от 1 (минимальный) до 7 (максимальный).

```
{Объект}.size[ = "{Размер шрифта}"];
```

Поддерживается IE начиная с 3.02.

size (<HR>)

Задает или возвращает высоту горизонтальной линейки в пикселах.

```
{Объект горизонтальной линии}.size[ = "{Высота в пикселах}"];
```

Поддерживается IE начиная с 3.02.

size (элементы управления)

Задает или возвращает размер элемента управления. Если это поле ввода, задает его размер в символах текста. Если список или всплывающее меню, задает размер в пунктах. Во всех остальных случаях никак не влияет на внешний вид элемента управления.

```
{Объект поля ввода}.size[ = "{Размер в символах}"];
{Объект списка}.size[ = "{Количество одновременно видимых пунктов}"];
```

Поддерживается IE начиная с 3.02.

sourceIndex

Возвращает позицию элемента страницы в коллекции `all`.

```
{Объект}.sourceIndex;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

span

Задает или возвращает количество колонок, на которое распространяются параметры, заданные в тегах <COL> и <COLGROUP>.

```
{Объект}.span[ = "{Количество колонок}"];
```

Поддерживается IE начиная с 3.02.

specified

Возвращает `true`, если значение атрибута было задано, и `false` — в противном случае.

```
{Объект атрибута}.specified;
```

Доступно только для чтения.

Пример использования свойства `specified`.

```
if (oList.attributes(0).specified) {...}
```

Поддерживается IE начиная с 5.0.

src (<APPLET>, <BGSOUND>, <EMBED>, <XML>, фреймы, рисунки, слои)

Задает или возвращает интернет-адрес файла, содержащего загружаемый элемент страницы: Java-апплет, данные, обрабатываемые с помощью расширения Web-обозревателя, содержимое фрейма, слоя, фоновый звук или рисунок.

```
{Внедренный объект}.src[ = "{Интернет-адрес файла с Java-апплетом}"];
{Объект рисунка}.src[ = "{Интернет-адрес файла рисунка}"];
```

Поддерживается IE начиная с 3.02 для <BGSOUND>, <EMBED>, <FRAME>, <IFRAME> и <IMG>, начиная с 4.0 для <APPLET> и <INPUT TYPE="image"> и начиная с 5.0 для <XML>. Поддерживается N начиная с 3.0.

src (<SCRIPT>)

Задает или возвращает интернет-адрес файла с текстом скрипта.

```
{Объект скрипта}.src[ = "{Интернет-адрес файла с текстом скрипта}"];
```

Поддерживается IE начиная с 4.0.

srcElement

Задает или возвращает ссылку на элемент страницы, вызвавший событие.

```
event.srcElement[ = {Объект}];
```

В IE 4.x доступно только для чтения.

Поддерживается IE начиная с 4.0.

srcFilter

Задает или возвращает ссылку на фильтр, вызвавший событие onfilterchange.

```
event.srcFilter[ = {Объект фильтра}];
```

В IE 4.x доступно только для чтения.

Поддерживается IE начиная с 4.0.

srcUrn

Задает или возвращает имя сетевого ресурса URN поведения, вызвавшего событие.

```
event.srcUrn[ = {Поведение}];
```

Поддерживается IE начиная с 5.0.

start (<IMG>)

Задает или возвращает момент, с которого начнет воспроизводиться видеоклип, имя файла которого было задано в атрибуте DYNSRC.

```
{Объект рисунка}.start[ = "fileopen|mouseover"];
```

Доступно два значение: `fileopen` (значение по умолчанию) запускает клип сразу после его загрузки, а `mouseover` — только после того, как пользователь разместит над ним курсор мыши.

Поддерживается IE начиная с 3.02.

start (<OL>)

Задает или возвращает начальную позицию нумерации элементов пронумерованного списка <OL>.

```
{Объект нумерованного списка}.start[ = "{Начальный номер}"];
```

Значение по умолчанию 1.

Поддерживается IE начиная с 3.02.

status (window)

Задает или возвращает текст, отображаемый в строке состояния окна Web-обозревателя.

```
window.status[ = "{Текст}"];
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

status (флажки и радиокнопки)

Задает или возвращает состояние флажка или радиокнопки: включен, не включен, не инициализирован.

```
{Объект}.status[ = true|false|null];
```

Доступны три значения:

- `true` — флажок или радиокнопка включена;
- `false` (значение по умолчанию) — не включена;
- `null` — не инициализирована.

Поддерживается IE начиная с 4.0.

statusbar

Задает или возвращает состояние строки состояния: показана ли она на экране или скрыта.

```
window.statusbar[ = true|false];
```

Значение `true` показывает строку состояния на экране, значение `false` скрывает.

Поддерживается N начиная с 4.0.

style

Возвращает строковое определение встроенного стиля элемента страницы.

```
{Объект стиля}.style;
```

Доступно только для чтения.

Поддерживается IE начиная с 3.02 для текстовых элементов страницы и начиная с 4.0 для нетекстовых (рисунки, фреймы, расширения и т. п.).

styleFloat

Задает или возвращает обозначение края страницы, к которому будет "прижиматься" элемент. Например, рисунок может "прижиматься" к левому или правому краю страницы.

```
{Объект стиля}.styleFloat[ = "none|left|right"];
```

Доступны три значения:

- `none` (значение по умолчанию) — заставляет элемент страницы появляться там, где он задан, т. е. никуда не "прижиматься";
- `left` — "прижимает" элемент страницы к левому ее краю. Например, рисунок может находиться слева, а текст — "обтекать" его справа;
- `right` — "прижимает" элемент страницы к правому ее краю. При этом наш рисунок будет находиться справа, а текст — "обтекать" его слева.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0 для кнопок и внедренных объектов и начиная с 5.0 для остальных элементов страницы.

suffixes

Возвращает список расширений файлов, соответствующих текущему типу данных MIME.

```
{Объект типа MIME}.suffixes;
```

Доступно только для чтения.

Поддерживается N начиная с 3.0.

systemLanguage

Возвращает строковое обозначение языка текущей версии операционной системы клиента. Коды языков перечислены в приложении 1.

```
navigator.systemLanguage;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

tabIndex

Задает или возвращает номер в последовательности обхода элементов управления при нажатии клавиши <Tab>.

```
{Объект}.tabIndex[ = "{Порядковый номер в последовательности}"];
```

Значение по умолчанию 0. Если значения `tabIndex` одинаковы или вообще не заданы, обход выполняется в порядке появления элементов управления в исходном тексте Web-страницы. Если задано отрицательное значение, элемент управления исключается из последовательности.

Поддерживается IE начиная с 4.0 для элементов управления и начиная с 5.0 для остальных элементов страницы. Отрицательные значения атрибута поддерживаются начиная с 5.01.

tableLayout

Задает или возвращает признак таблицы: будут ли значения ширины ее ячеек "фиксированы" или нет.

```
{Объект таблицы}.style.tableLayout[ = "auto|fixed"];
```

Доступны два значения: `auto` (значение по умолчанию) устанавливает ширину ячейки по ширине ее содержимого, а `fixed` использует для установки значений ширины ячеек значения свойств `width` или, если они не заданы, просто дает ячейкам равную ширину. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.0.

tabStop

Задает или возвращает признак поведения: будут ли элементы страницы по умолчанию принимать фокус текстового ввода и находиться в последовательности обхода по нажатии клавиши <Tab>.

```
defaults.tabStop[ = true|false];
```

По умолчанию элементы страницы не принимают фокус текстового ввода и не участвуют в последовательности обхода (значение `false`). Чтобы наделить их такой особенностью, дайте этому свойству значение `true`.

Поддерживается IE начиная с 5.5.

tagName

Возвращает имя тега элемента страницы.

```
{Объект}.tagName;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

tagUrn

Задает или возвращает имя сетевого ресурса URN, заданное в объявлении пространства имен.

```
{Объект}.tagUrn;
```

Поддерживается IE начиная с 5.0.

target (event)

Задает или возвращает ссылку на элемент страницы, в котором изначально наступило данное событие.

```
event.target[ = {Объект}];
```

Поддерживается N начиная с 4.0.

target (гиперссылки и формы)

Задает или возвращает имя фрейма или окна Web-обозревателя, куда будет загружена Web-страница, на которую ссылается гиперссылка.

```
{Объект}.target[ = "{Имя окна или фрейма}|_blank|_parent|_search|_self|
↪_top"];
```

Кроме собственно имени окна или фрейма доступны также пять предопределенных значений:

- `_blank` — загружает Web-страницу в новое окно;
- `_parent` — загружает Web-страницу в родительское окно;
- `_search` — загружает Web-страницу в панель поиска Web-обозревателя (доступно в IE начиная с версии 5.0);

- _self — загружает Web-страницу в то же самое окно (значение по умолчанию);
- _top — загружает Web-страницу в самое верхнее окно в иерархии.

Поддерживается IE начиная с 3.02 для <A>, <AREA>, <BASE> и <FORM> и начиная с 4.0 для <LINK>. Поддерживается N начиная с 2.0.

text (<A>)

Возвращает текст "якоря" или гиперссылки.

```
{Объект}.text;
```

Поддерживается N начиная с 2.0 для гиперссылок и начиная с 4.0 для "якорей".

text (body)

Задает или возвращает цвет текста документа.

```
body.text[ = "{Цвет}"];
```

Поддерживается IE начиная с 3.02.

text (<OPTION>)

Задает или возвращает текст, содержащийся в пункте списка.

```
{Объект пункта списка}.text[ = "{Текст}"];
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

text (<SCRIPT>, <TITLE>, комментарии)

Задает или возвращает текст, содержащийся в элементе страницы.

```
{Объект}.text[ = "{Текст}"];
```

Поддерживается IE начиная с 4.0.

text (TextRange)

Задает или возвращает текст, содержащийся в объекте.

```
{Объект}.text[ = "{Текст}"];
```

Поддерживается IE начиная с 4.0.

textAlign

Задает или возвращает горизонтальное выравнивание текста.

```
{Объект стиля}.textAlign[ = "left|right|center|justify"];
```

Доступны четыре значения:

- `left` (значение по умолчанию) — выравнивает текст по левому краю;
- `right` — по правому краю;
- `center` — по центру;
- `justify` — по обоим краям (по ширине).

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 3.02; значение `justify` поддерживается начиная с 4.0. Поддерживается N начиная с 4.0.

textAlignLast

Задает или возвращает горизонтальное выравнивание последней строки абзаца.

```
{Объект стиля}.textAlignLast[ = "auto|inherit|left|right|center|
↳justify"];
```

Доступны шесть значений:

- `auto` (значение по умолчанию) — выравнивает последнюю строку абзаца так же, как остальные строки (основываясь на значении свойства `textAlign`);
- `inherit` — так же, как выровнен текст родителя;
- `left` — по левому краю;
- `right` — по правому краю;
- `center` — по центру;
- `justify` — по обоим краям (по ширине).

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.5.

textAutospace

Задает или возвращает интервал между фрагментами текста, написанными на разных языках.

```
{Объект стиля}.textAutospace[ = "none|ideograph-alpha|ideograph-numeric|
↳ideograph-parenthesis|ideograph-space"];
```

Доступны пять значений:

- `none` (значение по умолчанию) — запрещает добавлять интервал между фрагментами текста;

- `ideograph-alpha` — добавляет интервал между иероглифическими и неиероглифическими (латинскими, кириллическими, греческими и т. д.) фрагментами текста;
- `ideograph-numeric` — добавляет интервал между иероглифическим текстом и цифрами;
- `ideograph-parenthesis` — добавляет интервал между иероглифическим текстом и круглыми скобками;
- `ideograph-space` — увеличивает ширину интервалов, граничащих с иероглифическим текстом.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.0.

textDecoration

Задает или возвращает специальное оформление текста: подчеркнутый, зачеркнутый и т. п.

```
{Объект стиля}.textDecoration[ = "none|underline|overline|line-through|
↳blink"];
```

Доступны пять значений:

- `none` — отменяет любое специальное оформление (значение по умолчанию для большинства тегов);
- `underline` — подчеркивает текст (значение по умолчанию для тегов <A>, <INS> и <U>);
- `overline` — "надчеркивает" текст;
- `line-through` — зачеркивает текст (значение по умолчанию для тегов <DEL>, <S> и <STRIKE>);
- `blink` — заставляет текст мерцать (IE не поддерживается).

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 3.02. Поддерживается N начиная с 4.0.

textDecorationBlink

Задает или возвращает признак текста: будет ли он мерцать (имеет ли свойство `textDecoration` значение `blink`).

```
{Объект стиля}.textDecorationBlink[ = true|false];
```

Значение true заставляет текст мерцать, а false отменяет мерцание. Но в действительнсти IE не поддерживает отображение мерцающего текста.

Поддерживается IE начиная с 3.02 для текстовых элементов страницы и начиная с 4.0 для нетекстовых.

textDecorationLineThrough

Задает или возвращает признак текста: будет ли он зачеркнут (имеет ли свойство textDecoration значение line-through).

```
{Объект стиля}.textDecorationLineThrough[ = true|false];
```

Значение true делает текст зачеркнутым, а false убирает зачеркивание.

Поддерживается IE начиная с 3.02 для текстовых элементов страницы и начиная с 4.0 для нетекстовых.

textDecorationNone

Задает или возвращает признак текста: имеет ли он какое-либо дополнительное оформление (имеет ли свойство textDecoration значение none).

```
{Объект стиля}.textDecorationNone[ = true|false];
```

Значение true убирает все дополнительное оформление.

Поддерживается IE начиная с 3.02 для текстовых элементов страницы и начиная с 4.0 для нетекстовых.

textDecorationOverline

Задает или возвращает признак текста: будет ли он "надчеркнут" (имеет ли свойство textDecoration значение overline).

```
{Объект стиля}.textDecorationOverline[ = true|false];
```

Значение true делает текст "надчеркнутым", а false убирает "надчеркивание".

Поддерживается IE начиная с 3.02 для текстовых элементов страницы и начиная с 4.0 для нетекстовых.

textDecorationUnderline

Задает или возвращает признак текста: будет ли он подчеркнут (имеет ли свойство textDecoration значение underline).

```
{Объект стиля}.textDecorationUnderline[ = true|false];
```

Значение true делает текст подчеркнутым, а false убирает подчеркивание.

Поддерживается IE начиная с 3.02 для текстовых элементов страницы и начиная с 4.0 для нетекстовых.

textIndent

Задает или возвращает величину отступа красной строки.

```
{Объект стиля}.textIndent[ = "{Отступ}|{Отступ}%"];
```

Отступ может быть задан как абсолютной величиной, так и процентом от ширины родителя. Значение по умолчанию `0`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE и N начиная с 4.0.

textJustify

Задает или возвращает тип выравнивания текста по ширине. Значение свойства `textAlign` при этом должно быть равно `justify`.

```
{Объект стиля}.textJustify[ = "auto|newspaper|distribute|
↳distribute-all-lines|distribute-center-last|inter-word|inter-ideograph|
↳inter-cluster|kashida"];
```

Доступны восемь значений:

- `auto` (значение по умолчанию) — отдает управление выравниванием по ширине на усмотрение Web-обозревателя;
- `newspaper` — выравнивает строки, изменяя расстояния между словами и между символами;
- `distribute` — аналогично `newspaper` и предназначено для азиатских языков (тайский и пр.);
- `distribute-all-lines` — аналогично `distribute` за тем исключением, что последняя строка абзаца также подвергается полному выравниванию. Предназначено для иероглифических языков;
- `distribute-center-last` — не реализовано;
- `inter-word` — выравнивает строки, изменяя только расстояния между словами;
- `inter-ideograph` — выравнивает строки иероглифического текста, изменяя расстояния между словами и между иероглифами;
- `inter-cluster` — выравнивает строки текста на азиатских языках, не содержащих пробелов между словами;
- `kashida` — выравнивает строки текста на арабском языке, изменяя ширину самих символов.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.0.

textKashidaSpace

Задает или возвращает процент, на который будут расширяться символы арабского языка при выравнивании текста по ширине. Может использоваться, лишь если свойство `textJustify` равно `auto`, `distribute`, `kashida` или `newspaper`.

```
{Объект стиля}.textKashidaSpace[ = "{Расширение}%|inherit"];
```

Величина отступа может быть задана как процент свободного пространства между символами, на которое они могут расширяться. Значение `0%` (используется по умолчанию) означает, что расширение символов недопустимо, а вместо них будет расширяться свободное пространство; значение `100%` — что допустимо расширение лишь символов, но не свободного пространства. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.5.

textTransform

Задает или возвращает принцип преобразования регистра символов текста.

```
{Объект стиля}.textTransform[ = "none|capitalize|uppercase|lowercase"];
```

Доступны четыре значения:

- `none` (значение по умолчанию) — отключает любые преобразования регистра символов;
- `capitalize` — преобразует первую букву каждого слова текста в верхний регистр;
- `uppercase` — преобразует все символы текста в верхний регистр;
- `lowercase` — в нижний регистр.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE и N начиная с 4.0.

textUnderlinePosition

Задает или возвращает местонахождение линии подчеркивания: выше или ниже текста. Имеет смысл, если свойство `textDecoration` равно `underline` или `overline`.

```
{Объект стиля}.textUnderlinePosition[ = "below|above"];
```

Доступны два значения: `below` (значение по умолчанию) помещает линию подчеркивания под текстом, а `above` — над текстом ("надчеркивание"). Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.5.

tFoot

Возвращает ссылку на секцию основания <TFOOT> таблицы <TABLE>.

```
{Объект таблицы}.tFoot;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

tHead

Возвращает ссылку на секцию заголовка <THEAD> таблицы <TABLE>.

```
{Объект таблицы}.tHead;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

title

Задает или возвращает текст "всплывающей" подсказки для элемента страницы. Эта подсказка появляется при наведении на элемент страницы курсора мыши.

```
{Объект}.title[ = "{Текст подсказки}"];
```

Поддерживается IE начиная с 4.0.

title (document)

Возвращает заголовок Web-страницы, созданный с помощью тега <TITLE>.

```
document.title;
```

Доступно только для чтения.

Поддерживается N начиная с 2.0.

title (<STYLE>)

Задает или возвращает заголовок для таблицы стилей. Используется только для ее идентификации.

```
{Объект таблицы стилей}.title[ = "{Текст заголовка}"];
```

Поддерживается IE начиная с 4.0.

toElement

Задает или возвращает ссылку на элемент управления, находящийся под курсором мыши. Имеет смысл лишь при возникновении событий `onmouseover` и `onmouseout`.

```
event.toElement[ = {Элемент страницы}];
```

В IE 4.x доступно только для чтения.

Поддерживается IE начиная с 4.0.

toolbar

Задает или возвращает состояние панели инструментов: показана ли она на экране или скрыта.

```
window.toolbar[ = true|false];
```

Значение `true` соответствует отображению панели инструментов на экране, значение `false` скрывает.

Поддерживается N начиная с 4.0.

top (TextRectangle)

Задает или возвращает вертикальную координату верхней границы объекта `TextRectangle` в пикселах.

```
{Объект TextRectangle}.top[ = {Y}];
```

Поддерживается IE начиная с 5.0.

top (window)

Возвращает ссылку на самое верхнее окно-родитель.

```
window.top;
```

Доступно только для чтения.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

top (слой)

Задает или возвращает вертикальную координату верхней границы слоя в пикселах.

```
{Объект слоя}.top[ = {Y}];
```

Поддерживается N начиная с 4.0.

top (CSS)

Задает или возвращает вертикальную позицию верхней границы свободно позиционированного элемента относительно родителя.

```
{Объект стиля}.top[ = "auto|{Y}|{Y}%"];
```

Координата может быть задана как абсолютной величиной, так и процентом от высоты родителя. Предопределенное значение `auto` заставляет Web-обозреватель позиционировать элемент самостоятельно. Значение по умолчанию `auto`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

topMargin

Задает или возвращает расстояние от верхней границы окна Web-обозревателя до верхней границы содержимого Web-страницы в пикселах.

```
body.topMargin[ = "{Расстояние}"];
```

Значение по умолчанию 15.

Поддерживается IE начиная с 3.02.

trueSpeed

Задает или возвращает способ вычисления нового положения прокручиваемого текста в <MARQUEE>.

```
{Объект}.trueSpeed[ = true|false];
```

По умолчанию новое положение вычисляется каждые 60 миллисекунд, независимо от значений атрибутов `SCROLLAMOUNT` и `SCROLLDELAY` (значение `false`). Если вы хотите, чтобы новое положение текста вычислялось каждые `{SCROLLDELAY}` миллисекунд (более точно), дайте этому свойству значение `true`. Текст будет двигаться плавней, но прокрутка будет отнимать больше ресурсов у компьютера.

Поддерживается IE начиная с 4.0.

type (<BUTTON>)

Возвращает тип кнопки <BUTTON>.

```
{Объект кнопки}.type;
```

Возвращается одно из трех значений:

- button — обычная кнопка (значение по умолчанию);
- reset — кнопка сброса формы;
- submit — кнопка отправки данных формы.

Свойство доступно только для чтения.

Поддерживается IE начиная с 4.0.

type (<EMBED>, <LINK> и <OBJECT>)

Задает или возвращает тип данных MIME элемента ActiveX.

```
{Внедренный объект}.type[ = "{Тип данных MIME}"];
```

Поддерживается IE начиная с 3.02.

type (event)

Задает или возвращает имя наступившего события без префикса "on".

```
event.type[ = "{Имя события}"];
```

В IE 4.x это свойство доступно только для чтения.

Поддерживается IE и N начиная с 4.0.

type (<LI>, <OL> и <UL>)

Задает или возвращает тип маркера маркированного и тип нумерации нумерованного списка.

```
{Объект нумерованного списка}.type[ = "A|a|I|i|1"];
{Объект маркированного списка}.type[ = "disc|circle|square"];
```

Для нумерованного списка <OL> доступны пять значений:

- A — нумерует позиции списка большими латинскими буквами;
- a — малыми латинскими буквами;
- I — большими римскими цифрами;
- i — малыми римскими цифрами;
- 1 — арабскими цифрами (значение по умолчанию).

Для маркированного списка <UL> доступны три значения:

- disc — маркирует позиции списка сплошным кружком (значение по умолчанию);
- circle — окружностью без заливки;
- square — квадратиком.

Поддерживается IE начиная с 3.02 для <LI> и <OL> и начиная с 4.0 для <UL>.

type (plugin)

Возвращает описание текущего типа данных MIME.

```
{Объект типа MIME}.type;
```

Доступно только для чтения.

Поддерживается N начиная с 3.0.

type (<SCRIPT>)

Задает или возвращает тип интерпретатора (виртуальной машины) скриптового языка в стандарте MIME. Должен соответствовать языку, указанному свойством language.

```
{Объект скрипта}.type[ = "{Тип интерпретатора в стандарте MIME}"];
```

Доступны шесть значений, перечисленных в табл. П1.2.

Поддерживается IE начиная с 4.0.

type (selection)

Возвращает тип содержимого объекта selection.

```
selection.type;
```

Может возвращать три значения: none — ничего, text — текст и control — элемент страницы, позволяющий изменять свои размеры.

Поддерживается IE начиная с 4.0.

type (<STYLE>)

Задает или возвращает тип таблицы стилей в стандарте MIME.

```
{Объект таблицы стилей}.type[ = "{Тип данных MIME}"];
```

Для таблицы стилей он должен быть равен text/css.

Поддерживается IE начиная с 4.0.

type (элементы управления)

Задает первоначально или возвращает тип элемента управления: кнопка, флажок, поле ввода или что-то другое.

```
{Объект}.type[ = "button|checkbox|file|hidden|image|password|radio|reset|
↳submit|text"];
```

Для записи доступны десять значений, перечисленных в табл. П1.1. Запись значения в это свойство допускается только один раз. Еще три значения могут быть возвращены этим свойством: `textarea` — область редактирования `<TEXTAREA>`, `select-one` — список `<SELECT>` и `select-multiple` — список `<SELECT>` с установленным атрибутом `MULTIPLE` (или свойством `multiple` равным `true`).

Поддерживается IE начиная с 3.02, для `<SELECT>` — начиная с 4.0. Поддерживается N начиная с 3.0 и только для чтения.

typeDetail

Возвращает дополнительные сведения о типе содержимого объекта `selection`.

```
selection.typeDetail;
```

Всегда возвращается `undefined`. Расширения Web-обозревателя и внедренные объекты могут предусматривать свои значения для этого свойства.

Поддерживается IE начиная с 5.5.

unicodeBidi

Задает или возвращает поведение встроенных элементов при изменении направления письма с помощью свойства `direction`.

```
{Объект стиля}.unicodeBidi[ = "normal|embed|bidi-override"];
```

Доступны три значения:

- `normal` (значение по умолчанию) — меняет направление письма и у родителя;
- `embed` — меняет направление письма только у встроенного элемента;
- `bidi-override` — аналогично `embed` за тем исключением, что направление письма меняется согласно значению свойства `direction`, независимо от локальных установок Web-обозревателя.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.0.

uniqueID

Возвращает автоматически сгенерированный уникальный идентификатор элемента страницы.

```
{Объект}.uniqueID;
```

При первом обращении к этому свойству генерируется уникальный идентификатор и присваивается свойству `id` элемента. При последующих обращениях возвращается именно он.

Поддерживается IE начиная с 5.0.

units

Задает или возвращает единицу измерения ширины и высоты внедренного объекта `<EMBED>`.

```
{Внедренный объект}.units[ = "px|em"];
```

Доступны два значения:

- `px` — задает пикселы в качестве единицы измерения;
- `em` — задает в качестве единицы измерения ширину и высоту символа текущего шрифта.

Поддерживается IE начиная с 3.02.

updateInterval

Задает или возвращает интервал между обновлениями окна Web-обозревателя в миллисекундах.

```
screen.updateInterval[ = {Интервал}];
```

Каждые `{updateInterval}` миллисекунд изображение будет извлекаться из экранного буфера и помещаться в окно Web-обозревателя на экране. Значение `0` отключает обновление окна, что может вызвать, скажем, неправильное отображение анимации.

Поддерживается IE начиная с 4.0.

URL

Задает или возвращает интернет-адрес текущей Web-страницы. Аналогично свойству `location.href`.

```
document.URL[ = "{Интернет-адрес документа}"];
```

В N доступно только для чтения.

Поддерживается IE начиная с 4.0 и N начиная с 2.0.

URLUnencoded

Возвращает полностью декодированный интернет-адрес текущей Web-страницы.

```
document.URLUnencoded;
```

Поскольку в интернет-адресе не допускается употреблять некоторые символы (например, пробелы), они кодируются с использованием их числовых кодов. Это свойство возвращает интернет-адрес текущей страницы без всякой кодировки, т. е. таким, словно бы в интернет-адресе можно было применять любые символы. Свойство доступно только для чтения.

Поддерживается IE начиная с 5.5.

urn

Задает или возвращает имя сетевого ресурса (Uniform Resource Name).

```
{Объект гиперссылки}.urn[ = "{Имя сетевого ресурса}"];
```

Поддерживается IE начиная с 4.0 для `<A>` и начиная с 5.5 для `namespace`.

useMap

Задает или возвращает имя (и, возможно, интернет-адрес) списка "горячих" областей для карты-изображения.

```
{Объект изображения}.useMap[ = "{Интернет-адрес и имя списка "горячих"
↳областей}"];
```

Поддерживается IE начиная с 3.02.

userAgent

Возвращает строку, идентифицирующую название и версию программы Web-обозревателя, сведения о совместимости и операционной системе клиента.

```
navigator.userAgent;
```

Доступно только для чтения.

Поддерживается IE начиная с 3.02 для `navigator` и начиная с 4.0 для `clientInformation`. Поддерживается N начиная с 2.0.

userLanguage

Возвращает строковое обозначение языка, заданного в региональных настройках операционной системы клиента. Коды языков перечислены в приложении 1.

```
navigator.userLanguage;
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0.

vAlign (<CAPTION>)

Задает или возвращает местонахождение заголовка таблицы: вверху или внизу.

```
{Объект заголовка таблицы}.vAlign[ = "top|bottom"];
```

Доступны два значения:

- top — помещает заголовок вверху таблицы (значение по умолчанию);
- bottom — внизу.

Поддерживается IE начиная с 4.0.

vAlign (элементы таблицы)

Задает или возвращает вертикальное выравнивание текста в таблице и ее элементах.

```
{Объект}.vAlign[ = "middle|baseline|bottom|top"];
```

Доступны четыре значения:

- middle — выравнивает текст посередине элемента таблицы (значение по умолчанию);
- baseline — выравнивает базовую линию первой строки по базовой линии элемента таблицы;
- bottom — выравнивает текст по низу элемента таблицы;
- top — по верху.

Поддерживается IE начиная с 3.02 для <TD> и <TR> и начиная с 4.0 для <COL>, <COLGROUP>, <TBODY>, <TFOOT>, <TH> и <THEAD>.

value (<LI>)

Задает или возвращает номер позиции нумерованного списка.

```
{Объект позиции списка}.value[ = "{Номер}"];
```

Поддерживается IE начиная с 3.02.

value (<OPTION>)

Задает или возвращает значение, которое будет послано формой при выборе текущей позиции списка.

```
{Объект позиции списка}.value[ = "{Значение для передачи серверной
↳программе}"];
```

По умолчанию, если атрибут VALUE не указан, форма передает серверной программе значение, помещенное в теге <OPTION>. Если атрибут VALUE указан, форма передаст его значение.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

value (<SELECT>)

Задает или возвращает значение, которое будет послано формой.

```
{Объект списка}.value[ = "{Значение для передачи серверной программе}"];
```

Поддерживается IE начиная с 3.02.

value (кнопки)

Задает или возвращает надпись, помещаемую на кнопке.

```
{Объект кнопки}.value[ = "{Надпись}"];
```

Надпись, помещаемая на кнопке по умолчанию, зависит от программы Web-обозревателя.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

value (поле ввода имени файла)

Возвращает введенное имя файла.

```
{Объект}.value[ = "{Имя файла}"];
```

Доступно только для чтения.

Поддерживается IE начиная с 4.0 и N начиная с 2.0.

value (скрытое поле)

Задает или возвращает значение, посылаемое серверной программе скрытым полем.

```
{Объект скрытого поля}.value[ ="{Значение}"]
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

value (текстовые поля и области редактирования)

Задает или возвращает значение, введенное в текстовое поле или область редактирования.

```
{Объект}.value[ = "{Значение}"];
```

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

value (флажки и радиокнопки)

Задает или возвращает значение, которое форма отправит серверной программе, если пользователь включит флажок или радиокнопку.

```
{Объект}.value[ ="{Значение}"];
```

По умолчанию для флажка посылается значение on, если он выбран. Для радиокнопки значение по умолчанию отсутствует.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

vcard_name

Задает или возвращает значение одного из свойств объекта vCard (сведений о пользователе) для использования во всплывающем списке автозаполнения полей ввода.

```
{Объект поля ввода}.vcard_name[ = "{Свойство объекта vCard}"];
```

Все доступные для указания в этом атрибуте свойства перечислены в табл. П1.3.

Поддерживается IE начиная с 5.0.

verticalAlign

Задает или возвращает вертикальное выравнивание элемента страницы внутри родителя.

```
{Объект стиля}.verticalAlign[ = "auto|baseline|sub|super|top|text-top|
↳middle|bottom|text-bottom"];
```

Доступны девять значений:

- auto — выравнивает элемент страницы согласно значению атрибута layout-flow;
- baseline — (значение по умолчанию) задает выравнивание базовой линии элемента страницы по базовой линии родителя;
- sub — превращает текст в нижний индекс;
- super — превращает текст в верхний индекс;
- top — выравнивает верх элемента страницы по самому верху родителя;
- text-top — выравнивает верх текста элемента страницы по верху текста родителя;
- middle — выравнивает центр элемента страницы по центру родителя;
- bottom — выравнивает низ элемента страницы по низу родителя;

- `text-bottom` — выравнивает низ текста элемента страницы по низу текста родителя.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

viewInheritStyle

Задает или возвращает признак: будут ли элементы Web-страницы наследовать стили, заданные для всей страницы, или нет.

```
defaults.viewInheritStyle[ = true|false];
```

По умолчанию все элементы Web-страницы наследуют стили, заданные в ее заголовке (значение `true`). Чтобы отменить это, дайте свойству значение `false`.

Поддерживается IE начиная с 5.5.

viewLink

Задает или возвращает ссылку на объект `document`, предоставляющий содержимое для главного элемента. Используется для создания и управления поведениями.

```
defaults.viewLink[ = {Объект}];
```

Поддерживается IE начиная с 5.5.

viewMasterTab

Задает или возвращает признак: будет ли главный элемент Web-страницы включаться в последовательность обхода первоначальной страницы или нет. Используется для создания и управления поведениями.

```
defaults.viewMasterTab[ = true|false];
```

По умолчанию главный элемент Web-страницы включается в последовательность обхода по нажатии клавиши <Tab> (значение `true`). Чтобы отменить это, дайте свойству значение `false`.

Поддерживается IE начиная с 5.5.

visibility (слой)

Задает или возвращает признак слоя: будет ли он видимым.

```
{Объект слоя}.visibility[ = "inherit|show|hide"];
```

Доступны три значения:

- `inherit` (значение по умолчанию) — заставляет слой вести себя так же, как родительский слой (наследовать "видимость");
- `visible` — делает слой видимым;
- `hidden` — невидимым.

Поддерживается N начиная с 4.0.

visibility (CSS)

Задает или возвращает признак элемента страницы: будет ли он видимым.

```
{Объект стиля}.visibility[ = "inherit|visible|hidden"];
```

Доступны три значения:

- `inherit` (значение по умолчанию) — заставляет элемент страницы вести себя так же, как родитель (наследовать "видимость");
- `visible` — делает элемент страницы видимым;
- `hidden` — невидимым.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

vLink

Задает или возвращает цвет посещенных гиперссылок в документе.

```
body.vLink[ = "{Цвет}"];
```

Поддерживается IE начиная с 3.02.

vlinkColor

Задает или возвращает цвет посещенных гиперссылок в документе.

```
document.vlinkColor[ = "{Цвет}"];
```

Значение по умолчанию `#800080`.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

volume

Задает или возвращает громкость воспроизведения фонового звука `<BGSOUND>`.

```
{Объект}.volume[ = "{Громкость}"];
```

Значение громкости может быть от −10 000 до 0 (полная громкость).

Поддерживается IE начиная с 4.0.

vspace

Задает или возвращает расстояние по вертикали от элемента страницы до окружающего его текста.

```
{Объект}.vspace[ = "{Вертикальный отступ}"];
```

В N доступно только для чтения.

Поддерживается IE начиная с 3.02 для <APPLET>, <IMG>, <INPUT TYPE="image">, <MARQUEE> и <OBJECT> и начиная с 4.0 для <IFRAME>. Поддерживается N начиная с 3.0.

which

В зависимости от наступившего события, задает или возвращает номер нажатой кнопки мыши или ASCII-код нажатой клавиши клавиатуры.

```
event.which[ = 1|2|3|{Код клавиши}];
```

Здесь 1 обозначает левую кнопку, 2 — среднюю, 3 — правую.

Поддерживается N начиная с 4.0.

whiteSpace

Задает или возвращает признак: будут ли строки текста, содержащегося в элементе страницы, автоматически "разрываться", если они не помещаются в нем по ширине.

```
{Объект стиля}.whiteSpace[ = "normal|nowrap|pre"];
```

Доступны три значения:

- normal (значение по умолчанию) — включает автоматической "разрыв" длинных строк;
- nowrap — отключает автоматической "разрыв" строк. Чтобы "разорвать" строку вручную, вставьте в нужном месте тег <BR>;
- pre — не поддерживается.

Поддерживается IE начиная с 5.5. Поддерживается N начиная с 4.0.

width

Задает или возвращает ширину элемента страницы либо в пикселах, либо в процентах от доступного пространства.

```
{Объект}.width[ = "{Ширина}"];
```

В N доступно только для чтения.

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

width (document)

Возвращает ширину документа в пикселах.

```
document.width;
```

Доступно только для чтения.

Поддерживается N начиная с 4.0.

width (event)

Задает или возвращает ширину окна или фрейма в пикселах.

```
event.width[ = {X}];
```

Поддерживается N начиная с 4.0.

width (screen)

Возвращает полную ширину экрана компьютера в пикселах.

```
screen.width;
```

Доступно только для чтения.

Поддерживается IE и N начиная с 4.0.

width (CSS)

Задает или возвращает ширину свободно позиционированного элемента.

```
{Объект стиля}.width[ = "auto|{X}|{X}%"];
```

Ширина может быть задана как абсолютной величиной, так и процентом от ширины родителя. Предопределенное значение `auto` заставляет Web-обозреватель устанавливать ширину элемента самостоятельно. Значение по умолчанию `auto`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE и N начиная с 4.0.

window

Возвращает ссылку на объект `window`, представляющий содержимое слоя.

```
{Объект слоя}.window;
```

Доступно только для чтения.

Поддерживается N начиная с 4.0.

wordBreak

Задает или возвращает признак: будут ли строки текстов, содержащих фрагменты на разных языках, "разрываться" по словам, а не только по пробелам, или нет.

```
{Объект стиля}.workBreak[ = "normal|break-all|keep-all"];
```

Доступны три значения:

- `normal` (значение по умолчанию) — разрешает строкам "разрываться" по слову;
- `break-all` — предназначено для текстов на азиатских языках с небольшими иноязычными фрагментами;
- `keep-all` — предназначено для текстов, включающих фрагменты на иероглифических языках.

Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.0.

wordSpacing

Задает или возвращает интервал, добавляемый между словами в тексте.

```
{Объект стиля}.wordSpacing[ = "normal|{Величина}"];
```

Значение этого свойства может быть задано либо абсолютной величиной в одной из поддерживаемых CSS единиц измерения, либо предопределенным значением `normal`, задающим стандартную величину расстояния между символами. Значение по умолчанию `normal`.

Поддерживается IE начиная с 4.01.

wordWrap

Задает или возвращает признак: будет ли строка, выходящая за границы элемента страницы и не содержащая пробелов, переноситься по слову.

```
{Объект стиля}.wordWrap[ = "normal|break-word"];
```

Доступны два значения: `normal` (значение по умолчанию) запрещает строкам переноситься по слову, а `break-word` разрешает. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.5.

wrap (<PRE>)

Задает или возвращает признак: будут ли строки текста с заданным форматированием в теге <PRE> переноситься, если ширины элемента страницы не хватит, чтобы отобразить их полностью.

```
{Объект}.wrap[ = true|false];
```

По умолчанию строки текста с заданным форматированием не переносятся (значение false). Если вы хотите, чтобы они переносились, дайте этому свойству значение true.

Поддерживается IE начиная с 5.5.

wrap (<TEXTAREA>)

Задает или возвращает способ обработки переноса строк в области редактирования.

```
{Объект}.wrap[ = "soft|hard|off"];
```

Доступны три значения:

- soft — включает перенос строк; при этом результирующий текст не будет содержать символы возврата каретки (значение по умолчанию);
- hard — включает перенос строк и заставляет область редактирования вставлять в нужные места результирующего текста символы возврата каретки;
- off — отключает перенос строк.

Поддерживается IE начиная с 4.0.

writingMode

Задает или возвращает направление строк текста: горизонтальное или вертикальное.

```
{Объект стиля}.writingMode[ = "lr-tb|tb-rl"];
```

Доступны два значения. lr-tb (значение по умолчанию) задает обычное горизонтальное расположение строк текста; текст пишется слева направо и сверху вниз. tb-rl поворачивает текст на 90° по часовой стрелке; при этом он будет писаться сверху вниз и справа налево. Свойство доступно для чтения и записи, в случае объекта currentStyle — только для чтения.

Поддерживается IE начиная с 5.5.

x (<A>)

Возвращает горизонтальную координату левой границы "якоря" или гиперссылки в пикселах.

```
{Объект}.x;
```

Поддерживается N начиная с 2.0 для гиперссылок и начиная с 4.0 для "якорей".

x (event)

Задает или возвращает горизонтальную координату курсора мыши относительно родителя в пикселах.

```
event.x[ = {X}];
```

Поддерживается IE начиная с 4.0 только для чтения и начиная с 5.0 для чтения и записи. Поддерживается N начиная с 4.0.

x (слой)

Задает или возвращает горизонтальную координату левой стороны слоя относительно родителя в пикселах. Аналогично `left`.

```
{Объект слоя}.x[ = {X}];
```

Поддерживается N начиная с 4.0.

XMLDocument

Возвращает ссылку на объект XML-документа.

```
document.XMLDocument;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.0.

XSLDocument

Возвращает ссылку на объект таблицы стилей XSL.

```
document.XSLDocument;
```

Доступно только для чтения.

Поддерживается IE начиная с 5.0.

y (<A>)

Возвращает вертикальную координату верхней границы "якоря" или гиперссылки в пикселах.

```
{Объект}.y;
```

Поддерживается N начиная с 2.0 для гиперссылок и начиная с 4.0 для "якорей".

y (event)

Задает или возвращает вертикальную координату курсора мыши относительно родителя в пикселах.

```
event.y[ = {Y}];
```

Поддерживается IE начиная с 4.0 только для чтения и начиная с 5.0 для чтения и записи. Поддерживается N начиная с 4.0.

y (слой)

Задает или возвращает вертикальную координату верхней стороны слоя относительно родителя в пикселах. Аналогично `top`.

```
{Объект слоя}.y[ = {Y}];
```

Поддерживается N начиная с 4.0.

zIndex (слой)

Задает или возвращает порядок перекрытия слоями друг друга.

```
{Объект слоя}.zIndex[ = {Порядок перекрытия}];
```

Порядок перекрытия задается целыми положительными числами, начиная с нуля. При этом элементы с большим значением этого атрибута будут перекрывать элементы с меньшим значением.

Поддерживается N начиная с 4.0.

zIndex (CSS)

Задает или возвращает порядок перекрытия свободно позиционированными объектами друг друга.

```
{Объект стиля}.zIndex[ = "auto|{Порядок перекрытия}"];
```

Порядок перекрытия задается положительным или отрицательным целым числом. При этом элементы с большим значением этого атрибута будут перекрывать элементы с меньшим значением. Предопределенное значение `auto` задает порядок перекрытия по умолчанию, когда элементы, определенные в HTML-коде раньше, перекрываются определенными позже. Значение по умолчанию `auto`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 4.0.

zoom

Задает или возвращает масштаб отображения элемента страницы.

```
{Объект стиля}.zoom[ = "normal|{Масштаб}|{Масштаб}%"];
```

Масштаб может быть задан как числом с плавающей точкой, обозначающим степень увеличения или уменьшения, так и процентной величиной. Предопределенное значение `normal` задает масштаб 1.0 или 100%. Значение по умолчанию `normal`. Свойство доступно для чтения и записи, в случае объекта `currentStyle` — только для чтения.

Поддерживается IE начиная с 5.5.

Методы

Метод — это операция или функция, связанная с объектом документа и позволяющая манипулировать его данными. Методы перечислены в алфавитном порядке. Приводится формальное описание метода, поясняются его семантика, параметры и возвращаемое значение. Дается описание поддержки основными Web-обозревателями.

add

Добавляет новый объект в коллекцию.

```
{Коллекция}.add({Объект}[, {Индекс}]);
```

Метод принимает два параметра. Первый из них задает ссылку на добавляемый в коллекцию объект, необязательный второй — индекс в коллекции, который будет ему соответствовать. Если второй параметр не задан, будет использован следующий свободный индекс в коллекции.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

add(namespace)

Создает новый объект `namespace` и добавляет его в коллекцию `namespaces`.

```
namespaces.add("{Имя}", "{Интернет-адрес файла определения объекта}"
↳[, "{Интернет-адрес файла поведения}"]);
```

Метод принимает три параметра. Первый из них задает имя создаваемого объекта, второй — интернет-адрес файла определения объекта, необязательный третий — интернет-адрес файла поведения, которое будет использоваться с этим объектом.

Метод возвращает ссылку на созданный объект `namespace`.

Поддерживается IE начиная с 5.5.

addBehavior

Добавляет к текущему элементу страницы поведение.

```
{Объект}.addBehavior("{Интернет-адрес файла поведения}|
↳#{Имя элемента ActiveX, реализующего это поведение}|
↳#default#{Имя встроенного поведения}"[, {Интерфейс}]);
```

Метод принимает два параметра. Первый из них — интернет-адрес файла поведения, имя поведения по умолчанию, реализованного Web-обозревателем, или имя объекта `<OBJECT>`, реализующего это представление. Необязательный второй параметр задает ссылку на интерфейс `IElementBehaviorFactory` поведения.

Метод возвращает целочисленный идентификатор добавленного поведения, который может потом использовать, например, для его удаления.

Поддерживается IE начиная с 5.0.

AddChannel

Выводит на экран диалоговое окно, позволяющее пользователю добавить канал или отредактировать его интернет-адрес, если такой канал уже добавлен.

```
external.AddChannel("{Интернет-адрес}");
```

Метод принимает один параметр — интернет-адрес файла CDF, содержащий определение канала.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

AddDesktopComponent

Добавляет на активный рабочий стол новый компонент.

```
external.AddDesktopComponent("{Интернет-адрес}", "image|website"
↳[, {Слева}][, {Сверху}][, {Ширина}][, {Высота}]);
```

Метод принимает шесть параметров. Первый из них — интернет-адрес Web-сайта или файла изображения. Второй может принимать два возможных значения в зависимости от типа отображаемого содержимого: `image` определяет изображение, а `website` — Web-сайт. Также могут приниматься еще четыре необязательных параметра, задающие для компонента активного рабочего стола горизонтальную координату левой стороны, вертикальную координату верхней стороны, ширину и высоту соответственно.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

AddFavorite

Выводит на экран диалоговое окно, позволяющее пользователю добавить интернет-адрес в список **Избранное**.

```
external.AddFavorite("{Интернет-адрес}"[, "{Заголовок}"]);
```

Метод принимает два параметра. Первый из них — собственно интернет-адрес. Второй — необязательный — задает заголовок, под которым этот адрес появится в списке **Избранное**.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

addImport

Добавляет таблицу стилей в коллекцию `imports` текущей таблицы стилей.

```
{Объект таблицы стилей}.addImport("{Интернет-адрес}"[, {Индекс}]);
```

Метод принимает два параметра. Первый из них задает интернет-адрес добавляемой таблицы стилей, необязательный второй — индекс в коллекции, который будет ей соответствовать. Если второй параметр не задан, будет использован следующий свободный индекс в коллекции.

Метод возвращает числовой индекс, который займет в коллекции вновь добавленная таблица стилей.

Поддерживается IE начиная с 4.0.

addPageRule

Добавляет новое правило для печатных страниц в текущую таблицу стилей.

```
{Объект таблицы стилей}.addPageRule("{Селектор}", "{Определение стиля}"
↳[, {Индекс}]);
```

Метод принимает три параметра. Первый из них задает наименование селектора добавляемого правила: `:first` для первой страницы, `:left` для четной и `:right` для нечетной. Второй параметр задает определение стиля для добавляемого правила. Необязательный третий параметр задает индекс в коллекции, который будет ей соответствовать. Если третий параметр не задан, будет использован следующий свободный индекс в коллекции.

Возвращаемое значение метода не используется.

Поддерживается IE начиная с 5.5.

addReadRequest

Добавляет запрос в очередь запросов объекта `userProfile`.

```
userProfile.addReadRequest("{Свойство}");
```

Метод принимает единственный параметр — имя одного из свойств vCard, перечисленных в табл. П1.3.

Метод возвращает true, если запрос был успешно добавлен в очередь, и false — в противном случае.

Поддерживается IE начиная с 4.0.

addRule

Добавляет новое правило в текущую таблицу стилей.

```
{Объект таблицы стилей}.addRule("{Селектор}", "{Определение стиля}"
↳[, {Индекс}]);
```

Метод принимает три параметра. Первый из них задает наименование селектора добавляемого правила: оно может иметь вид "H1" или "H1 B". Второй параметр задает определение стиля для добавляемого правила. Необязательный третий параметр задает индекс в коллекции, который будет ей соответствовать. Если третий параметр не задан, или его значение равно -1, будет использован следующий свободный индекс в коллекции.

Возвращаемое значение метода не используется.

Поддерживается IE начиная с 4.0.

alert

Выводит на экран диалоговое окно предупреждения.

```
window.alert("{Текст предупреждения}");
```

Метод принимает единственный параметр — текст предупреждения.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

appendChild

Помещает вновь созданный элемент на Web-страницу.

```
{Объект}.appendChild({Объект});
```

Метод принимает единственный параметр — ссылку на добавляемый элемент.

Метод возвращает ссылку на добавленный элемент.

Поддерживается IE начиная с 5.0.

applyElement

Делает заданный элемент страницы либо родителем, либо потомком текущего элемента.

```
{Объект}.applyElement({Объект}[, "outside|inside"]);
```

Метод принимает два параметра. Первый из них задает ссылку на элемент страницы, который будет либо родителем, либо потомком текущего элемента. Второй — необязательный — задает, чем же, собственно, будет переданный в первом параметре элемент страницы для текущего элемента. Значение `outside` делает его родителем текущего элемента (это значение по умолчанию), а значение `inside` — потомком.

Метод возвращает ссылку на переданный в первом параметре элемент.

Поддерживается IE начиная с 5.0.

assign

Загружает новую Web-страницу в текущее окно Web-обозревателя.

```
location.assign("{Интернет-адрес}");
```

Метод принимает единственный параметр — интернет-адрес новой Web-страницы.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

atob

Декодирует строку текста, закодированную алгоритмом base-64.

```
window.atob("{Закодированная строка}");
```

Метод принимает единственный параметр — закодированную строку.

Метод возвращает раскодированную строку.

Поддерживается N начиная с 4.0.

attachEvent

Привязывает к обработчику события функцию, которая будет вызвана после наступления этого события. К одному обработчику могут быть привязаны несколько функций.

```
{Объект}.attachEvent("{Событие}", {Имя функции});
```

Метод принимает два параметра: первый задает имя события, а второй — имя функции, которая будет вызвана при его наступлении.

Метод возвращает `true`, если функция была успешно привязана к обработчику, и `false` — в противном случае.

Поддерживается IE начиная с 5.0.

AutoCompleteSaveForm

Сохраняет содержимое полей ввода формы в хранилище данных для последующего автозаполнения.

```
external.AutoCompleteSaveForm({Форма});
```

Метод принимает единственный параметр — ссылку на объект формы, данные которого будут сохранены.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 5.0.

AutoScan

Принимает строку, являющуюся неполным интернет-адресом Web-сайта, дополняет его с помощью встроенной функции автоввода и пытается соединиться с сайтом.

```
external.AutoScan("{Неполный интернет-адрес}", "{Интернет-адрес
↳страницы сообщения об ошибке}"[, "{Имя окна или фрейма}"]);
```

Метод принимает три параметра. Первый из них задает неполный интернет-адрес. Второй — интернет-адрес Web-страницы, отображаемой в случае неудачи (например, если такого интернет-адреса не существует). Необязательный третий позволяет задать имя окна или фрейма, в котором будут выводиться результаты.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 5.0.

back

Загружает в окно Web-обозревателя предыдущую Web-страницу из списка "истории".

```
history.back();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 3.02. Поддерживается N начиная с 2.0 для `history` и начиная с 4.0 для `window`.

blur

Заставляет элемент страницы потерять фокус ввода.

```
{Объект}.blur();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 4.0 для гиперссылок, элементов управления, внедренных объектов, фреймов, таблиц, <DIV>, <SPAN> и window и начиная с 5.0 — для остальных элементов. Поддерживается N начиная с 2.0.

borderWidths

Задает значения толщины всех границ элемента страницы в один прием.

```
{Объект стиля}.borderWidths("{Верхняя}", "{Правая}", "{Нижняя}",
↳"{Левая}");
```

Метод принимает четыре строковых параметра, задающие значения толщины верхней, правой, нижней и левой границ соответственно.

Метод не возвращает никакого значения.

Поддерживается N начиная с 4.0.

captureEvents

Позволяет перехватить события заданного типа, наступившие в других элементах страницы.

```
document.captureEvents("{Список имен событий}");
```

Метод принимает единственный параметр — список строковых имен событий, разделенных вертикальной чертой.

Метод не возвращает никакого значения.

Поддерживается N начиная с 4.0.

clear(document)

Удаляет все содержимое текущей Web-страницы.

```
document.clear();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 3.02.

clear(selection)

Удаляет все содержимое выделенного фрагмента текущей Web-страницы.

```
selection.clear();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 4.0.

clearAttributes

Удаляет все атрибуты и их значения текущего элемента страницы.

```
{Объект}.clearAttributes();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 5.0.

clearData

Удаляет данные в каком-либо формате в объектах `clipboardData` или `dataTransfer`.

```
{Объект}.clearData(["{Список форматов, разделенных пробелами}"]);
```

Метод принимает единственный необязательный параметр, задающий формат удаляемых данных. Он может иметь следующие значения:

- `Text` — текстовые данные;
- `URL` — интернет-адрес;
- `File` — файл;
- `HTML` — код HTML;
- `Image` — графическое изображение.

Если параметр не задан, удаляются все данные.

Поддерживается IE начиная с 5.0.

clearInterval

Удаляет интервальный таймер, определенный методом `setInterval`.

```
window.clearInterval({Идентификатор таймера});
```

Метод принимает один параметр — идентификатор интервального таймера, возвращенный методом `setInterval`.

Метод не возвращает никакого значения.

Поддерживается IE и N начиная с 4.0.

clearRequest

Очищает очередь запросов объекта `userProfile`.

```
userProfile.clearRequest;
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 4.0.

clearTimeout

Удаляет тайм-аут, определенный методом `setTimeout`.

```
window.clearTimeout({Идентификатор таймаута});
```

Метод принимает один параметр — идентификатор тайм-аута, возвращенный методом `setTimeout`.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

click

Имитирует щелчок левой кнопкой мыши по элементу страницы.

```
{Объект}.click();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 3.02 для `<INPUT TYPE="button">` и начиная с 4.0 — для остальных элементов. Поддерживается N начиная с 2.0.

cloneNode

Создает копию текущего элемента страницы.

```
{Объект}.cloneNode([true|false]);
```

Метод принимает единственный необязательный параметр, значение `false` которого запрещает копирование вместе с элементом страницы его содержимого, а `true` — разрешает. Если параметр не указан, используется значение `false`.

Метод возвращает ссылку на вновь созданную копию элемента страницы.

Поддерживается IE начиная с 5.0.

close (document)

Заставляет Web-страницу немедленно обновить свое содержимое после использования методов `write`.

```
document.close();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 4.0 и N начиная с 2.0.

close (window)

Закрывает текущее окно Web-обозревателя.

```
window.close();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

collapse

Сжимает содержимое текущего объекта `TextRange` в точку и помещает его в начале или конце исходного содержимого объекта.

```
{Объект TextRange}.collapse([true|false]);
```

Метод принимает единственный необязательный параметр, значение `true` которого перемещает новое содержимое объекта в начало исходного, а `false` — в конец. Если параметр не указан, используется значение `true`.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

compareEndPoints

Сравнивает границы текущего объекта `TextRange` с границами другого.

```
{Объект TextRange}.compareEndPoints(StartToEnd|StartToStart|EndToStart|
↳EndToEnd, {Второй сравниваемый объект TextRange});
```

Метод принимает два параметра. Первый из них задает, какие границы объектов будут сравниваться. Возможны четыре значения:

- `StartToEnd` — начальная граница текущего объекта сравнивается с конечной границей второго объекта;
- `StartToStart` — начальная с начальной;
- `EndToStart` — конечная с начальной;
- `EndToEnd` — конечная с конечной.

Второй параметр задает ссылку на второй сравниваемый объект `TextRange`.

Метод возвращает одно из трех значений:

- `-1` — граница текущего объекта расположена левее (ближе к началу документа), чем граница второго объекта;
- `0` — они совпадают;
- `1` — граница второго объекта расположена правее, чем граница текущего.

Поддерживается IE начиная с 4.0.

componentFromPoint

Возвращает ссылку на фрагмент элемента страницы, имеющий заданные координаты.

```
{Объект}.componentFromPoint({X}, {Y});
```

Метод принимает два параметра, задающие в пикселах горизонтальную и вертикальную координаты соответственно.

Метод возвращает одно из текстовых значений, перечисленных в табл. П3.1.

Таблица П3.1. *Возвращаемые значения метода* `componentFromPoint`

Значение	Описание
Пустая строка	Один из дочерних элементов текущего элемента страницы
`outside`	Элемент страницы, расположенный за границами текущего элемента
`scrollbarDown`	Нижняя кнопка-стрелка полосы прокрутки
`scrollbarHThumb`	Ползунок горизонтальной полосы прокрутки
`scrollbarLeft`	Левая кнопка-стрелка полосы прокрутки
`scrollbarPageDown`	Область, расположенная ниже ползунка, на вертикальной полосе прокрутки
`scrollbarPageLeft`	Область, расположенная левее ползунка, на горизонтальной полосе прокрутки
`scrollbarPageRight`	Область, расположенная правее ползунка, на горизонтальной полосе прокрутки
`scrollbarPageUp`	Область, расположенная выше ползунка, на вертикальной полосе прокрутки
`scrollbarRight`	Правая кнопка-стрелка полосы прокрутки
`scrollbarUp`	Верхняя кнопка-стрелка полосы прокрутки
`scrollbarVThumb`	Ползунок вертикальной полосы прокрутки
`handleBottom`	Ползунок изменения размера на нижней границе
`handleBottomLeft`	Ползунок изменения размера на нижнем левом углу
`handleBottomRight`	Ползунок изменения размера на нижнем правом углу
`handleLeft`	Ползунок изменения размера на левой границе
`handleRight`	Ползунок изменения размера на правой границе
`handleTop`	Ползунок изменения размера на верхней границе

Таблица П3.1 (окончание)

Значение	Описание
handleTopLeft	Ползунок изменения размера на верхнем левом углу
handleTopRight	Ползунок изменения размера на верхнем правом углу

Поддерживается IE начиная с 5.0.

confirm

Выводит на экран диалоговое окно предупреждения с кнопками **OK** и **Отмена**.

```
window.confirm(["{Текст предупреждения}"]);
```

Метод принимает единственный необязательный параметр — текст предупреждения. Если он не задан, окно предупреждения не будет содержать текста.

Метод возвращает `true`, если пользователь нажал кнопку **OK**, и `false`, если кнопку **Отмена**.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

contains

Проверяет, является ли данный элемент страницы потомком текущего элемента.

```
{Объект}.contains({Объект});
```

Метод принимает единственный параметр — ссылку на проверяемый элемент страницы.

Метод возвращает `true`, если переданный в параметре элемент страницы является потомком текущего элемента, и `false` — в противном случае.

Поддерживается IE начиная с 4.0.

createCaption

Создает пустой заголовок таблицы `<CAPTION>`.

```
{Объект таблицы}.createCaption();
```

Метод не принимает параметров.

Метод возвращает ссылку на созданный элемент заголовка таблицы.

Поддерживается IE начиная с 4.0.

createControlRange

Создает коллекцию controlRange.

```
body.createControlRange();
```

Метод не принимает параметров.

Метод возвращает ссылку на созданную коллекцию.

Поддерживается IE начиная с 5.0.

createElement

Создает элемент страницы.

```
document.createElement("{Тег}");
```

Метод принимает единственный параметр — имя тега для создания элемента страницы. Могут быть заданы любые теги, кроме <FRAME> и <IFRAME>.

Метод возвращает ссылку на вновь созданный элемент.

Поддерживается IE начиная с 4.0 для <AREA>, <IMG> и <OPTION> и начиная с 5.0 — для всех остальных.

createEventObject

Создает новый объект event для передачи контекста события, используя метод fireEvent.

```
document.createEventObject([{Существующий объект event}]);
```

Метод принимает единственный необязательный параметр — ссылку на уже существующий объект event, на котором будет основан вновь создаваемый объект.

Метод возвращает ссылку на вновь созданный объект event.

Поддерживается IE начиная с 5.5.

createPopup

Создает всплывающее окно.

```
window.createPopup();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 5.5.

createRange

Создает объект TextRange из выделенного фрагмента Web-страницы.

```
selection.createRange();
```

Метод не принимает параметров.

Метод возвращает ссылку на созданный объект.

Поддерживается IE начиная с 4.0.

createRangeCollection

Создает коллекцию объектов `TextRange` из всех выделенных фрагментов Web-страницы. Но, т. к. современные версии Web-обозревателей не поддерживают множественное выделение, эта коллекция содержит всего один объект `TextRange`.

```
selection.createRangeCollection();
```

Метод не принимает параметров.

Метод возвращает ссылку на созданную коллекцию.

Поддерживается IE начиная с 5.5.

createStyleSheet

Создает новую таблицу стилей для Web-страницы.

```
document.createStyleSheet(["{Интернет-адрес}"][, {Индекс}]);
```

Метод принимает два необязательных параметра. Первый из них задает интернет-адрес файла таблицы стилей, второй — индекс в коллекции `styleSheets`, который будет ей соответствовать. Если второй параметр не задан, будет использован следующий свободный индекс в коллекции.

Метод возвращает ссылку на вновь созданный объект таблицы стилей.

Поддерживается IE начиная с 4.0.

createTextNode

Создает объект `TextNode`.

```
document.createTextNode(["{Текст}"]);
```

Метод принимает единственный необязательный параметр — строку текста, которая станет содержимым создаваемого объекта.

Метод возвращает ссылку на созданный объект.

Поддерживается IE начиная с 5.0.

createTextRange

Создает объект `TextRange`.

```
{Объект}.createTextRange();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 4.0.

createTFoot

Создает пустой раздел основания таблицы <TFOOT>.

```
{Объект таблицы}.createTFoot();
```

Метод не принимает параметров.

Метод возвращает ссылку на созданный элемент раздела основания таблицы.

Поддерживается IE начиная с 4.0.

createTHead

Создает пустую секцию заголовка таблицы <THEAD>.

```
{Объект таблицы}.createTHead();
```

Метод не принимает параметров.

Метод возвращает ссылку на созданный элемент раздела заголовка таблицы.

Поддерживается IE начиная с 4.0.

deleteCaption

Удаляет заголовок таблицы <CAPTION>.

```
{Объект таблицы}.deleteCaption();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 4.0.

deleteCell

Удаляет ячейку таблицы.

```
{Объект строки таблицы}.deleteCell([{Индекс}]);
```

Метод принимает единственный необязательный параметр — порядковый номер ячейки в коллекции cells. Если параметр пропущен, удаляется последняя ячейка.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

deleteRow

Удаляет строку таблицы.

```
{Объект таблицы или ее секции}.deleteRow([{Индекс}]);
```

Метод принимает единственный необязательный параметр — порядковый номер строки в коллекции rows. Если параметр пропущен, удаляется последняя строка.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

deleteTFoot

Удаляет основание таблицы <TFOOT>.

```
{Объект таблицы}.deleteTFoot();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 4.0.

deleteTHead

Удаляет заголовок таблицы <THEAD>.

```
{Объект таблицы}.deleteTHead();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 4.0.

detachEvent

Открепляет от обработчика события ранее привязанную методом attachEvent функцию.

```
{Объект}.detachEvent("{Событие}", {Имя функции});
```

Метод принимает два параметра: первый задает имя события, а второй — имя функции, которая вызывается при его наступлении.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 5.0.

disableExternalCapture

Отключает перехват внешних событий.

```
window.disableExternalCapture();
```

Метод не принимает параметров и не возвращает никакого значения.

Поддерживается N начиная с 4.0.

doImport

Импортирует в пространство имен поведение.

```
{Объект}.doImport("{Интернет-адрес}");
```

Метод принимает один параметр — интернет-адрес файла поведения.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 5.5.

doReadRequest

Выполняет все запросы, находящиеся в очереди объекта `userProfile`.

```
userProfile.doReadRequest({Код доступа}[, "{Псевдоним}"]
↳[, "{Интернет-адрес}"][, "{Путь}"][, {Дата устаревания}]);
```

Метод принимает до пяти параметров. Первый, единственный обязательный, параметр задает код доступа. Для него доступно тринадцать значений от `0` до `12` в порядке убывания защищенности и приоритета. Второй параметр задает псевдоним для доступа к защищенной информации. Третий, четвертый и пятый параметры задают адрес Web-сайта, путь местонахождения файлов Web-страниц, с которых разрешено получать запросы, и дату устаревания запросов соответственно.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

doScroll

Заставляет содержимое элемента страницы прокручиваться, имитируя щелчки по полосам прокрутки.

```
{Объект}.doScroll(["{Действие}"]);
```

Метод принимает единственный необязательный параметр, задающий действие, которое выполняется над полосой прокрутки. Все доступные значения перечислены в табл. П3.2.

Таблица П3.2. *Значения параметра метода* `doScroll`

Значение	Описание
`scrollbarDown`	Щелчок по нижней кнопке-стрелке. Это значение по умолчанию
`scrollbarHThumb`	Перемещение бегунка горизонтальной полосы прокрутки

Таблица П3.2 (окончание)

Значение	Описание
scrollbarLeft	Щелчок по левой кнопке-стрелке
scrollbarPageDown	Прокрутка на страницу вниз
scrollbarPageLeft	Прокрутка на страницу влево
scrollbarPageRight	Прокрутка на страницу вправо
scrollbarPageUp	Прокрутка на страницу вверх
scrollbarRight	Щелчок по правой кнопке-стрелке
scrollbarUp	Щелчок по верхней кнопке-стрелке
scrollbarVThumb	Перемещение бегунка вертикальной полосы прокрутки
down	То же самое, что scrollbarDown
left	То же самое, что scrollbarLeft
pageDown	То же самое, что scrollbarPageDown
pageLeft	То же самое, что scrollbarPageLeft
pageRight	То же самое, что scrollbarPageRight
pageUp	То же самое, что scrollbarPageUp
right	То же самое, что scrollbarRight
up	То же самое, что scrollbarUp

Поддерживается IE начиная с 5.0.

dragDrop

Инициирует событие ondragstart и начало операции перетаскивания.

```
(Объект).dragDrop();
```

Метод не принимает параметров.

Метод возвращает true, если операция drag-n-drop успешно завершена, и false, если она была прервана в обработчике события ondragstart.

Поддерживается IE начиная с 5.5.

duplicate

Дублирует объект TextRange.

```
(Объект).duplicate();
```

Метод не принимает параметров.

Метод возвращает указатель на копию текущего объекта `TextRange`.

Поддерживается IE начиная с 4.0.

elementFromPoint

Возвращает ссылку на элемент страницы, находящийся по данным координатам.

```
document.elementFromPoint({X}, {Y});
```

Метод принимает два параметра, задающие в пикселах горизонтальную и вертикальную координаты соответственно.

Метод возвращает ссылку на элемент страницы, имеющий заданные координаты.

Поддерживается IE начиная с 4.0.

empty

Убирает выделение с текста Web-страницы.

```
selection.empty();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 4.0.

enableExternalCapture

Разрешает перехват внешних событий.

```
window.enableExternalCapture();
```

Метод не принимает параметров и не возвращает никакого значения.

Поддерживается N начиная с 4.0.

execCommand

Выполняет одну из предопределенных команд над содержимым документа или объекта `TextRange` или `controlRange`.

```
{Объект}.execCommand("{Команда}"[, true|false][, {Параметр команды}]);
```

Метод принимает три параметра. Первый из них задает собственно обозначение команды, которая будет выполнена над содержимым объекта. Список доступных команд приведен далее в этом приложении. Второй — необязательный — параметр задает, будет ли Web-обозреватель запрашивать у пользователя дополнительные параметры для выполняемой команды (значение

true; используется по умолчанию), если она это поддерживает, или нет (значение false). И третий, также необязательный, параметр позволяет задать дополнительные данные для выполняемой команды.

Метод возвращает true, если команда была выполнена благополучно, и false — в противном случае.

Поддерживается IE начиная с 4.0 для document и TextRange и начиная с 5.0 для controlRange.

execScript

Выполняет заданный код на одном из скриптовых языков.

```
window.execScript("{Код скрипта}", "{Язык}");
```

Метод принимает два параметра. Первый из них задает собственно исходный код скрипта в виде текстовой строки. Второй параметр задает обозначение скриптового языка, на котором написан этот скрипт. Для скриптов на JavaScript следует задать значения "JScript" или "javascript".

Метод всегда возвращает null.

Поддерживается IE начиная с 4.0.

expand

Расширяет содержимое объекта TextRange так, чтобы частично входящий в него элемент текста входил в него полностью.

```
{Объект TextRange}.expand("character|word|sentence|textedit");
```

Метод принимает единственный параметр, задающий тип элемента текста:

- character — символ;
- word — слово;
- sentence — предложение;
- textedit — весь абзац.

Метод возвращает true, если содержимое объекта TextRange было расширено успешно, и false — в противном случае.

Поддерживается IE начиная с 4.0.

find

Ищет текст в содержимом Web-страницы, загруженной в текущее окно.

```
window.find(["{Искомый текст}"[, true|false, true|false]]);
```

Метод принимает три необязательных параметра. Первый из них задает собственно искомую строку. Второй позволяет задать, будет ли поиск проводиться с учетом регистра букв (значение `true`) или нет (значение `false`, используемое по умолчанию). Третий задает направление поиска: от начала к концу (значение `false`, используемое по умолчанию) или от конца к началу (значение `true`). Если ни один из параметров не задан, Web-обозреватель отобразит диалоговое окно поиска.

Метод возвращает `true`, если искомый текст был найден, и `false` — в противном случае.

Поддерживается N начиная с 4.0.

findText

Ищет текст в содержимом объекта `TextRange` и в случае удачного поиска изменяет расположение `TextRange` так, чтобы он включал в себя найденный фрагмент.

```
{Объект TextRange}.findText("{Искомый текст}"[, {Номер вхождения}]
↳[, {Параметры}]);
```

Метод принимает три параметра. Первый из них задает собственно искомую строку. Второй — необязательный — номер вхождения искомой строки в тексте. Третий — также необязательный — целое число, задающее параметры поиска. Оно может принимать значения `2` — искать только целые слова, а не фрагменты, `4` — искать с совпадением регистра символов и `6` — сумма `2` и `4`.

Метод возвращает `true`, если искомый текст был найден, и `false` — в противном случае.

Поддерживается IE начиная с 4.0.

fireEvent

Вызывает наступление определенного события для заданного объекта.

```
{Объект}.fireEvent("{Событие}"[, {Объект}]);
```

Метод принимает два параметра. Первый задает название события, которое должно наступить. Второй — необязательный — задает ссылку на элемент страницы, для которого наступит это событие. По умолчанию событие наступает для текущего объекта.

Метод возвращает `true`, если событие успешно произошло, и `false` — в противном случае.

Поддерживается IE начиная с 5.5.

firstPage

Отображает в таблице, привязанной к данным, первую порцию (страницу) этих данных.

```
{Объект таблицы}.firstPage();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 5.0.

focus

Переносит фокус ввода на текущий элемент страницы.

```
{Объект}.focus();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 3.02 для элементов управления, начиная с 4.0 для гиперссылок, внедренных объектов, фреймов, таблиц, <BUTTON>, <SELECT>, <SPAN> и window и начиная с 5.0 для остальных элементов. Поддерживается N начиная с 2.0 для элементов управления и начиная с 3.0 для window.

forward

Загружает в окно Web-обозревателя следующую Web-страницу из списка "истории".

```
history.forward();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 3.02. Поддерживает N начиная с 2.0 для history и начиная с 4.0 для window.

getAdjacentText

Возвращает текст, находящийся возле тегов текущего элемента страницы.

```
{Объект}.getAdjacentText("beforeBegin|afterBegin|beforeEnd|afterEnd");
```

Метод принимает единственный параметр, задающий место, откуда будет взят текст. Доступны четыре значения:

- beforeBegin — текст, расположенный непосредственно перед открывающим тегом;
- afterBegin — после открывающего тега, но перед всем остальным содержимым элемента;

- `beforeEnd` — перед закрывающим тегом, но после всего остального содержимого элемента;
- `afterEnd` — текст, расположенный непосредственно после закрывающего тега.

Поддерживается IE начиная с 5.0.

getAttribute

Возвращает значение атрибута текущего элемента страницы. Если атрибут не поддерживается тегом, возвращает `null`.

```
{Объект}.getAttribute("{Имя атрибута}"[, {Параметры}]);
```

Метод принимает два параметра. Первый из них задает имя атрибута. Необязательный второй позволяет задать дополнительные параметры в виде целого числа. Значение `0` (используется по умолчанию) задает поиск без учета регистра символов и, если атрибут не задан, возвращает значение по умолчанию; значение `1` — поиск с учетом регистра символов; `2` возвращает значение, только если атрибут задан, в противном случае возвращает `null`; `3` является комбинацией значений `1` и `2`.

Поддерживается IE начиная с 4.0.

getAttribute (userProfile)

Возвращает значение именованного атрибута.

```
userProfile.getAttribute("{Имя атрибута}");
```

Метод принимает единственный параметр — имя атрибута. Доступные имена атрибутов перечислены в табл. П1.3.

Поддерживается IE начиная с 4.0.

getBookmark

Возвращает "закладку" — уникальную текстовую строку, позволяющую в дальнейшем идентифицировать данный фрагмент текста.

```
{Объект TextRange}.getBookmark();
```

Метод не принимает параметров.

Поддерживается IE начиная с 4.0.

getBoundingClientRect

Возвращает объект `TextRectangle`, задающий местоположение и размеры текущего элемента страницы.

```
{Объект}.getBoundingClientRect();
```

Метод не принимает параметров.

Поддерживается IE начиная с 5.0.

getClientRects

Возвращает коллекцию объектов `TextRectangle`, задающих местоположение и размеры каждой строки текста текущего элемента страницы.

```
{Объект}.getClientRects();
```

Метод не принимает параметров.

Поддерживается IE начиная с 5.0.

getData

Возвращает данные, хранимые в объекте `clipboardData` или `dataTransfer`.

```
{Объект}.getData("{Формат}");
```

Метод принимает единственный параметр — формат возвращаемых данных. Значение `text` позволит вернуть обычный текст, а значение `URL` — интернет-адрес.

Поддерживается IE начиная с 5.0.

getElementById

Возвращает ссылку на первый элемент страницы с заданным именем (значением атрибута `ID`).

```
document.getElementById("{Имя}");
```

Метод принимает единственный параметр — имя элемента страницы.

Поддерживается IE начиная с 5.0.

getElementByName

Возвращает коллекцию элементов страницы с заданным значением атрибута `NAME`.

```
document.getElementByName("{Имя}");
```

Метод принимает единственный параметр — имя элемента страницы.

Поддерживается IE начиная с 5.0.

getElementByTagName

Возвращает коллекцию элементов страницы с заданным именем тега.

```
{Объект}.getElementByTagName("{Тег}");
```

Метод принимает единственный параметр — тег элементов страницы.

Поддерживается IE начиная с 5.0.

getExpression

Возвращает текст выражения, присвоенного динамическому атрибуту стиля.

```
{Объект стиля}.getExpression("{Имя атрибута}");
```

Метод принимает единственный параметр — имя атрибута стиля.

Поддерживается IE начиная с 5.0.

getSelection

Возвращает строку, содержащую выделенный пользователем текст.

```
document.getSelection();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается N начиная с 4.0.

go (IE)

Загружает в окно Web-обозревателя заданную по номеру Web-страницу из списка "истории".

```
history.go({Номер позиции});
```

Метод принимает единственный параметр — номер позиции в списке "истории".

Метод не возвращает никакого значения.

Поддерживается IE начиная с 3.02.

go (N)

Загружает в окно Web-обозревателя заданную по номеру Web-страницу из списка "истории".

```
history.go("{Интернет-адрес}"|{Приращение});
```

Метод принимает единственный параметр — либо интернет-адрес одной из ранее посещенных Web-страниц, либо целочисленное приращение, сдвиг текущей позиции в списке "истории".

Метод не возвращает никакого значения.

Поддерживается N начиная с 2.0.

handleEvent

Вызывает обработчик заданного события.

```
{Объект}.handleEvent({Событие});
```

Метод принимает единственный параметр — ссылку на объект `event`.

Метод не возвращает никакого значения.

Поддерживается N начиная с 4.0.

hasChildNodes

Возвращает `true`, если элемент страницы имеет дочерние элементы, и `false` — в противном случае.

```
{Объект}.hasChildNodes();
```

Метод не принимает параметров.

Поддерживается IE начиная с 5.0.

hasFocus

Возвращает `true`, если элемент страницы имеет фокус ввода, и `false` — в противном случае.

```
{Объект}.hasFocus();
```

Метод не принимает параметров.

Частично поддерживается IE начиная с 5.5.

hide

Делает всплывающее окно невидимым.

```
{Объект}.hide();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 5.5.

home

Заставляет Web-обозреватель загрузить "домашнюю" Web-страницу.

```
window.home();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается N начиная с 4.0.

ImportExportFavorites

Позволяет импортировать или экспортировать "избранные" закладки.

```
external.ImportExportFavorites(true|false, "{Путь}");
```

Метод принимает два параметра. Первый из них задает операцию: импорт (значение `true`) или экспорт (значение `false`). Второй задает путь, откуда будут импортироваться или куда будут экспортироваться закладки. Если задана пустая строка, Web-обозреватель выдаст на экран стандартный диалог выбора папки.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

inRange

Возвращает `true`, если переданный в параметре объект `TextRange` содержится внутри текущего объекта или равен ему, и `false` — в противном случае.

```
{Объект TextRange}.inRange({Сравниваемый объект TextRange});
```

Метод принимает единственный параметр — ссылку на сравниваемый объект `TextRange`.

Поддерживается IE начиная с 4.0.

insertAdjacentElement

Вставляет вновь созданный элемент страницы внутрь текущего элемента или рядом с ним.

```
{Объект}.insertAdjacentElement("beforeBegin|afterBegin|beforeEnd|
↳afterEnd", {Вставляемый элемент страницы});
```

Метод принимает два параметра. Первый из них задает место, куда будет вставлен элемент страницы. Доступны четыре значения:

- `beforeBegin` — непосредственно перед открывающим тегом текущего элемента страницы;
- `afterBegin` — после открывающего тега, но перед всем остальным содержимым элемента;
- `beforeEnd` — перед закрывающим тегом, но после всего остального содержимого элемента;
- `afterEnd` — непосредственно после закрывающего тега текущего элемента страницы.

Второй параметр задает ссылку на вставляемый элемент страницы.

Метод возвращает ссылку на вставленный элемент страницы.

Поддерживается IE начиная с 5.0.

insertAdjacentHTML

Вставляет фрагмент HTML-кода внутрь или рядом с текущим элементом страницы.

```
{Объект}.insertAdjacentHTML("beforeBegin|afterBegin|beforeEnd|
↳afterEnd", "{Вставляемый HTML-код}");
```

Метод принимает два параметра. Первый из них задает место, куда будет вставлен элемент страницы. Доступны четыре значения:

- `beforeBegin` — непосредственно перед открывающим тегом текущего элемента страницы;
- `afterBegin` — после открывающего тега, но перед всем остальным содержимым элемента;
- `beforeEnd` — перед закрывающим тегом, но после всего остального содержимого элемента;
- `afterEnd` — непосредственно после закрывающего тега текущего элемента страницы.

Второй параметр задает собственно вставляемый HTML-код.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

insertAdjacentText

Вставляет текстовый фрагмент внутрь или рядом с текущим элементом страницы. Всякое HTML-форматирование при этом игнорируется.

```
{Объект}.insertAdjacentText("beforeBegin|afterBegin|beforeEnd|
↳afterEnd", "{Вставляемый текст}");
```

Метод принимает два параметра. Первый из них задает место, куда будет вставлен элемент страницы. Доступны четыре значения:

- `beforeBegin` — непосредственно перед открывающим тегом текущего элемента страницы;
- `afterBegin` — после открывающего тега, но перед всем остальным содержимым элемента;
- `beforeEnd` — перед закрывающим тегом, но после всего остального содержимого элемента;

- `afterEnd` — непосредственно после закрывающего тега текущего элемента страницы.

Второй параметр задает собственно вставляемый текст.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

insertBefore

Вставляет новый элемент в Web-страницу.

```
{Объект}.insertBefore({Вставляемый элемент страницы}[, "{Дочерний
↳элемент}"]);
```

Метод принимает два параметра. Первый из них задает собственно ссылку на вновь созданный элемент страницы, который и должен быть в нее вставлен. Второй — необязательный — задает ссылку на дочерний элемент, перед которым он будет вставлен.

Метод возвращает ссылку на вставленный элемент страницы.

Поддерживается IE начиная с 5.0.

insertCell

Вставляет ячейку в строку таблицы.

```
{Объект строки таблицы}.insertCell([{Индекс}]);
```

Метод принимает единственный необязательный параметр — индекс новой ячейки в коллекции `cells`. Значение по умолчанию `-1` заставляет добавить ячейку в конец коллекции.

Метод возвращает ссылку на вставленную ячейку.

Поддерживается IE начиная с 4.0.

insertRow

Вставляет ячейку в таблицу или одну из ее секций.

```
{Объект}.insertRow([{Индекс}]);
```

Метод принимает единственный необязательный параметр — индекс новой строки в коллекции `rows`. Значение по умолчанию `-1` заставляет добавить строку в конец коллекции.

Метод возвращает ссылку на вставленную строку.

Поддерживается IE начиная с 4.0.

isEqual

Возвращает `true`, если переданный в параметре объект `TextRange` равен текущему объекту, и `false`, если не равен.

```
{Объект TextRange}.isEqual({Сравниваемый объект TextRange});
```

Метод принимает единственный параметр — ссылку на сравниваемый объект `TextRange`.

Поддерживается IE начиная с 4.0.

IsSubscribed

Возвращает `true`, если клиент подписан на данный канал, и `false` — в противном случае.

```
external.IsSubscribed("{Интернет-адрес CDF-файла}");
```

Метод принимает единственный параметр — интернет-адрес CDF-файла, описывающего канал.

Поддерживается IE начиная с 4.0.

item

Возвращает элемент коллекции под заданным индексом (номером).

```
{Коллекция}.item({Индекс}[, {Вторичный индекс}]);
```

Метод принимает два параметра. Первый из них — индекс нужного элемента коллекции. Индекс может быть как числовым, так и строковым значением; в первом случае просто возвращается элемент под данным номером, а во втором — элемент, значение атрибутов `ID` или `NAME` которого совпадает с переданным значением. Если таковых элементов несколько, возвращается новая коллекция, содержащая все найденные элементы коллекции. В этом случае может быть задан вторым параметром вторичный индекс, задающий номер элемента вторичной коллекции. Если подходящий элемент коллекции не найден, возвращается `null`.

Для коллекций `bookmarks`, `childNodes`, `children`, `controlRange`, `pages` и `rules` поддерживаются только числовые индексы.

Для коллекций `attributes`, `behaviorUrns`, `filters`, `frames`, `imports`, `namespaces`, `styleSheets`, `TextRange` и `TextRectangle`, кроме того, недоступны вторичные индексы.

Поддерживается IE начиная с 4.0.

javaEnabled

Возвращает `true`, если пользователем разрешено исполнение скриптовых программ, и `false` — в противном случае.

```
navigator.javaEnabled();
```

Метод не принимает параметров.

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

lastPage

Отображает в таблице, привязанной к данным, последнюю порцию (страницу) этих данных.

```
{Объект таблицы}.lastPage();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 5.0.

load

Загружает в слой Web-страницу.

```
{Объект слоя}.load("{Интернет-адрес страницы}", {Ширина слоя});
```

Метод принимает два параметра: первый задает интернет-адрес загружаемой в слой Web-страницы, а второй — новую ширину слоя в пикселах.

Метод не возвращает никакого значения.

Поддерживается N начиная с 4.0.

margins

Задает размеры полей между элементом страницы и его соседями.

```
{Объект стиля}.margins("{Верхнее}", "{Правое}", "{Нижнее}", "{Левое}");
```

Метод принимает четыре строковых параметра, задающие размеры верхнего, правого, нижнего и левого полей соответственно.

Метод не возвращает никакого значения.

Поддерживается N начиная с 4.0.

mergeAttributes

Копирует в текущий элемент страницы все атрибуты с их значениями из другого элемента.

```
{Объект}.mergeAttributes({Элемент страницы-источник}[, true|false]);
```

Метод принимает два параметра. Первый из них задает ссылку на элемент страницы, откуда будут копироваться атрибуты. Второй — необязательный — поможет разрешить конфликты при копировании атрибутов. Значение `true` (используемое по умолчанию) задает сохранение атрибутов текущего элемента, значения которого уже были выставлены, а значение `false` копирует все атрибуты. Начиная с версии 5.5 IE можно также копировать атрибуты, доступные только для чтения (например, `ID`).

Метод не возвращает никакого значения.

Поддерживается IE начиная с 5.0.

move

Сжимает содержимое объекта `TextRange` в точку и перемещает ее на заданное количество элементов текста.

```
{Объект TextRange}.move("character|word|sentence|textedit"
↳[, {Количество}]);
```

Метод принимает два параметра. Первый из них задает тип элемента текста:

- `character` — символ;
- `word` — слово;
- `sentence` — предложение;
- `textedit` — область текста, находившаяся в этом объекте ранее.

Второй параметр задает количество элементов текста, на которое будет сдвинут объект `TextRange`. Значение по умолчанию `1`.

Метод возвращает количество элементов текста, на которое фактически был сдвинут `TextRange`.

Поддерживается IE начиная с 4.0.

moveAbove

Помещает текущий слой выше заданного.

```
{Объект слоя}.moveAbove({Слой});
```

Метод принимает единственный параметр — ссылку на слой.

Метод не возвращает никакого значения.

Поддерживается N начиная с 4.0.

moveBelow

Помещает текущий слой ниже заданного.

```
{Объект слоя}.moveBelow({Слой});
```

Метод принимает единственный параметр — ссылку на слой.

Метод не возвращает никакого значения.

Поддерживается N начиная с 4.0.

moveBy (window)

Перемещает окно Web-обозревателя на заданное количество пикселов.

```
window.moveBy({X}, {Y});
```

Метод принимает два параметра, задающие в пикселах горизонтальное и вертикальное смещения окна соответственно.

Метод не возвращает никакого значения.

Поддерживается IE и N начиная с 4.0.

moveBy (слой)

Перемещает слой на заданное количество пикселов.

```
{Объект слоя}.moveBy({X}, {Y});
```

Метод принимает два параметра, задающие в пикселах горизонтальное и вертикальное смещения слоя соответственно.

Метод не возвращает никакого значения.

Поддерживается N начиная с 4.0.

moveEnd

Перемещает конечную границу объекта `TextRange` на заданное количество элементов текста.

```
{Объект TextRange}.moveEnd("character|word|sentence|textedit"
↳[, {Количество}]);
```

Метод принимает два параметра. Первый из них задает тип элемента текста:

- `character` — символ;
- `word` — слово;
- `sentence` — предложение;
- `textedit` — область текста, находившаяся в этом объекте ранее.

Второй параметр задает количество элементов текста, на которое будет сдвинута граница объекта `TextRange`. Значение по умолчанию `1`.

Метод возвращает количество элементов текста, на которое фактически была сдвинута граница `TextRange`.

Поддерживается IE начиная с 4.0.

moveRow

Перемещает строку таблицы или ее секции на новую позицию.

```
{Объект}.moveRow({Старый индекс}, {Новый индекс});
```

Метод принимает два параметра, задающие индексы строки в коллекции `rows` — старый и новый соответственно.

Метод возвращает ссылку на перемещенную строку.

Поддерживается IE начиная с 5.0.

moveStart

Перемещает начальную границу объекта `TextRange` на заданное количество элементов текста.

```
{Объект TextRange}.moveStart("character|word|sentence|textedit"
↳[, {Количество}]);
```

Метод принимает два параметра. Первый из них задает тип элемента текста:

- `character` — символ;
- `word` — слово;
- `sentence` — предложение;
- `textedit` — область текста, находившаяся в этом объекте ранее.

Второй параметр задает количество элементов текста, на которое будет сдвинута граница объекта `TextRange`. Значение по умолчанию `1`.

Метод возвращает количество элементов текста, на которое фактически была сдвинута граница `TextRange`.

Поддерживается IE начиная с 4.0.

moveTo (window)

Перемещает левый верхний угол окна Web-обозревателя в заданную точку относительно экрана.

```
window.moveTo({X}, {Y});
```

Метод принимает два параметра, задающие в пикселах горизонтальное и вертикальное смещения левого верхнего угла окна соответственно.

Метод не возвращает никакого значения.

Поддерживается IE и N начиная с 4.0.

moveTo (слой)

Перемещает левый верхний угол слоя в заданную точку относительно экрана.

```
{Объект слоя}.moveTo({X}, {Y});
```

Метод принимает два параметра, задающие в пикселах горизонтальное и вертикальное смещения левого верхнего угла слоя соответственно.

Метод не возвращает никакого значения.

Поддерживается N начиная с 4.0.

moveToAbsolute

Перемещает левый верхний угол слоя в заданную точку относительно страницы.

```
{Объект слоя}.moveToAbsolute({X}, {Y});
```

Метод принимает два параметра, задающие в пикселах горизонтальное и вертикальное смещения левого верхнего угла слоя соответственно.

Метод не возвращает никакого значения.

Поддерживается N начиная с 4.0.

moveToBookmark

Перемещает объект `TextRange` на фрагмент текста, заданный "закладкой".

```
{Объект TextRange}.moveToBookmark("{Закладка}");
```

Метод принимает единственный параметр — "закладку", полученную с помощью метода `getBookmark`.

Метод возвращает `true`, если произошло успешное перемещение на нужный фрагмент текста, и `false` — в противном случае.

Поддерживается IE начиная с 4.0.

moveToElementText

Перемещает объект `TextRange` на фрагмент текста, находящийся в заданном элементе страницы.

```
{Объект TextRange}.moveToElementText({Элемент страницы});
```

Метод принимает единственный параметр — ссылку на элемент страницы.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 5.0.

moveToPoint

Сжимает объект TextRange и перемещает его в заданную точку.

```
{Объект TextRange}.moveToPoint({X}, {Y});
```

Метод принимает два параметра, задающие в пикселах координаты точки относительно левого верхнего угла окна — горизонтальную и вертикальную соответственно.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

namedRecordset

Возвращает ссылку на объект именованного поднабора данных.

```
{Внедренный объект}.namedRecordset("{Имя}"[, "{Путь}"]);
```

Метод принимает два параметра: первый задает имя поднабора данных, а необязательный второй — путь к набору данных.

Поддерживается IE начиная с 5.0.

navigate

Загружает новую Web-страницу в текущее окно Web-обозревателя.

```
window.navigate("{Интернет-адрес}");
```

Метод принимает единственный параметр — интернет-адрес новой Web-страницы.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 3.02.

NavigateAndFind

Загружает новую Web-страницу в текущем окне Web-обозревателя и выделяет в ней заданный текст, если он есть.

```
external.NavigateAndFind("{Интернет-адрес}", "{Искомый текст}",
↳"{Фрейм}");
```

Метод принимает три параметра. Первый задает интернет-адрес новой Web-страницы. Второй — текст, который следует найти на этой новой странице. Третий — имя фрейма, в который будет выведена новая страница; его значение может быть пустой строкой.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

nextPage

Отображает в таблице, привязанной к данным, следующую порцию (страницу) этих данных.

```
{Объект таблицы}.nextPage();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 4.0.

open (document)

Открывает Web-страницу перед использованием методов `write` и `writeln`.

```
document.open("{Тип данных MIME}"[, "{Имя новой страницы}"]);
```

Метод принимает два параметра. Первый из них задает тип данных MIME, которые будут переданы в страницу; в настоящее время поддерживается только `text/html`. Второй — необязательный — параметр может принимать только значение `replace`, позволяющее заменить текущую позицию в списке "истории".

Метод возвращает ссылку на текущую Web-страницу.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

open (window)

Открывает новое окно Web-обозревателя.

```
window.open(["{Интернет-адрес}"][, "{Имя окна}"][, "{Параметры}"]
↳[, true|false]);
```

Метод принимает четыре необязательных параметра. Первый из них указывает интернет-адрес Web-страницы, которая будет загружена в новое окно. Второй задает имя нового окна или одно из предопределенных значений:

- `_blank` — открывает страницу в новом окне без имени;
- `_parent` — в родительском окне;
- `_search` — в панели поиска (доступно только в IE 5.0 и более новых);
- `_self` — в том же окне;
- `_top` — в самом верхнем родительском окне.

Третий параметр задает список дополнительных характеристик окна; все они перечислены в табл. 3.13. Четвертый параметр позволяет указать, заменить ли текущую позицию в списке "истории" на новую (значение `true`) или оставить все как было (значение `false`).

Метод возвращает ссылку на новое окно.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

paddings

Задает отступы между элементом страницы и рамкой.

```
{Объект стиля}.paddings("{Верхний}", "{Правый}", "{Нижний}", "{Левый}");
```

Метод принимает четыре строковых параметра, задающие верхний, правый, нижний и левый отступы соответственно.

Метод не возвращает никакого значения.

Поддерживается N начиная с 4.0.

parentElement

Возвращает ссылку на родительский элемент данного объекта `TextRange` или `null`, если такого нет.

```
{Объект TextRange}.parentElement();
```

Метод не принимает параметров.

Поддерживается IE начиная с 4.0.

pasteHTML

Вставляет фрагмент HTML-кода в объект `TextRange`, заменяя все его содержимое.

```
{Объект TextRange}.pasteHTML("{Вставляемый HTML-код}");
```

Метод принимает единственный параметр, собственно вставляемый фрагмент HTML-кода.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

preference

Позволяет получить или изменить значения некоторых пользовательских настроек.

```
navigator.preference("{Имя настройки}"[, {Значение}]);
```

Метод принимает два параметра: первый задает имя настройки, а второй — значение. Второй параметр является необязательным; если его пропустить, метод вернет текущее значение настройки. Если же второй параметр указан, вернется новое ее значение.

Имена всех доступных настроек перечислены в табл. П3.3.

Таблица П3.3. *Имена настроек, используемые в методе* `preference`

Настройка	Описание	Значение
`general.always_load_images`	Загружать или не загружать рисунки автоматически	`true` — загружать, `false` — не загружать
`security.enable_java`	Разрешить или запретить выполнение Java-апплетов	`true` — разрешить, `false` — запретить
`javascript.enabled`	Разрешить или запретить выполнение Java-скриптов	`true` — разрешить, `false` — запретить
`browser.enable_style_sheets`	Разрешить или запретить использование таблиц стилей	`true` — разрешить, `false` — запретить
`autoupdate.enabled`	Разрешить или запретить SmartUpdate (автоматическое обновление программы Web-обозревателя через Интернет)	`true` — разрешить, `false` — запретить
`network.cookie_cookieBehavior`	Разрешить получение всех cookie	`0`
`network.cookie_cookieBehavior`	Получать только те cookie, которые отправляются обратно на Web-сервер	`1`
`network.cookie_cookieBehavior`	Запретить все cookie	`2`
`network.cookie.warnAboutCookies`	Спрашивать или не спрашивать у пользователя разрешения на получение cookie	`true` — спрашивать, `false` — не спрашивать

Поддерживается N начиная с 4.0.

previousPage

Отображает в таблице, привязанной к данным, предыдущую порцию (страницу) этих данных.

```
{Объект таблицы}.previousPage();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 4.0.

print

Печатает Web-страницу, находящуюся в текущем окне, на принтере.

```
window.print();
```

Метод не принимает параметров и не возвращает никакого значения.

Поддерживается IE начиная с 5.0 и N начиная с 4.0.

prompt

Выводит на экран диалоговое окно с полем ввода, предлагающее пользователю ввести некое значение.

```
window.prompt(["{Текст приглашения}"][, "{Начальное значение}"]);
```

Метод принимает два необязательных параметра. Первый из них задает текст приглашения, который будет отображаться в диалоговом окне. Второй задает текст, изначально находящийся в поле ввода.

Метод возвращает числовое или текстовое значение, введенное пользователем в поле ввода.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

queryCommandEnabled

Возвращает `true`, если для данного фрагмента страницы может быть выполнена данная команда, и `false` — в противном случае.

```
{Объект}.queryCommandEnabled("{Команда}");
```

Метод принимает единственный параметр, задающий обозначение команды, выполняемой с помощью метода `execCommand`. Список доступных команд приведен далее в этом приложении.

Поддерживается IE начиная с 4.0 для `document` и `TextRange` и начиная с 5.0 для `controlRange`.

queryCommandIndeterm

Возвращает `true`, если команда, доступная для данного фрагмента страницы, находится в неопределенном состоянии, и `false` — в противном случае.

```
{Объект}.queryCommandIndeterm("{Команда}");
```

Метод принимает единственный параметр, задающий обозначение команды, выполняемой с помощью метода execCommand. Список доступных команд приведен далее в этом приложении.

Поддерживается IE начиная с 4.0 для document и TextRange и начиная с 5.0 для controlRange.

queryCommandState

Возвращает true, если данная команда была выполнена, и false — в противном случае.

```
{Объект}.queryCommandState("{Команда}");
```

Метод принимает единственный параметр, задающий обозначение команды, выполняемой с помощью метода execCommand. Список доступных команд приведен далее в этом приложении.

Поддерживается IE начиная с 4.0 для document и TextRange и начиная с 5.0 для controlRange.

queryCommandSupported

Возвращает true, если данным фрагментом страницы поддерживается данная команда, и false — в противном случае.

```
{Объект}.queryCommandSupported("{Команда}");
```

Метод принимает единственный параметр, задающий обозначение команды, выполняемой с помощью метода execCommand. Список доступных команд приведен далее в этом приложении.

Поддерживается IE начиная с 4.0 для document и TextRange и начиная с 5.0 для controlRange.

queryCommandValue

Возвращает значение команды, доступное для данного фрагмента страницы, или false, если таковое не поддерживается командой.

```
{Объект}.queryCommandValue("{Команда}");
```

Метод принимает единственный параметр, задающий обозначение команды, выполняемой с помощью метода execCommand. Список доступных команд приведен далее в этом приложении.

Поддерживается IE начиная с 4.0 для document и TextRange и начиная с 5.0 для controlRange.

random

Возвращает псевдослучайную строку символов заданной длины.

```
crypto.random({Длина строки в байтах});
```

Метод принимает единственный параметр — длину возвращаемой строки в байтах.

Поддерживается N начиная с 4.0.

recalc

Заставляет все динамические свойства перевычислить свои значения.

```
document.recalc([true|false]);
```

Метод принимает один необязательный параметр. Его значение `false` (используемое по умолчанию) заставляет перевычислить только те выражения, аргументы которых изменились после последнего перевычисления. Значение `true` заставляет перевычислить все выражения.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 5.0.

refresh (plugins)

Обновляет содержимое коллекции `plugins`. Это может понадобиться, например, для того, чтобы сделать вновь установленные расширения доступными.

```
plugins.refresh();
```

Метод не принимает параметров и не возвращает никакого значения.

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

refresh (<TABLE>)

Обновляет содержимое таблицы.

```
{Объект таблицы}.refresh();
```

Метод не принимает параметров и не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

releaseCapture

Отменяет режим "захвата мыши" для текущего элемента страницы.

```
{Объект}.releaseCapture();
```

Метод не принимает параметров и не возвращает никакого значения.

Поддерживается IE начиная с 5.0.

releaseEvents

Отменяет перехват событий заданного типа.

```
document.releaseEvents("{Список имен события}");
```

Метод принимает единственный параметр — список строковых имен событий, разделенных вертикальной чертой.

Метод не возвращает никакого значения.

Поддерживается N начиная с 4.0.

reload

Перезагружает текущую Web-страницу.

```
location.reload([true|false]);
```

Метод принимает необязательный параметр. Его значение `false` (используемое по умолчанию) заставляет Web-обозреватель перезагрузить страницу с жесткого диска, где она была сохранена ранее, а `true` — прямо с Web-сервера.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 5.0 и N начиная с 3.0.

remove

Удаляет элемент коллекции.

```
{Коллекция}.remove({Индекс});
```

Метод принимает единственный параметр — индекс удаляемого элемента коллекции.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0 для `areas` и `options` и начиная с 5.0 для `controlRange`.

removeAttribute

Удаляет заданный атрибут из тега элемента страницы вместе со значением.

```
{Объект}.removeAttribute("{Имя атрибута}"[, 0|1]);
```

Метод принимает два параметра. Первый задает имя удаляемого атрибута. Второй — необязательный — задает, учитывать ли регистр символов при

поиске атрибута (значение 1, используемое по умолчанию) или нет (значение 0). Если имеется в наличии несколько атрибутов с именами, различающимися только регистром символов, удаляется лишь последний атрибут.

Метод возвращает true, если атрибут был успешно удален, и false в противном случае.

Поддерживается IE начиная с 4.0.

removeBehavior

Удаляет заданное поведение.

```
{Объект}.removeBehavior({Идентификатор поведения});
```

Метод принимает единственный параметр — идентификатор поведения, возвращенный методом addBehavior.

Метод возвращает true, если поведение было успешно удалено, и false — в противном случае.

Поддерживается IE начиная с 5.0.

removeChild

Удаляет дочерний элемент из текущего элемента.

```
{Объект}.removeChild({Объект});
```

Метод принимает единственный параметр — ссылку на удаляемый элемент страницы.

Метод возвращает ссылку на удаленный дочерний элемент.

Поддерживается IE начиная с 5.0.

removeExpression

Удаляет выражение, присвоенное динамическому атрибуту стиля.

```
{Объект стиля}.removeExpression("{Имя атрибута}");
```

Метод принимает единственный параметр — имя атрибута стиля.

Метод возвращает true, если выражение было успешно удалено, и false — в противном случае.

Поддерживается IE начиная с 5.0.

removeNode

Удаляет текущий элемент из Web-страницы.

```
{Объект}.removeNode([true|false]);
```

Метод принимает единственный параметр, задающий, удалять коллекцию `childNodes` удаляемого элемента страницы (значение `true`) или нет (значение `false`, используемое по умолчанию).

Метод возвращает ссылку на удаленный элемент.

Поддерживается IE начиная с 5.0.

removeRule

Удаляет правило из текущей таблицы стилей.

```
{Объект таблицы стилей}.removeRule([{Индекс}]);
```

Метод принимает единственный необязательный параметр — индекс удаляемого правила в коллекции. Если он не задан, будет удалено первое правило.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

replace

Замещает текущую Web-страницу новой с заменой соответствующей позиции в списке "истории".

```
location.replace("{Интернет-адрес новой Web-страницы}");
```

Метод принимает единственный параметр — интернет-адрес новой Web-страницы.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

replaceAdjacentText

Заменяет текст, находящийся внутри текущего элемента страницы или рядом с ним, на другой текст. Всякое HTML-форматирование при этом игнорируется.

```
{Объект}.replaceAdjacentText("beforeBegin|afterBegin|beforeEnd|
↪afterEnd", "{Заменяющий текст}");
```

Метод принимает два параметра. Первый из них задает место, куда будет вставлен элемент страницы. Доступны четыре значения:

- `beforeBegin` — непосредственно перед открывающим тегом текущего элемента страницы;
- `afterBegin` — после открывающего тега, но перед всем остальным содержимым элемента;

- `beforeEnd` — перед закрывающим тегом, но после всего остального содержимого элемента;
- `afterEnd` — непосредственно после закрывающего тега текущего элемента страницы.

Второй параметр задает собственно заменяющий текст.

Метод возвращает строку, содержащую замещенный текст.

Поддерживается IE начиная с 5.0.

replaceChild

Заменяет дочерний элемент на другой.

```
{Объект}.replaceChild({Новый объект}, {Старый объект});
```

Метод принимает два параметра, задающие ссылки на замещаемый и замещающий элементы страницы соответственно.

Метод возвращает ссылку на замененный дочерний элемент.

Поддерживается IE начиная с 5.0.

replaceNode

Заменяет текущий элемент Web-страницы на другой.

```
{Объект}.replaceNode({Новый объект});
```

Метод принимает единственный параметр, задающий ссылку на новый элемент страницы.

Метод возвращает ссылку на замененный элемент.

Поддерживается IE начиная с 5.0.

reset

Вызывает сброс данных формы. Аналогичен по действию нажатию кнопки `reset`.

```
{Объект формы}.reset();
```

Метод не принимает параметров и не возвращает никакого значения.

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

resizeBy (window)

Изменяет размеры окна Web-обозревателя на заданное количество пикселов.

```
window.resizeBy({X}, {Y});
```

Метод принимает два параметра, задающие в пикселах приращения ширины и высоты окна соответственно.

Метод не возвращает никакого значения.

Поддерживается IE и N начиная с 4.0.

resizeBy (слой)

Изменяет размеры слоя на заданное количество пикселов.

```
{Объект слоя}.resizeBy({X}, {Y});
```

Метод принимает два параметра, задающие в пикселах приращения ширины и высоты слоя соответственно.

Метод не возвращает никакого значения.

Поддерживается N начиная с 4.0.

resizeTo (window)

Задает размеры окна Web-обозревателя.

```
window.resizeTo({X}, {Y});
```

Метод принимает два параметра, задающие в пикселах ширину и высоту окна соответственно.

Метод не возвращает никакого значения.

Поддерживается IE и N начиная с 4.0.

resizeTo (слой)

Задает размеры слоя.

```
{Объект слоя}.resizeTo({X}, {Y});
```

Метод принимает два параметра, задающие в пикселах ширину и высоту слоя соответственно.

Метод не возвращает никакого значения.

Поддерживается N начиная с 4.0.

routeEvent

Передает событие элементу, в котором оно произошло, для дальнейшей обработки.

```
document.routeEvent({Событие});
```

Метод принимает единственный параметр — ссылку на объект `event`.

Метод не возвращает никакого значения.

Поддерживается N начиная с 4.0.

savePreferences

Позволяет сохранить пользовательские настройки.

```
navigator.savePreferences();
```

Метод не принимает параметров и не возвращает никакого значения.

Поддерживается N начиная с 4.0.

scroll

Прокручивает содержимое окна Web-обозревателя до заданной точки.

```
window.scroll({X}, {Y});
```

Метод принимает два параметра, задающие в пикселах координаты точки, до которой следует прокрутить содержимое окна, — горизонтальную и вертикальную соответственно.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0; в настоящее время признан устаревшим и не рекомендуется к использованию. Поддерживается N начиная с 3.0; в 4.0 удален.

scrollBy

Прокручивает содержимое окна Web-обозревателя на заданное количество пикселов.

```
window.scrollBy({X}, {Y});
```

Метод принимает два параметра, задающие в пикселах величины прокрутки по горизонтали и по вертикали окна соответственно. Положительные величины задают прокрутку вправо и вниз, чтобы прокрутить влево и вверх, задайте отрицательные значения приращений.

Метод не возвращает никакого значения.

Поддерживается IE и N начиная с 4.0.

scrollIntoView

Прокручивает содержимое окна Web-обозревателя так, чтобы текущий элемент страницы стал виден.

```
{Объект}.scrollIntoView([true|false]);
```

Метод принимает единственный необязательный параметр, задающий, должен ли текущий элемент быть виден у верхнего края окна (значение `true`, используемое по умолчанию) или у нижнего (значение `false`).

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

scrollTo

Прокручивает содержимое окна Web-обозревателя до заданной точки.

```
window.scrollTo({X}, {Y});
```

Метод принимает два параметра, задающие в пикселах координаты точки, до которой следует прокрутить содержимое окна, — горизонтальную и вертикальную соответственно.

Метод не возвращает никакого значения.

Поддерживается IE и N начиная с 4.0.

select (controlRange, TextRange)

Выделяет на Web-странице текст, являющийся содержимым текущего объекта.

```
{Объект}.select();
```

Метод не принимает параметров и не возвращает никакого значения.

Поддерживается IE начиная с 4.0 для `TextRange` и начиная с 5.0 для `controlRange`.

select (элементы управления)

Выделяет элемент управления.

```
{Объект элемента управления}.select();
```

Метод не принимает параметров и не возвращает никакого значения.

Поддерживается IE начиная с 4.0 и N начиная с 2.0.

setActive

Делает текущий элемент страницы активным без переноса на него фокуса ввода.

```
{Объект}.setActive();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 5.5.

setAttribute

Помещает в тег элемента страницы новый атрибут вместе со значением.

```
{Объект}.setAttribute("{Имя атрибута}", "{Значение атрибута}"[, 0|1]);
```

Метод принимает три параметра. Первый задает имя вновь помещаемого атрибута, а второй — его значение. Если имя вновь помещаемого атрибута совпадает с именем другого, уже существующего, последний будет перезаписан новым значением. Необязательный третий параметр задает, учитывать при этом регистр символов (значение `1`, используемое по умолчанию) или нет (значение `0`).

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

setCapture

Заставляет текущий элемент страницы перехватывать все события мыши вне зависимости от того, каким элементом страницы они были вызваны.

```
{Объект}.setCapture([true|false]);
```

Метод принимает единственный необязательный параметр, который задает, будет ли текущий элемент перехватывать все события, вызванные дочерними элементами (значение `true`, используемое по умолчанию), или нет (значение `false`). В последнем случае события направляются соответствующим элементам и потом постепенно "всплывают" к родителям (обычное поведение событий).

Метод не возвращает никакого значения.

Поддерживается IE начиная с 5.0.

setData

Помещает данные в объект `clipboardData` или `dataTransfer`.

```
{Объект}.setData("{Формат}", "{Данные}");
```

Метод принимает два параметра. Первый из них задает формат помещаемых данных: значение `text` позволит поместить обычный текст, а значение `URL` — интернет-адрес. Второй параметр задает собственно помещаемые данные.

Метод возвращает `true`, если данные были успешно помещены, и `false` — в противном случае.

Поддерживается IE начиная с 5.0.

setEndPoint

Устанавливает границу текущего объекта TextRange по границе другого аналогичного объекта.

```
{Объект TextRange}.setEndPoint(StartToEnd|StartToStart|EndToStart|
↳EndToEnd, {Второй объект TextRange});
```

Метод принимает два параметра. Первый из них задает границы объектов. Возможны четыре значения:

- StartToEnd — начальная граница текущего объекта совмещается с конечной границей второго объекта;
- StartToStart — начальная с начальной;
- EndToStart — конечная с начальной;
- EndToEnd — конечная с конечной.

Второй параметр задает ссылку на второй объект TextRange.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

setExpression

Присваивает выражение динамическому атрибуту стиля.

```
{Объект стиля}.setExpression("{Имя атрибута}", "{Выражение}"[, "JScript|
↳VBScript|JavaScript"]);
```

Метод принимает три параметра. Первый из них задает имя атрибута стиля, второй — исходный текст выражения на скриптовом языке. Сам скриптовый язык задается в третьем необязательном параметре; доступны три значения: JScript (значение по умолчанию), JavaScript и VBScript.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 5.0.

setHotKeys

Разрешает или запрещает "горячие" клавиши в окне без строки меню.

```
window.setHotKeys(true|false);
```

Метод принимает единственный параметр. Значение true его разрешает "горячие" клавиши, значение false запрещает.

Поддерживается N начиная с 4.0.

setInterval

Создает новый интервальный таймер.

```
window.setInterval("{Выражение}"|{Функция}, {Интервал}[, "{Язык}"]);
```

Метод принимает три параметра. Первый из них задает либо имя функции, либо текст выражения, которое будет выполняться каждый раз по истечении заданного времени. Второй параметр задает само это время в миллисекундах. Необязательный третий параметр задает скриптовый язык, на котором написано выражение или функция.

Метод возвращает целочисленный идентификатор вновь созданного интервального таймера.

Поддерживается IE и N начиная с 4.0.

setResizable

Разрешает или запрещает пользователю изменять размеры окна.

```
window.setResizable(true|false);
```

Метод принимает единственный параметр, значение `true` которого разрешает изменение размеров окна, значение `false` запрещает.

Поддерживается N начиная с 4.0.

setTimeout

Создает новый тайм-аут.

```
window.setTimeout("{Выражение}"|{Функция}, {Интервал}[, "{Язык}"]);
```

Метод принимает три параметра. Первый из них задает либо имя функции, либо текст выражения, которое выполнится по истечении заданного времени. Второй параметр задает само это время в миллисекундах. Необязательный третий параметр задает скриптовый язык, на котором написано выражение или функция: `JScript` (значение по умолчанию), `JavaScript` и `VBScript`.

Метод возвращает целочисленный идентификатор вновь созданного тайм-аута.

Поддерживается IE начиная с 3.02. Поддерживается N начиная с 2.0; вызов функции — только с 4.0.

setZOption

Задает порядок перекрытия текущего окна другими окнами.

```
window.setZOption("alwaysRaised|alwaysLowered|z-lock");
```

Метод принимает единственный параметр, для которого доступны три значения:

- `alwaysRaised` — заставляет текущее окно находиться над остальными окнами, даже если оно неактивно;
- `alwaysLowered` — заставляет текущее окно находиться под остальными окнами, даже если оно активно;
- `z-lock` — заставляет текущее окно сохранять свою "координату Z".

Поддерживается N начиная с 4.0.

show

Показывает всплывающее окно.

```
{Объект}.show({X}, {Y}, {Ширина}, {Высота}[, {Объект}]);
```

Метод принимает пять параметров. Первые четыре из них задают горизонтальную и вертикальную координаты левого верхнего угла окна, его ширину и высоту. Последний необязательный параметр задает ссылку на объект, относительно которого будут вычисляться эти координаты; по умолчанию это экран компьютера `screen`.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 5.5.

ShowBrowserUI

Открывает одно из диалоговых окон программы Web-обозревателя.

```
external.ShowBrowserUI("LanguageDialog|OrganizeFavorites", null);
```

Метод принимает два параметра. Первый из них задает диалоговое окно, которое будет открыто при вызове этого метода. Доступны два значения:

- `LanguageDialog` — "Выбор языка";
- `OrganizeFavorites` — "Упорядочить избранное".

Второй параметр всегда равен `null`.

Значение, возвращаемое методом, не используется.

Поддерживается IE начиная с 5.0.

showHelp

Выводит окно справки.

```
window.showHelp("{Интернет-адрес}"[, "{Контекстный идентификатор}"]);
```

Метод принимает два параметра. Первый из них задает интернет-адрес Web-страницы, которая будет выводиться в окне справки. Необязательный второй параметр задает идентификатор контекстной справки в строковом или целочисленном виде.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 4.0.

showModalDialog

Выводит модальное диалоговое Web-окно.

```
window.showModalDialog("{Интернет-адрес}"[, {Аргументы}][, "{Параметры}"]);
```

Метод принимает три параметра. Первый из них задает интернет-адрес Web-страницы, которая будет выведена как модальное диалоговое окно. Необязательный второй параметр задает дополнительные данные, которые будут переданы в диалоговое окно; можно задать данные любого типа, в том числе и массив. Необязательный же третий параметр задает дополнительные характеристики самого окна; все они перечислены в табл. 5.11.

Метод возвращает значение, присвоенное свойству `returnValue` диалогового окна.

Поддерживается IE начиная с 4.0.

showModelessDialog

Выводит немодальное диалоговое Web-окно.

```
window.showModelessDialog("{Интернет-адрес}"[, {Аргументы}]
↳[, "{Параметры}"]);
```

Метод принимает три параметра. Первый из них задает интернет-адрес Web-страницы, которая будет выводиться как немодальное диалоговое окно. Необязательный второй задает дополнительные данные, которые будут переданы в диалоговое окно; можно задать данные любого типа, в том числе и массив. Необязательный же третий параметр задает дополнительные характеристики самого окна; все они перечислены в табл. 5.11.

Метод возвращает ссылку на созданное окно.

Поддерживается IE начиная с 5.0.

signText

Шифрует строку текста.

```
crypto.signText("{Текст}", "ask|auto"[, "{Сертификат 1}"]...
↳[, "{Сертификат n}"]);
```

Метод принимает два обязательных параметра. Первый из них задает собственно шифруемый текст. Второй позволяет задать пользовательский сертификат, с помощью которого текст будет шифроваться: значение `ask` приведет к показу диалогового окна, в котором пользователь должен будет выбрать нужный сертификат, а значение `auto` отдаст выбор сертификата на откуп Web-обозревателю. В последнем случае необходимо будет задать дополнительные строковые параметры — список сертификатов.

Метод возвращает зашифрованный текст.

Поддерживается N начиная с 4.0.

splitText

Делит текст текущего объекта `TextNode` страницы на две части и помещает вторую часть во вновь созданный объект `TextNode`.

```
TextNode.splitText([{Индекс}]);
```

Метод принимает один необязательный параметр. Он задает индекс (номер) символа в строке, по которому строка будет разделена на две части. Если он пропущен или равен `0`, будет создан новый объект `TextNode`, содержащий пустую строку.

Метод возвращает ссылку на вновь созданный объект `TextNode`.

Поддерживается IE начиная с 5.0.

start

Запускает прокрутку текста в `<MARQUEE>`.

```
{Объект}.start();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 4.0.

stop (<MARQUEE>)

Останавливает прокрутку текста в `<MARQUEE>`.

```
{Объект}.stop();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается IE начиная с 4.0.

stop (window)

Останавливает загрузку Web-страницы.

```
window.stop();
```

Метод не принимает параметров и не возвращает значения.

Поддерживается N начиная с 4.0.

submit

Вызывает отправку данных формой серверному приложению. Аналогичен по действию нажатию кнопки `submit`.

```
{Объект формы}.submit();
```

Метод не принимает параметров и не возвращает никакого значения.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

swapNode

Меняет местами два элемента страницы: текущий и переданный в параметре.

```
{Объект}.swapNode({Второй элемент});
```

Метод принимает единственный параметр — ссылку на второй элемент страницы.

Метод возвращает ссылку на текущий элемент страницы.

Поддерживается IE начиная с 5.0.

tags

Возвращает коллекцию элементов страницы, созданных с помощью заданного тега, или `null`, если таковых нет.

```
{Коллекция}.tags("{Тег}");
```

Метод принимает единственный параметр — имя нужного тега.

Поддерживается IE начиная с 4.0.

taintEnabled

Возвращает `true`, если Web-обозреватель поддерживает бойкот данных, и `false` — в противном случае.

```
navigator.taintEnabled();
```

Метод не принимает параметров.

Поддерживается IE начиная с 4.0, но возвращается всегда `false`. Поддерживается только N 3.0, в 4.0 и более поздних версиях этот метод удален.

urns

Возвращает коллекцию элементов страницы, к которым было привязано данное поведение, или null, если таковых нет.

```
{Коллекция}.urns("{Поведение}");
```

Метод принимает единственный параметр — имя нужного поведения в формате URN.

Поддерживается IE начиная с 5.0.

write

Помещает в данное место Web-страницы HTML-код или простой текст, переданный в качестве параметра.

```
document.write("{Текст или HTML-код}");
```

Метод принимает единственный параметр — строку вставляемого текста или HTML-кода.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

writeln

Помещает в данное место Web-страницы HTML-код или простой текст, переданный в качестве параметра. Завершает вставленный текст или HTML-код символом возврата каретки, что может иметь значение в тексте с заданным форматом.

```
document.writeln("{Текст или HTML-код}");
```

Метод принимает единственный параметр — строку вставляемого текста или HTML-кода.

Метод не возвращает никакого значения.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

События

onabort

Наступает, когда пользователь прерывает загрузку графического изображения.

Действие по умолчанию: прерывание загрузки соответствующего файла. Не "всплывает".

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

onactivate

Наступает, когда элемент страницы получает фокус ввода.

Действие по умолчанию отсутствует.

Поддерживается IE начиная с 5.5.

onafterprint

Наступает сразу после вывода на принтер или предварительный просмотр текущей Web-страницы.

Действие по умолчанию отсутствует. Не "всплывает".

Поддерживается IE начиная с 5.0.

onafterupdate

Наступает сразу после того, как данные будут перенесены из элемента управления в соответствующее поле базы данных. Доступно только для элементов управления, привязанных к данным.

Действие по умолчанию: сохранение данных. Не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 4.0.

onbeforecopy

Наступает перед копированием данных из текущего элемента страницы в Буфер обмена Windows.

Действие по умолчанию отсутствует.

Поддерживается IE начиная с 5.0.

onbeforecut

Наступает перед переносом данных из текущего элемента страницы в Буфер обмена Windows. Присвоив значение `false` свойству `returnValue` в обработчике этого события, можно разрешить операцию вырезания текста для текущего элемента страницы.

Действие по умолчанию отсутствует.

Поддерживается IE начиная с 5.0.

onbeforedeactivate

Наступает перед потерей фокуса текущим элементом страницы.

Действие по умолчанию отсутствует.

Поддерживается IE начиная с 5.5.

onbeforeeditfocus

Наступает перед переходом элемента страницы в режим редактирования. Доступно только если свойство `designMode` установлено в `on`.

Действие по умолчанию: перевод элемента страницы в режим редактирования.

Поддерживается IE начиная с 5.0.

onbeforepaste

Наступает непосредственно перед вставкой данных из Буфера обмена в текущий элемент страницы. Присвоив значение `false` свойству `returnValue` в обработчике этого события, можно разрешить операцию вставки текста для текущего элемента страницы.

Действие по умолчанию отсутствует.

Поддерживается IE начиная с 5.0.

onbeforeprint

Наступает перед выводом на принтер или предварительным просмотром текущей Web-страницы.

Действие по умолчанию: вывод на принтер или предварительный просмотр текущей Web-страницы. Не "всплывает" и не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 5.0.

onbeforeunload

Наступает перед "выгрузкой" текущей Web-страницы в результате перехода на другую страницу или закрытия окна Web-обозревателя.

Действие по умолчанию: сигнализация, что Web-страница сейчас будет закрыта. Не "всплывает".

Поддерживается IE начиная с 4.0.

onbeforeupdate

Наступает перед переносом данных из элемента управления в соответствующее поле базы данных. Доступно только для элементов управления, привязанных к данным.

Действие по умолчанию: сигнализация, что данные были изменены.

Поддерживается IE начиная с 4.0.

onblur

Наступает, когда окно или элемент страницы теряет фокус ввода.

Действие по умолчанию: перенос фокуса ввода на другой элемент страницы. Не "всплывает" и не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 3.02 для `<SELECT>` и `<TEXTAREA>`, начиная с 4.0 для остальных элементов управления, фреймов, гиперссылок, внедренных объектов, таблиц, `<DIV>`, `<HR>`, `<SPAN>` и `window` и начиная с 5.0 для остальных элементов. Поддерживается N начиная с 2.0.

onbounce

Наступает, когда текст элемента `<MARQUEE>` достигает одной из его сторон, сразу перед изменением направления движения. Доступно только если свойство `behavior` установлено в `alternate`.

Действие по умолчанию: изменение направление движения текста. Не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 4.0.

oncellchange

Наступает, когда изменяются данные, находящиеся в базе данных.

Действие по умолчанию: сигнализация об изменении данных в базе данных. Не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 5.0.

onchange

Наступает, когда содержимое элемента управления изменяется.

Действие по умолчанию: изменение содержимого элемента управления. Не "всплывает".

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

onclick

Наступает при щелчке левой кнопкой мыши по элементу страницы, также при нажатии клавиши <Enter> на форме, использовании клавиши-ускорителя или выборе пункта в списке.

Действие по умолчанию разное для каждого конкретного элемента страницы.

Поддерживается IE начиная с 4.0 и N начиная с 2.0.

oncontextmenu

Наступает при щелчке правой кнопкой мыши по элементу страницы, перед выводом контекстного меню.

Действие по умолчанию: вывод контекстного меню.

Поддерживается IE начиная с 5.0.

oncontrolselect

Наступает при попытке пользователя выбрать элемент страницы, имеющий собственный пользовательский интерфейс.

Действие по умолчанию: активизация ползунков изменения размера.

Поддерживается IE начиная с 5.5.

oncopy

Наступает при копировании данных из текущего элемента страницы в Буфер обмена Windows.

Действие по умолчанию: копирование выделенного фрагмента в Буфер обмена.

Поддерживается IE начиная с 5.0.

oncut

Наступает при переносе данных из текущего элемента страницы в Буфер обмена Windows.

Действие по умолчанию: перенос выделенного фрагмента в Буфер обмена.

Поддерживается IE начиная с 5.0.

ondataavailable

Наступает каждый раз, когда очередная партия данных переносится из базы данных на Web-страницу.

Действие по умолчанию: сигнализация, что данные готовы для отображения. Не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 4.0.

ondatasetchanged

Наступает, когда данные в базе данных изменяются, а также когда начальная порция данных готова для переноса на Web-страницу.

Действие по умолчанию: сигнализация, что данные изменились. Не "всплывает".

Поддерживается IE начиная с 4.0.

ondatasetcomplete

Наступает, когда все данные перенесены из базы данных на Web-страницу.

Действие по умолчанию: сигнализация, что все данные готовы для отображения. Не "всплывает".

Поддерживается IE начиная с 4.0.

ondblclick

Наступает при двойном щелчке левой кнопкой мыши по элементу страницы.

Действие по умолчанию разное для каждого конкретного элемента страницы.

Поддерживается IE и N начиная с 4.0.

ondeactivate

Наступает при потере фокуса текущим элементом страницы.

Действие по умолчанию отсутствует. Не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 5.5.

ondrag

Наступает во время операции drag-n-drop в элементе-источнике.

Действие по умолчанию: вызов функции-обработчика, если таковая есть.

Поддерживается IE начиная с 5.0.

ondragdrop

Наступает, когда пользователь "бросает" в окно Web-обозревателя ссылки на интернет-адреса.

Параметры: `type` — тип события, `target` — ссылка на элемент страницы, где оно наступило, `data` — массив строк, каждая из которых содержит "брошенный" в окно адрес, `modifiers` — модификаторы, указывающие, какая клавиша на клавиатуре была при этом нажата, `screenX` и `screenY` — экранные координаты курсора мыши.

Поддерживается N начиная с 4.0.

ondragend

Наступает, когда пользователь отпускает кнопку мыши, завершая операцию drag-n-drop, в элементе-источнике.

Действие по умолчанию: вызов функции-обработчика, если таковая есть.

Поддерживается IE начиная с 5.0.

ondragenter

Наступает, когда пользователь перетаскивает данные в допустимое место, в элементе-цели.

Действие по умолчанию: вызов функции-обработчика, если таковая есть.

Поддерживается IE начиная с 5.0.

ondragleave

Наступает, когда пользователь перетаскивает данные из допустимого места, в элементе-цели.

Действие по умолчанию: вызов функции-обработчика, если таковая есть.

Поддерживается IE начиная с 5.0.

ondragover

Наступает во время операции drag-n-drop в элементе-цели, когда пользователь перетаскивает данные над допустимым местом.

Действие по умолчанию: вызов функции-обработчика, если таковая есть.

Поддерживается IE начиная с 5.0.

ondragstart

Наступает, когда пользователь начинает перетаскивать данные, в элементе-источнике.

Действие по умолчанию: вызов функции-обработчика, если таковая есть.

Поддерживается IE начиная с 4.0.

ondrop

Наступает, когда пользователь отпускает кнопку мыши, завершая операцию drag-n-drop, в элементе-цели.

Действие по умолчанию: вызов функции-обработчика, если таковая есть.

Поддерживается IE начиная с 5.0.

onerror

Наступает, если при загрузке Web-страницы или ее элемента происходит ошибка.

Действие по умолчанию: вывод окна-предупреждения об ошибке. Не "всплывает".

Поддерживается IE начиная с 3.02 для `window` и начиная с 4.0 для остальных элементов. Поддерживается N начиная с 3.0.

onerrorupdate

Наступает, если при переносе измененных данных в базу данных происходит ошибка.

Действие по умолчанию: вызов функции-обработчика, если таковая есть. Не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 4.0.

onfilterchange

Наступает, когда визуальный фильтр изменяет свое состояние или когда визуальное преобразование заканчивает свою работу.

Действие по умолчанию: сигнализация о завершении работы фильтра или преобразования. Не "всплывает" и не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 4.0.

onfinish

Наступает, когда очередной цикл движения текста в элементе <MARQUEE> завершается. Доступно, только если свойство loop установлено в значение, отличное от 1.

Действие по умолчанию: прекращение цикла. Не "всплывает".

Поддерживается IE начиная с 4.0.

onfocus

Наступает, когда элемент страницы получает фокус ввода.

Действие по умолчанию: установка фокуса ввода на текущий элемент страницы. Не "всплывает" и не прерывается присвоением значения false свойству returnValue.

Поддерживается IE начиная с 3.02 для элементов управления и <TEXTAREA>, начиная с 4.0 для фреймов, гиперссылок, внедренных объектов, таблиц, <DIV>, <SELECT>, <SPAN> и window и начиная с 5.0 для остальных элементов. Поддерживается N начиная с 2.0.

onhelp

Наступает при нажатии клавиши <F1>.

Действие по умолчанию: открытие окна справки.

Поддерживается IE начиная с 4.0.

onkeydown

Наступает, когда пользователь нажимает клавишу на клавиатуре.

Действие по умолчанию: возвращение кода нажатой клавиши в свойстве keyCode.

Поддерживается IE и N начиная с 4.0.

onkeypress

Наступает, когда пользователь нажимает алфавитно-цифровую клавишу на клавиатуре.

Действие по умолчанию: возвращение кода нажатой клавиши в формате Unicode.

Поддерживается IE и N начиная с 4.0.

onkeyup

Наступает, когда пользователь отпускает ранее нажатую клавишу клавиатуры.

Действие по умолчанию: возвращение кода нажатой клавиши в свойстве `keyCode`. Не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE и N начиная с 4.0.

onlayoutcomplete

Наступает, когда Web-страница уже готова для печати на принтере или предварительного просмотра (объект `LayoutRect` заполнен).

Действие по умолчанию: вызов функции-обработчика, если таковая есть.

Поддерживается IE начиная с 5.5.

onload

Наступает сразу по окончании загрузки Web-страницы или ее элемента.

Действие по умолчанию: отображение страницы или ее элемента. Не "всплывает" и не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 3.02 для `window`, начиная с 4.0 для внедренных объектов, `<BODY>`, `<FRAMESET>`, `<IMG>`, `<LINK>` и `<SCRIPT>` и начиная с 5.5 для фреймов. Поддерживается N начиная с 2.0.

onlosecapture

Наступает, когда элемент страницы перестает перехватывать все события мыши.

Действие по умолчанию: вызов функции-обработчика, если таковая есть. Не "всплывает" и не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 5.0.

onmousedown

Наступает при щелчке любой кнопкой мыши по элементу страницы.

Действие по умолчанию разное для каждого конкретного элемента страницы.

Поддерживается IE и N начиная с 4.0.

onmouseenter

Наступает, когда пользователь помещает курсор мыши на элемент страницы.

Действие по умолчанию: вызов функции-обработчика, если таковая есть. Не "всплывает" и не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 5.5.

onmouseleave

Наступает, когда пользователь убирает курсор мыши с элемента страницы.

Действие по умолчанию: вызов функции-обработчика, если таковая есть. Не "всплывает" и не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 5.5.

onmousemove

Наступает, когда пользователь перемещает курсор мыши над элементом страницы.

Действие по умолчанию: вызов функции-обработчика, если таковая есть. Не "всплывает" и не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE и N начиная с 4.0.

onmouseout

Наступает, когда пользователь убирает курсор мыши с элемента страницы.

Действие по умолчанию: вызов функции-обработчика, если таковая есть. Не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

onmouseover

Наступает, когда пользователь помещает курсор мыши на элемент страницы.

Действие по умолчанию: вызов функции-обработчика, если таковая есть.

Поддерживается IE начиная с 4.0 и N начиная с 2.0.

onmouseup

Наступает при отпускании ранее нажатой кнопки мыши над элементом страницы.

Действие по умолчанию свое для каждого конкретного элемента страницы.

Поддерживается IE и N начиная с 4.0.

onmove

Наступает, когда пользователь перемещает окно Web-обозревателя или изменяет размеры фрейма.

Параметры: `type` — тип события, `target` — ссылка на элемент страницы, где оно наступило, `X` и `Y` — координаты левого верхнего угла окна или фрейма.

Поддерживается N начиная с 4.0.

onpaste

Наступает при вставке данных из Буфера обмена Windows в текущий элемент страницы.

Действие по умолчанию: вставка содержимого Буфера обмена.

Поддерживается IE начиная с 5.0.

onpropertychange

Наступает при изменении одного из свойств текущего элемента страницы.

Действие по умолчанию: сигнализация об изменении значения свойства. Не "всплывает" и не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 5.0.

onreadystatechange

Наступает при изменении состояния (значения свойства `readyState`) элемента страницы.

Действие по умолчанию: сигнализация об изменении значения свойства `readyState`. Не "всплывает" и не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 3.02 для `<Hn>` и `namespace`, начиная с 4.0 для `<LINK>`, `<OBJECT>`, `<SCRIPT>` и `document`, начиная с 5.0 для остальных элементов и начиная с 5.5 для `<IFRAME>`.

onreset

Наступает при сбросе данных формы (нажатии кнопки `reset`).

Действие по умолчанию: вызов функции-обработчика, если таковая есть. Не "всплывает".

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

onresize

Наступает непосредственно при изменении размеров окна, фрейма или другого элемента страницы.

Действие по умолчанию отсутствует. Не "всплывает" и не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 4.0 для всех элементов, кроме `<FRAME>` и `custom`, и начиная с 5.0 — для `<FRAME>` и `custom`. Поддерживается N начиная с 4.0.

onresizeend

Наступает по окончании изменения пользователем размеров элемента страницы, имеющего собственный пользовательский интерфейс.

Действие по умолчанию: изменение размеров элемента страницы. Не "всплывает" и не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 5.5.

onresizestart

Наступает, когда пользователь начинает изменять размеры элемента страницы, имеющего собственный пользовательский интерфейс.

Действие по умолчанию отсутствует. Не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 5.5.

onrowenter

Наступает при переходе на другую запись базы данных.

Действие по умолчанию: сигнализация о готовности данных, содержащихся в новой записи. Не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 4.0.

onrowexit

Наступает перед переходом с текущей записи базы данных на другую.

Действие по умолчанию: сигнализация об изменении записи. Не "всплывает".

Поддерживается IE начиная с 4.0.

onrowsdelete

Наступает непосредственно перед удалением текущей записи из базы данных.

Действие по умолчанию: сигнализация об удалении записи. Не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 5.0.

onrowsinserted

Наступает сразу после вставки новой записи в базу данных.

Действие по умолчанию: сигнализация о вставке новой записи. Не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 5.0.

onscroll

Наступает, когда пользователь прокручивает содержимое элемента страницы.

Действие по умолчанию: прокрутка содержимого элемента страницы. Не "всплывает" и не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 4.0.

onselect

Наступает, когда пользователь выделяет фрагмент содержимого поля ввода.

Параметры: `type` — тип события и `target` — ссылка на элемент страницы, где оно наступило.

Поддерживается N начиная с 2.0.

onselection

Наступает, когда пользователь выделяет фрагмент содержимого страницы или поля ввода.

Действие по умолчанию: выделение фрагмента содержимого страницы или поля ввода. Не "всплывает".

Поддерживается IE начиная с 3.02 для полей ввода и начиная с 4.0 для <BODY>.

onselectionchange

Наступает, когда меняется тип выделения (значение свойства selection.type).

Действие по умолчанию отсутствует. Не "всплывает" и не прерывается присвоением значения false свойству returnValue.

Поддерживается IE начиная с 5.5.

onselectstart

Наступает, когда пользователь начинает выделять фрагмент содержимого элемента страницы.

Действие по умолчанию: выделение фрагмента содержимого элемента страницы.

Поддерживается IE начиная с 4.0.

onstart

Наступает, когда текст в элементе <MARQUEE> начинает двигаться. Доступно, только если свойство loop установлено в 1.

Действие по умолчанию: начало движения. Не "всплывает" и не прерывается присвоением значения false свойству returnValue.

Поддерживается IE начиная с 4.0.

onstop

Наступает, когда пользователь прерывает загрузку текущей Web-страницы, нажав кнопку **Стоп** или покинув ее.

Действие по умолчанию: вызов функции-обработчика, если таковая есть. Не "всплывает" и не прерывается присвоением значения false свойству returnValue.

Поддерживается IE начиная с 5.0.

onsubmit

Наступает перед отправкой данных формой серверному приложению.

Действие по умолчанию: оправка введенных в форму данных. Не "всплывает".

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

onunload

Наступает перед выгрузкой текущей Web-страницы.

Действие по умолчанию: выгрузка страницы. Не "всплывает" и не прерывается присвоением значения `false` свойству `returnValue`.

Поддерживается IE начиная с 3.02 для `window` и начиная с 4.0 для `<BODY>` и `<FRAMESET>`. Поддерживается N начиная с 2.0.

Команды execCommand

В данном разделе перечислены в алфавитном порядке наименования команд, передаваемые в качестве первого параметра методу `execCommand`. Упоминаемые при описании команд параметры передаются методу `execCommand` в качестве третьего параметра.

2D-Position

Позволяет пользователю переместить элемент страницы на другое место с помощью drag-n-drop.

Требует задания строкового параметра. Значение "`on`" разрешает перемещать элемент, "`off`" — запрещает.

Поддерживается IE начиная с 5.5.

AbsolutePosition

Устанавливает свойство `position` элемента страницы в "`absolute`".

Требует задания логического параметра. Значение `true` позиционирует элемент абсолютно, `false` — статично.

Поддерживается IE начиная с 5.5.

BackColor

Задает или возвращает фоновый цвет элемента страницы.

Требует задания строкового параметра: имени цвета или шестизначного шестнадцатеричного значения. В последнем случае знак # не обязателен.

Поддерживается IE начиная с 4.0.

Bold

Делает шрифт выделенного текста жирным. При повторном применении убирает "жирность".

Поддерживается IE начиная с 4.0.

Copy

Копирует выделенный фрагмент страницы в Буфер обмена.

Поддерживается IE начиная с 4.0.

CreateBookmark

Создает "якорь" из выделенного текста или возвращает имя "якоря" для текста, на котором стоит текстовый курсор.

Требует задания строкового параметра — имени создаваемого "якоря".

Поддерживается IE начиная с 4.0.

CreateLink

Создает гиперссылку из выделенного текста.

Возможно задание строкового параметра — интернет-адреса гиперссылки.

Может отобразить диалоговое окно задания интернет-адреса.

Поддерживается IE начиная с 4.0.

Cut

Перемещает выделенный фрагмент страницы в Буфер обмена.

Поддерживается IE начиная с 4.0.

Delete

Удаляет выделенный фрагмент страницы.

Поддерживается IE начиная с 4.0.

FontName

Задает или возвращает имя шрифта текста.

Требует задания строкового параметра — имени шрифта (или списка имен шрифтов).

Поддерживается IE начиная с 4.0.

FontSize

Задает или возвращает размер шрифта текста.

Требует задания строкового или числового параметра — размера шрифта (от 1 до 7).

Поддерживается IE начиная с 4.0.

ForeColor

Задает или возвращает цвет текста элемента страницы.

Требует задания строкового параметра: имени цвета или шестизначного шестнадцатеричного значения. В последнем случае знак # не обязателен.

Поддерживается IE начиная с 4.0.

FormatBlock

Форматирует выделенный фрагмент как блок текста <DIV>, <P> или <SPAN>.

Требует задания строкового параметра — имени тега блока.

Поддерживается IE начиная с 4.0.

Indent

Форматирует выделенный фрагмент с отступом (тег <BLOCKQUOTE>).

Поддерживается IE начиная с 4.0.

InsertButton

Создает кнопку <BUTTON>. При этом выделенный фрагмент текста становится ее содержимым.

Возможно задание строкового параметра — значения атрибута ID создаваемой кнопки.

Поддерживается IE начиная с 4.0.

InsertFieldset

Создает группу элементов управления <FIELDSET>. При этом выделенный фрагмент текста становится ее содержимым.

Возможно задание строкового параметра — значения атрибута ID создаваемой группы.

Поддерживается IE начиная с 4.0.

InsertHorizontalRule

Создает горизонтальную линейку.

Возможно задание строкового параметра — значения атрибута ID создаваемой горизонтальной линейки.

Поддерживается IE начиная с 4.0.

InsertIFrame

Создает "плавающий" фрейм. При этом выделенный фрагмент текста становится его содержимым.

Возможно задание строкового параметра — значения атрибута ID создаваемого фрейма.

Поддерживается IE начиная с 4.0.

InsertImage

Создает графическое изображение.

Возможно задание строкового параметра — интернет-адреса и имени файла изображения.

Может отобразить диалоговое окно задания интернет-адреса файла изображения.

Поддерживается IE начиная с 5.0.

InsertInputButton

Создает командную кнопку.

Возможно задание строкового параметра — значения атрибута ID создаваемой кнопки.

Поддерживается IE начиная с 4.0.

InsertInputCheckbox

Создает флажок.

Возможно задание строкового параметра — значения атрибута ID создаваемого флажка.

Поддерживается IE начиная с 4.0.

InsertInputFileUpload

Создает элемент управления, позволяющий послать серверной программе файл.

Возможно задание строкового параметра — значения атрибута ID создаваемого элемента управления.

Поддерживается IE начиная с 4.0.

InsertInputHidden

Создает скрытое поле.

Возможно задание строкового параметра — значения атрибута ID создаваемого скрытого поля.

Поддерживается IE начиная с 4.0.

InsertInputImage

Создает графическое изображение — элемент управления.

Возможно задание строкового параметра — значения атрибута ID создаваемого изображения.

Поддерживается IE начиная с 4.0.

InsertInputPassword

Создает поле ввода пароля.

Возможно задание строкового параметра — значения атрибута ID создаваемого поля ввода.

Поддерживается IE начиная с 4.0.

InsertInputRadio

Создает радиокнопку.

Возможно задание строкового параметра — значения атрибута ID создаваемой радиокнопки.

Поддерживается IE начиная с 4.0.

InsertInputReset

Создает кнопку `reset` сброса данных формы.

Возможно задание строкового параметра — значения атрибута `ID` создаваемой кнопки.

Поддерживается IE начиная с 4.0.

InsertInputSubmit

Создает кнопку `sumbit` отправки данных формы.

Возможно задание строкового параметра — значения атрибута `ID` создаваемой кнопки.

Поддерживается IE начиная с 4.0.

InsertInputText

Создает обычное поле ввода.

Возможно задание строкового параметра — значения атрибута `ID` создаваемого поля ввода.

Поддерживается IE начиная с 4.0.

InsertMarquee

Создает пустой элемент `<MARQUEE>`.

Возможно задание строкового параметра — значения атрибута `ID` создаваемого элемента.

Поддерживается IE начиная с 4.0.

InsertOrderedList

Создает нумерованный список из выделенного текста.

Возможно задание строкового параметра — значения атрибута `ID` создаваемого списка.

Поддерживается IE начиная с 4.0.

InsertParagraph

Создает абзац из выделенного текста.

Возможно задание строкового параметра — значения атрибута `ID` создаваемого абзаца.

Поддерживается IE начиная с 4.0.

InsertSelectDropdown

Создает всплывающее меню. Выделенные строки становятся его пунктами.

Возможно задание строкового параметра — значения атрибута `ID` создаваемого меню.

Поддерживается IE начиная с 4.0.

InsertSelectListbox

Создает список. Выделенные строки становятся его пунктами.

Возможно задание строкового параметра — значения атрибута `ID` создаваемого списка.

Поддерживается IE начиная с 4.0.

InsertTextArea

Создает область редактирования текста.

Возможно задание строкового параметра — значения атрибута `ID` создаваемой области редактирования.

Поддерживается IE начиная с 4.0.

InsertUnorderedList

Создает маркированный список из выделенного текста.

Возможно задание строкового параметра — значения атрибута `ID` создаваемого списка.

Поддерживается IE начиная с 4.0.

Italic

Делает шрифт выделенного текста курсивом. При вторичном применении делает шрифт обычным.

Поддерживается IE начиная с 4.0.

JustifyCenter

Выравнивает абзац, на котором стоит текстовый курсор, по центру.

Поддерживается IE начиная с 4.0.

JustifyLeft

Выравнивает абзац, на котором стоит текстовый курсор, по левому краю.

Поддерживается IE начиная с 4.0.

JustifyRight

Выравнивает абзац, на котором стоит текстовый курсор, по правому краю.

Поддерживается IE начиная с 4.0.

LiveResize

Включает или отключает режим "живого" отображения изменения размеров или местоположения элемента страницы.

Требует задания строкового параметра. Значение "on" включает режим "живого" отображения, "off" — отключает. В последнем случае новое местоположение или новые размеры элемента страницы отобразятся только по окончании их изменения пользователем.

Поддерживается IE начиная с 5.5.

MultipleSelection

Включает или отключает режим множественного выделения. Если этот режим включен, пользователь может выделить одновременно несколько фрагментов Web-страницы, удерживая клавишу <Shift> или <Ctrl>.

Требует задания строкового параметра. Значение "on" включает режим множественного выделения, "off" — отключает.

Поддерживается IE начиная с 5.5.

Outdent

Уменьшает отступ (тег <BLOCKQUOTE>) у выделенного фрагмента.

Поддерживается IE начиная с 4.0.

MultipleSelection

Переключает режим ввода текста между вставкой и заменой.

Требует задания логического параметра. Значения true или null (а также отсутствие параметра) включает режим замены, false — режим вставки.

Поддерживается IE начиная с 4.0.

Paste

Копирует содержимое Буфера обмена на место выделенного фрагмента страницы.

Поддерживается IE начиная с 4.0.

Refresh

Вызывает обновление Web-страницы в окне Web-обозревателя.

Поддерживается IE начиная с 4.0.

RemoveFormat

Удаляет все HTML-форматирование из выделенного фрагмента.

Поддерживается IE начиная с 4.0.

SaveAs

Сохраняет текущую Web-страницу в файле.

Возможно задание строкового параметра — пути и имени файла, в котором будет сохранена Web-страница. Если путь содержит более чем одно имя папки, разделите их знаком \\.

Может отобразить диалоговое окно выбора пути и имени файла для сохранения.

Поддерживается IE начиная с 4.0.

SelectAll

Выделяет все содержимое Web-страницы.

Поддерживается IE начиная с 4.0.

UnBookmark

Удаляет "якорь" из текста, на котором находится текстовый курсор.

Поддерживается IE начиная с 4.0.

Underline

Делает шрифт выделенного текста подчеркнутым. При вторичном применении убирает подчеркивание.

Поддерживается IE начиная с 4.0.

UnBookmark

Удаляет гиперссылку, на которой находится текстовый курсор, превращая ее в обычный текст.

Поддерживается IE начиная с 4.0.

Unselect

Убирает выделение.

Поддерживается IE начиная с 4.0.

Объекты

Список объектов в алфавитном порядке. Для каждого объекта приводятся списки свойств, методов, событий, объектов и коллекций.

<!-- -->

Вставляет в код HTML невидимые комментарии.

Свойства: `innerHTML`, `outerHTML`, `tagName`, `text`.

События: `onlayoutcomplete`.

Поддерживается IE начиная с 4.0.

<A> (IE)

Задает либо гиперссылку, либо "якорь" для ссылки.

Свойства: `accessKey`, `canHaveChildren`, `canHaveHTML`, `className`, `clientHeight`, `clientLeft`, `clientTop`, `clientWidth`, `contentEditable`, `dataFld`, `dataSrc`, `dir`, `disabled`, `firstChild`, `hash`, `hideFocus`, `host`, `hostname`, `href`, `id`, `innerHTML`, `innerText`, `isContentEditable`, `isDisabled`, `isTextEdit`, `lang`, `language`, `lastChild`, `Methods`, `name`, `nameProp`, `nextSibling`, `nodeName`, `nodeType`, `nodeValue`, `offsetHeight`, `offsetLeft`, `offsetParent`, `offsetTop`, `offsetWidth`, `outerHTML`, `outerText`, `parentElement`, `parentNode`, `parentTextEdit`, `pathname`, `port`, `previousSibling`, `protocol`, `readyState`, `recordNumber`, `rel`, `rev`, `scopeName`, `scrollHeight`, `scrollLeft`, `scrollTop`, `scrollWidth`, `search`, `sourceIndex`, `STYLE`, `tabIndex`, `tagName`, `tagUrn`, `target`, `title`, `uniqueID`, `urn`.

Методы: `addBehavior`, `appendChild`, `applyElement`, `attachEvent`, `blur`, `clearAttributes`, `click`, `cloneNode`, `componentFromPoint`, `contains`, `detachEvent`, `dragDrop`, `fireEvent`, `focus`, `getAdjacentText`, `getAttribute`, `getBoundingClientRect`, `getClientRects`, `getElementsByTagName`, `getExpression`, `hasChildNodes`, `insertAdjacentElement`, `insertAdjacentHTML`,

insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 3.0.

<A NAME="..."> (N)

Задает "якорь" для ссылки.

Свойства: name, text, x, y.

Поддерживается N начиная с 2.0; свойства были добавлены в 4.0.

<A HREF="..."> (N)

Задает гиперссылку.

Свойства: hash, host, hostname, href, pathname, port, protocol, search, target, text, x, y.

Методы: handleEvent.

События: onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmouseout, onmouseover, onmouseup.

Поддерживается N начиная с 2.0; событие onmouseout добавлено в 3.0; свойства x и y и метод handleEvent были добавлены в 4.0.

<ACRONYM>

Используется для создания условных обозначений.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight,

offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, cloneNode, componentFromPoint, detachEvent, fireEvent, focus, getAdjacentText, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertBefore, mergeAttributes, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, setActive, setExpression, swapNode.

События: onactivate, onbeforedeactivate, onblur, oncontrolselect, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onkeydown, onkeypress, onkeyup, onmouseenter, onmouseleave, onreadystatechange, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<ADDRESS>

Используется для размещения на странице адресов, предупреждений об авторских правах и т. п.

Свойства: accessKey, blockDirection, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<APPLET> (IE)

Помещает на страницу Java-апплет.

Свойства: accessKey, altHTML, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, code, codeBase, dataFld, dataSrc, firstChild, hideFocus, hspace, id, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, name, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, src, STYLE, tabIndex, tagName, tagUrn, title, uniqueID, vspace.

Методы: addBehavior, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, hasChildNodes, insertAdjacentElement, mergeAttributes, namedRecordset, releaseCapture, removeAttribute, removeBehavior, removeExpression, replaceAdjacentText, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, oncellchange, onclick, oncontextmenu, oncontrolselect, oncut, ondataavailable, ondatasetchanged, ondatasetcomplete, ondblclick, ondeactivate, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onload, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onrowenter, onrowexit, onrowsdelete, onrowsinserted, onscroll.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<APPLET> (N)

Помещает на страницу Java-апплет.

Своих свойств и методов не имеет.

Поддерживается N начиная с 3.0.

<AREA> (IE)

Задает "горячую" область карты-изображения.

Свойства: accessKey, alt, canHaveHTML, className, coords, dir, firstChild, hash, hideFocus, host, hostname, href, id, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, noHref, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, pathname, port, previousSibling, protocol, readyState, recordNumber, scopeName, search, shape, sourceIndex, STYLE, tabIndex, tagName, tagUrn, target, title, uniqueID.

Методы: addBehavior, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeExpression, replaceAdjacentText, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<AREA> (N)

Задает "горячую" область карты-изображения.

Свойства: hash, host, hostname, href, pathname, port, protocol, search, target, text, x, y.

Методы: handleEvent.

События: ondblclick, onmouseout, onmouseover.

Поддерживается N начиная с 3.0; свойства x и y и метод handleEvent были добавлены в 4.0.

Attribute

Представляет доступ к атрибуту HTML-тега. Доступен как элемент коллекции attributes.

Свойства: nodeName, nodeType, nodeValue, specified.

Поддерживается IE начиная с 5.0.

<B>

Используется для выделения текста жирным шрифтом.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<BASE>

Задает базовый интернет-адрес, относительно которого будут вычисляться все относительные интернет-адреса, встречающиеся в коде страницы.

Свойства: canHaveHTML, firstChild, href, id, isContentEditable, isDisabled, isTextEdit, lastChild, layoutGrid, nextSibling, nodeName, nodeType, nodeValue, parentElement, parentNode, parentTextEdit, previousSibling, readyState, scopeName, sourceIndex, tagName, tagUrn, target, uniqueID.

Методы: addBehavior, applyElement, attachEvent, clearAttributes, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, hasChildNodes, insertAdjacentElement, mergeAttributes, removeAttribute, removeBehavior, replaceAdjacentText, setAttribute, swapNode.

События: onlayoutcomplete, onmouseenter, onmouseleave, onreadystatechange.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<BASEFONT>

Задает шрифт по умолчанию.

Свойства: canHaveHTML, color, face, firstChild, id, isContentEditable, isDisabled, isTextEdit, lastChild, nextSibling, nodeName, nodeType, nodeValue, parentElement, parentNode, parentTextEdit, previousSibling, readyState, scopeName, size, sourceIndex, tagName, tagUrn, uniqueID.

Методы: addBehavior, applyElement, attachEvent, clearAttributes, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, mergeAttributes, removeAttribute, removeBehavior, replaceAdjacentText, setAttribute, swapNode.

События: onlayoutcomplete, onmouseenter, onmouseleave, onreadystatechange.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<BDO>

Позволяет сменить порядок чтения для выделенного текста.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, tabIndex, tagName, tagUrn, title.

Методы: appendChild, applyElement, blur, clearAttributes, cloneNode, componentFromPoint, contains, fireEvent, focus, getAdjacentText, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertBefore, mergeAttributes, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, setActive, setExpression, swapNode.

События: onactivate, onafterupdate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, oncellchange, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onerrorupdate, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresizeend, onresizestart, onscroll, onselectstart.

Объекты: currentStyle.

Коллекции: all, attributes, childNodes, children, filters.

Поддерживается IE начиная с 5.0.

<BGSOUND>

Помещает на страницу звуковой файл, проигрываемый в фоновом режиме.

Свойства: balance, canHaveHTML, id, isContentEditable, isDisabled, loop, nextSibling, nodeName, nodeType, nodeValue, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, scopeName, sourceIndex, src, tagName, tagUrn, uniqueID, volume.

Методы: addBehavior, applyElement, attachEvent, clearAttributes, cloneNode, componentFromPoint, detachEvent, dragDrop, fireEvent, getAttribute, getElementsByTagName, insertAdjacentElement, mergeAttributes, removeAttribute, removeBehavior, setAttribute, swapNode.

События: onlayoutcomplete, onmouseenter, onmouseleave, onreadystatechange.

Объекты: currentStyle.

Коллекции: all, attributes, behaviorUrns.

Поддерживается IE начиная с 4.0.

<BIG>

Задает отображение текста крупным шрифтом.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<BLOCKQUOTE>

Форматирует текст как цитату.

Свойства: accessKey, blockDirection, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable,

dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<BODY>

Задает тело документа.

Свойства: accessKey, aLink, background, bgColor, bgProperties, blockDirection, bottomMargin, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, leftMargin, link, nextSibling, nodeName, nodeType, nodeValue, noWrap, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, parentElement, parentNode, parentTextEdit, previousSibling, readyState, rightMargin, scopeName, scroll, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, text, title, topMargin, uniqueID, vLink.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, createControlRange, createTextRange, detachEvent, doScroll, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onafterprint, onbeforecut, onbeforedeactivate, onbeforepaste, onbeforeprint, onbeforeunload, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfilterchange, onkeydown, onkeypress, onkeyup, onload, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresizeend, onresizestart, onscroll, onselect, onselectstart, onunload.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children, filters.

Поддерживается IE начиная с 4.0.

Вставляет "жесткий" разрыв строки.

Свойства: canHaveHTML, className, clear, id, isContentEditable, isDisabled, isTextEdit, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, sourceIndex, STYLE, tabIndex, tagName, uniqueID.

Методы: addBehavior, applyElement, attachEvent, clearAttributes, cloneNode, componentFromPoint, detachEvent, dragDrop, fireEvent, getAdjacentText, getAttribute, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeExpression, replaceAdjacentText, scrollIntoView, setAttribute, setCapture, setExpression, swapNode.

События: onlayoutcomplete, onlosecapture, onreadystatechange.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: attributes, behaviorUrns.

Поддерживается IE начиная с 4.0.

<BUTTON>

Помещает на странице кнопку с любым содержанием (таблица, рисунок, фрагмент текста с HTML-форматированием).

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dataFld, dataFormatAs, dataSrc, dir, disabled, firstChild, form, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, name, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, type, uniqueID, value.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, createTextRange, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondragenter, ondragleave, ondragover, ondrop, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children, filters.

Поддерживается IE начиная с 4.0.

<CAPTION>

Помещает в таблицу заголовок.

Свойства: accessKey, align, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop,

offsetWidth, outerHTML, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID, vAlign.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, d etachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<CENTER>

Используется для центрирования блока текста.

Свойства: accessKey, blockDirection, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText,

insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<CITE>

Используется для создания цитат.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove,

onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

clientInformation

Представляет сведения о программе Web-обозревателя.

Свойства: appCodeName, appMinorVersion, appName, appVersion, browserLanguage, cookieEnabled, cpuClass, onLine, platform, systemLanguage, userAgent, userLanguage.

Методы: javaEnabled, taintEnabled.

Коллекции: plugins.

Объекты: userProfile.

Поддерживается IE начиная с 4.0.

clipboardData

Позволяет получить доступ к данным, находящимся в Буфере обмена Windows.

Методы: clearData, getData, setData.

Поддерживается IE начиная с 5.0.

<CODE>

Используется для помещения на страницу примеров, скажем, исходных текстов программ.

Свойства: canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent,

fireEvent, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setAttribute, setCapture, setExpression, swapNode.

События: onbeforecopy, onbeforecut, onbeforepaste, onclick, oncontextmenu, oncopy, oncut, ondblclick, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<COL>

Задает параметры одной или нескольких колонок таблицы.

Свойства: align, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, dir, firstChild, id, innerHTML, isContentEditable, isDisabled, isTextEdit, lang, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, span, STYLE, tagName, tagUrn, uniqueID, vAlign.

Методы: addBehavior, appendChild, applyElement, attachEvent, clearAttributes, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertBefore, mergeAttributes, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setAttribute, setExpression, swapNode.

События: onlayoutcomplete, onreadystatechange.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<COLGROUP>

Задает параметры группы колонок таблицы.

Свойства: align, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, dir, firstChild, id, innerHTML, isContentEditable, isDisabled, isTextEdit, lang, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, span, STYLE, tagName, tagUrn, uniqueID, vAlign.

Методы: addBehavior, appendChild, applyElement, attachEvent, clearAttributes, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertBefore, mergeAttributes, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setAttribute, setExpression, swapNode.

События: onreadystatechange.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<COMMENT>

Вставляет в код HTML невидимые комментарии.

Свойства: canHaveChildren, canHaveHTML, firstChild, id, isContentEditable, isDisabled, isTextEdit, lang, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetParent, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, sourceIndex, tagName, tagUrn, text, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, clearAttributes, cloneNode, componentFromPoint, detachEvent, dragDrop, fireEvent, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, removeAttribute, removeBehavior, removeChild, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setAttribute, swapNode.

События: onpropertychange, onreadystatechange.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: attributes, behaviorUrns, childNodes.

Поддерживается IE начиная с 4.0.

crypto

Позволяет получить доступ к криптографической подсистеме Web-обозревателя.

Методы: random, signText.

Поддерживается N начиная с 4.0.

currentStyle

Представляет доступ к текущему стилю элемента страницы, включающему установки, заданные в таблицах, встроенных стилях и атрибутах.

Свойства: accelerator, backgroundAttachment, backgroundColor, backgroundImage, backgroundPositionX, backgroundPositionY, backgroundRepeat, behavior, blockDirection, borderBottomColor, borderBottomStyle, borderBottomWidth, borderCollapse, borderColor, borderLeftColor, borderLeftStyle, borderLeftWidth, borderRightColor, borderRightStyle, borderRightWidth, borderStyle, borderTopColor, borderTopStyle, borderTopWidth, borderWidth, bottom, clear, clipBottom, clipLeft, clipRight, clipTop, color, cursor, direction, display, fontFamily, fontSize, fontStyle, fontVariant, fontWeight, hasLayout, height, imeMode, layoutFlow, layoutGridChar, layoutGridLine, layoutGridMode, layoutGridType, left, letterSpacing, lineBreak, lineHeight, listStyleImage, listStylePosition, listStyleType, margin, marginBottom, marginLeft, marginRight, marginTop, overflow, overflowX, overflowY, padding, paddingBottom, paddingLeft, paddingRight, paddingTop, pageBreakAfter, pageBreakBefore, position, right, rubyAlign, rubyOverhang, rubyPosition, scrollbar3dLightColor, scrollbarArrowColor, scrollbarBaseColor, scrollbarDarkShadowColor, scrollbarFaceColor, scrollbarHighlightColor, scrollbarShadowColor, scrollbarTrackColor, styleFloat, tableLayout, textAlign, textAlignLast, textAutospace, textDecoration, textIndent, textJustify, textKashidaSpace, textTransform, textUnderlinePosition, top, unicodeBidi, verticalAlign, visibility, width, wordBreak, wordSpacing, wordWrap, writingMode, zIndex, zoom.

Методы: getAttribute, getExpression, setAttribute, setExpression.

Поддерживается IE начиная с 5.0.

custom

Пользователь может вставлять в код Web-страницы свои собственные теги, определенные или не определенные в пространстве имен. В последнем случае теги классифицируются Web-обозревателем как неизвестные и никак не обрабатываются. За объектное представление пользовательских тегов отвечает объект `custom`.

Свойства: `accessKey`, `blockDirection`, `canHaveChildren`, `canHaveHTML`, `className`, `clientHeight`, `clientLeft`, `clientTop`, `clientWidth`, `contentEditable`, `dir`, `disabled`, `hideFocus`, `id`, `innerHTML`, `innerText`, `isContentEditable`, `isDisabled`, `isTextEdit`, `lang`, `language`, `offsetHeight`, `offsetLeft`, `offsetParent`, `offsetTop`, `offsetWidth`, `outerHTML`, `outerText`, `parentElement`, `parentTextEdit`, `readyState`, `recordNumber`, `scopeName`, `scrollHeight`, `scrollLeft`, `scrollTop`, `scrollWidth`, `sourceIndex`, `STYLE`, `tabIndex`, `tagName`, `tagUrn`, `title`.

Методы: `addBehavior`, `applyElement`, `attachEvent`, `blur`, `clearAttributes`, `click`, `componentFromPoint`, `contains`, `detachEvent`, `doScroll`, `fireEvent`, `focus`, `getAdjacentText`, `getAttribute`, `getBoundingClientRect`, `getClientRects`, `getElementsByTagName`, `getExpression`, `insertAdjacentElement`, `insertAdjacentHTML`, `insertAdjacentText`, `mergeAttributes`, `releaseCapture`, `removeAttribute`, `removeBehavior`, `removeExpression`, `replaceAdjacentText`, `scrollIntoView`, `setActive`, `setAttribute`, `setCapture`, `setExpression`.

События: `onactivate`, `onafterupdate`, `onbeforecopy`, `onbeforecut`, `onbeforedeactivate`, `onbeforeeditfocus`, `onbeforepaste`, `onbeforeupdate`, `onblur`, `onclick`, `oncontextmenu`, `oncontrolselect`, `oncopy`, `oncut`, `ondblclick`, `ondeactivate`, `ondrag`, `ondragend`, `ondragenter`, `ondragleave`, `ondragover`, `ondragstart`, `ondrop`, `onfilterchange`, `onfocus`, `onhelp`, `onkeydown`, `onkeypress`, `onkeyup`, `onlosecapture`, `onmousedown`, `onmouseenter`, `onmouseleave`, `onmousemove`, `onmouseout`, `onmouseover`, `onmouseup`, `onpaste`, `onpropertychange`, `onreadystatechange`, `onresize`, `onresizeend`, `onresizestart`, `onscroll`, `onselectstart`.

Объекты: `currentStyle`, `document`, `runtimeStyle`, `style`.

Коллекции: `all`, `behaviorUrns`, `children`, `filters`.

Поддерживается IE начиная с 5.0.

dataTransfer

Позволяет получить доступ к данным, переносимым в ходе операции drag-n-drop.

Свойства: `dropEffect`, `effectAllowed`.

Методы: `clearData`, `getData`, `setData`.

Поддерживается IE начиная с 5.0.

<DD>

Помечает текст определения в списке определений <DL>.

Свойства: accessKey, blockDirection, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, noWrap, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlayoutcomplete, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

defaults

Позволяет установить значения свойств представления элемента страницы, используемые по умолчанию.

Свойства: blockDirection, canHaveHTML, contentEditable, disabled, tabStop, viewLink, viewMasterTab.

Объект: style.

Поддерживается IE начиная с 5.5.

<DEL>

Помечает удаленный из документа текст.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, contentEditable, dir, disabled, firstChild, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, cloneNode, componentFromPoint, detachEvent, fireEvent, focus, getAdjacentText, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertBefore, mergeAttributes, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, setExpression, swapNode.

События: onblur, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onkeydown, onkeypress, onkeyup, onreadystatechange, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<DFN>

Помечает описание термина.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText,

insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<DIR>

Используется для создания списка коротких строк, аналогичного списку файлов.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress,

onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<DIV>

Определяет отдельный элемент страницы: простой текстовый абзац или более сложный фрагмент с HTML-форматированием.

Свойства: accessKey, align, blockDirection, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dataFld, dataFormatAs, dataSrc, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, noWrap, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, doScroll, dragDrop, fireEvent, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlayoutcomplete, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onscroll, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children, filters.

Поддерживается IE начиная с 4.0.

<DL>

Используется для создания списка определений.

Свойства: accessKey, blockDirection, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, compact, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onlayoutcomplete, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

document (IE)

Позволяет получить доступ к свойствам текущей Web-страницы.

Свойства: activeElement, alinkColor, bgColor, cookie, defaultCharset, designMode, dir, documentElement, domain, expando, fgColor, fileCreatedDate, fileModifiedDate, fileSize, lastModified, linkColor, media, parentWindow, protocol, readyState, referrer, uniqueID, URL, URLUnencoded, vlinkColor, XMLDocument, XSLDocument.

Методы: attachEvent, clear, close, createElement, createEventObject, createStyleSheet, createTextNode, detachEvent, elementFromPoint, execCommand, focus, getElementById, getElementsByName, getElementsByTagName, hasFocus, mergeAttributes, open, queryCommandEnabled, queryCommandIndeterm, queryCommandState, queryCommandSupported, queryCommandValue, recalc, releaseCapture, setActive, write, writeln.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onhelp, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresizeend, onresizestart, onselectionchange, onstop.

Объекты: body, location, selection, title.

Коллекции: all, anchors, applets, childNodes, children, embeds, forms, frames, images, links, namespaces, scripts, styleSheets.

Поддерживается IE начиная с 3.02 для window и начиная с 5.0 для custom.

document (N)

Позволяет получить доступ к свойствам текущей Web-страницы.

Свойства: alinkColor, bgColor, cookie, domain, fgColor, height, lastModified, linkColor, referrer, title, URL, vlinkColor, width.

Методы: captureEvents, close, getSelection, handleEvent, open, releaseEvents, routeEvent, write, writeln.

События: onblur, onclick, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onload, onmousedown, onmouseup, onunload.

Коллекции: anchors, applets, contextual, classes, embeds, forms, ids, images, layers, links, plugins, tags.

Поддерживается N начиная с 2.0; свойство domain, события onblur и onfocus, коллекции applets, embeds, forms, images и plugins добавлены в 3.0; методы captureEvents, contextual, getSelection, handleEvent, releaseEvents и routeEvents, коллекции classes, ids, layers и tags добавлены в 4.0.

<DT>

Помечает текст термина в списке определений <DL>.

Свойства: accessKey, blockDirection, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName,

nodeType, nodeValue, noWrap, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlayoutcomplete, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<EM>

Выделяет текст терминов.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect,

getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<EMBED>

Помещает на страницу нетекстовый элемент, обрабатываемый с помощью расширения Web-обозревателя.

Свойства: accessKey, align, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, dir, firstChild, height, hidden, hideFocus, id, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, name, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, palette, parentElement, parentNode, parentTextEdit, pluginspage, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, src, STYLE, tagName, tagUrn, title, uniqueID, units, width.

Методы: addBehavior, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, hasChildNodes, insertAdjacentElement, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeExpression, replaceAdjacentText, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, onfocus, onhelp, onload, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup,

onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onscroll.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

event (IE)

Позволяет получить доступ к свойствам последнего наступившего события.

Свойства: altKey, altLeft, button, cancelBubble, clientX, clientY, contentOverFlow, ctrlKey, ctrlLeft, dataFld, fromElement, keyCode, nextPage, offsetX, offsetY, propertyName, qualifier, reason, recordset, repeat, returnValue, saveType, screenX, screenY, shiftKey, shiftLeft, srcElement, srcFilter, srcUrn, toElement, type, x, y.

Коллекции: bookmarks, boundElements.

Объект: dataTransfer.

Поддерживается IE начиная с 4.0.

event (N)

Позволяет получить доступ к свойствам последнего наступившего события.

Свойства: data, height, layerX, layerY, modifiers, pageX, pageY, screenX, screenY, target, type, which, width, x, y.

Поддерживается N начиная с 4.0.

external

Позволяет получить доступ к возможностям, предлагаемым дополнительными компонентами Web-обозревателя.

Свойства: menuArguments.

Методы: AddChannel, AddDesktopComponent, AddFavorite, AutoCompleteSaveForm, AutoScan, ImportExportFavorites, IsSubscribed, NavigateAndFind, ShowBrowserUI.

Поддерживается IE начиная с 4.0.

<FIELDSET>

Используется для объединения нескольких элементов управления в группу.

Свойства: accessKey, align, blockDirection, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth,

contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children, filters.

Поддерживается IE начиная с 4.0.

<FONT>

Изменяет шрифт содержимого текста.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, color, contentEditable, dir, disabled, face, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, size, sourceIndex, STYLE, tabIndex, tagName, tagUrn, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent,

dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlayoutcomplete, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<FORM> (IE)

Помещает на страницу Web-форму.

Свойства: action, blockDirection, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, encoding, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, length, method, name, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, target, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, item, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, reset, scrollIntoView, setActive, setAttribute, setCapture, setExpression, submit, swapNode, urns.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave,

ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onreset, onresize, onresizeend, onresizestart, onsubmit.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children, elements.

Поддерживается IE начиная с 3.0.

<FORM> (N)

Помещает на страницу Web-форму.

Свойства: action, encoding, length, method, name, target.

Методы: handleEvent, reset, submit.

События: onreset, onsubmit.

Коллекции: elements.

Поддерживается N начиная с 2.0; метод reset добавлен в 3.0; метод handleEvent добавлен в 4.0.

<FRAME>

Помещает фрейм в набор фреймов <FRAMESET>.

Свойства: allowTransparency, borderColor, canHaveHTML, className, contentWindow, dataFld, dataSrc, firstChild, frameBorder, height, hideFocus, id, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, marginHeight, marginWidth, name, nextSibling, nodeName, nodeType, nodeValue, noResize, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, parentElement, parentNode, parentTextEdit, previousSibling, readyState, scopeName, scrolling, self, sourceIndex, src, tabIndex, tagName, tagUrn, title, uniqueID, width.

Методы: addBehavior, applyElement, attachEvent, blur, clearAttributes, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, hasChildNodes, insertAdjacentElement, mergeAttributes, removeAttribute, removeBehavior, replaceAdjacentText, setActive, setAttribute, swapNode.

События: onactivate, onbeforedeactivate, onblur, oncontextmenu, ondeactivate, onfocus, onload, onresize, onresizeend, onresizestart.

Объекты: runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 3.0.

<FRAMESET>

Задает набор фреймов.

Свойства: border, borderColor, canHaveChildren, canHaveHTML, className, cols, firstChild, frameBorder, frameSpacing, hideFocus, id, innerHTML, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, name, nextSibling, nodeName, nodeType, nodeValue, outerHTML, parentElement, parentNode, parentTextEdit, previousSibling, readyState, rows, scopeName, sourceIndex, tabIndex, tagName, tagUrn, title, uniqueID, width.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getElementsByTagName, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, removeAttribute, removeBehavior, removeChild, removeNode, replaceAdjacentText, replaceChild, replaceNode, setActive, setAttribute, swapNode.

События: onactivate, onafterprint, onbeforedeactivate, onbeforeprint, onbeforeunload, onblur, oncontrolselect, ondeactivate, onfocus, onload, onresizeend, onresizestart, onunload.

Объекты: runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<HEAD>

Задает HTML-заголовок документа.

Свойства: canHaveChildren, canHaveHTML, className, firstChild, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, lastChild, nextSibling, nodeName, nodeType, nodeValue, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, tagName, tagUrn, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, clearAttributes, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, getAdjacentText, getAttribute, getElementsByTagName, hasChildNodes, insertAdjacentElement, insertBefore, mergeAttributes, removeAttribute, removeBehavior, removeChild, removeNode, replaceAdjacentText, replaceChild, replaceNode, setAttribute, swapNode.

События: onlayoutcomplete, onreadystatechange.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

history (IE)

Предоставляет доступ к списку "истории" Web-обозревателя, где хранятся ссылки на уже посещенные пользователем Web-страницы.

Свойство: `length`.

Методы: `back`, `forward`, `go`.

Поддерживается IE начиная с 3.02.

history (N)

Предоставляет доступ к списку "истории" Web-обозревателя, где хранятся ссылки на уже посещенные пользователем Web-страницы.

Свойства: `current`, `length`, `next`, `previous`.

Методы: `back`, `forward`, `go`.

Поддерживается N начиная с 3.02; свойства `current`, `next` и `previous` добавлены в 3.0.

<Hn>

Форматирует текст как заголовок n-го уровня.

Свойства: `accessKey`, `align`, `blockDirection`, `canHaveChildren`, `canHaveHTML`, `className`, `clientHeight`, `clientLeft`, `clientTop`, `clientWidth`, `contentEditable`, `dir`, `disabled`, `firstChild`, `hideFocus`, `id`, `innerHTML`, `innerText`, `isContentEditable`, `isDisabled`, `isTextEdit`, `lang`, `language`, `lastChild`, `nextSibling`, `nodeName`, `nodeType`, `nodeValue`, `offsetHeight`, `offsetLeft`, `offsetParent`, `offsetTop`, `offsetWidth`, `outerHTML`, `outerText`, `parentElement`, `parentNode`, `parentTextEdit`, `previousSibling`, `readyState`, `recordNumber`, `scopeName`, `scrollHeight`, `scrollLeft`, `scrollTop`, `scrollWidth`, `sourceIndex`, `STYLE`, `tabIndex`, `tagName`, `tagUrn`, `title`, `uniqueID`.

Методы: `addBehavior`, `appendChild`, `applyElement`, `attachEvent`, `blur`, `clearAttributes`, `click`, `cloneNode`, `componentFromPoint`, `contains`, `detachEvent`, `fireEvent`, `focus`, `getAdjacentText`, `getAttribute`, `getBoundingClientRect`, `getClientRects`, `getElementsByTagName`, `getExpression`, `hasChildNodes`, `insertAdjacentElement`, `insertAdjacentHTML`, `insertAdjacentText`, `insertBefore`, `mergeAttributes`, `releaseCapture`, `removeAttribute`, `removeBehavior`, `removeChild`, `removeExpression`, `removeNode`, `replaceAdjacentText`, `replaceChild`, `replaceNode`, `scrollIntoView`, `setActive`, `setAttribute`, `setCapture`, `setExpression`, `swapNode`.

События: `onactivate`, `onbeforecopy`, `onbeforecut`, `onbeforedeactivate`, `onbeforepaste`, `onblur`, `onclick`, `oncontextmenu`, `oncontrolselect`, `oncopy`, `oncut`, `ondblclick`, `ondeactivate`, `ondrag`, `ondragend`, `ondragenter`, `ondragleave`,

ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<HR>

Вставляет в текст Web-страницы горизонтальную линейку.

Свойства: accessKey, align, canHaveHTML, className, color, firstChild, hideFocus, id, isContentEditable, isDisabled, isTextEdit, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, noShade, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID, width.

Методы: addBehavior, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeExpression, replaceAdjacentText, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlayourcomplete, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<HTML>

Заключает в себя весь текст HTML-документа.

Свойства: canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, dir, firstChild, id, innerHTML, innerText,

isContentEditable, isDisabled, isTextEdit, lastChild, nextSibling, nodeName, nodeType, nodeValue, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, tagName, tagUrn, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, clearAttributes, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, getAdjacentText, getAttribute, getElementsByTagName, hasChildNodes, insertAdjacentElement, insertBefore, mergeAttributes, removeAttribute, removeBehavior, removeChild, removeNode, replaceAdjacentText, replaceChild, replaceNode, setAttribute, swapNode.

События: onlayoutcomplete, onmouseenter, onmouseleave, onreadystatechange.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

</>

Используется для выделения текста курсивом.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueIDh.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove,

onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<IFRAME>

Вставляет на страницу "плавающий" фрейм.

Свойства: align, allowTransparency, border, canHaveChildren, canHaveHTML, className, contentWindow, dataFld, dataSrc, firstChild, frameBorder, hideFocus, hspace, id, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, marginHeight, marginWidth, name, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrolling, sourceIndex, src, STYLE, tabIndex, tagName, tagUrn, title, uniqueID, vspace.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, removeAttribute, removeBehavior, removeChild, removeExpresion, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setExpression, swapNode.

События: onactivate, onbeforedeactivate, onblur, oncontrolselect, ondeactivate, onfocus, onload, onreadystatechange, onresizeend, onresizestart.

Объекты: runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<IMG> (IE)

Помещает на страницу рисунок.

Свойства: accessKey, align, alt, border, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, complete, dataFld, dataSrc, dir, dynsrc, fileCreatedDate, fileModifiedDate, fileSize, fileUpdatedDate, firstChild, height, hideFocus, hspace, id, isContentEditable, isDisabled,

isTextEdit, lang, language, lastChild, loop, lowsrc, name, nameProp, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, protocol, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, src, start, STYLE, tabIndex, tagName, tagUrn, title, uniqueID, useMap, vspace, width.

Методы: addBehavior, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeExpression, replaceAdjacentText, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onabort, onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onerror, onfilterchange, onfocus, onhelp, onload, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children, filters.

Поддерживается IE начиная с 4.0.

<IMG> (N)

Помещает на страницу рисунок.

Свойства: border, complete, height, hspace, lowsrc, name, src, vspace, width.

Методы: handleEvent.

События: onabort, onerror, onkeydown, onkeypress, onkeyup, onload.

Поддерживается IE начиная с 3.0; метод handleEvent добавлен в 4.0.

<INPUT TYPE="button"> (IE)

Помещает на страницу обычную командную кнопку.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dataFld, dataFormatAs, dataSrc, defaultValue, dir, disabled, firstChild, form, hideFocus, id,

isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, name, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, size, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, type, uniqueID, value, width.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, createTextRange, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: attributes, behaviorUrns, filters.

Поддерживается IE начиная с 3.0.

<INPUT TYPE="button"> (N)

Помещает на страницу обычную командную кнопку.

Свойства: form, name, type, value.

Методы: blur, click, focus, handleEvent.

События: onblur, onclick, onfocus, onmousedown, onmouseup.

Поддерживается N начиная с 2.0; свойство type, методы blur и focus, события onblur и onfocus добавлены в 3.0; метод handleEvent добавлен в 4.0.

<INPUT TYPE="checkbox"> (IE)

Помещает на страницу флажок.

Свойства: accessKey, canHaveChildren, canHaveHTML, checked, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dataFld,

dataSrc, defaultChecked, defaultValue, dir, disabled, firstChild, form, hideFocus, id, indeterminate, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, name, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, size, sourceIndex, status, STYLE, tabIndex, tagName, tagUrn, title, type, uniqueID, value, width.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, *removeBehavior*, *removeChild*, *removeExpression*, *removeNode*, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, select, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: attributes, behaviorUrns, filters.

Поддерживается IE начиная с 3.0.

<INPUT TYPE="checkbox"> (N)

Помещает на страницу флажок.

Свойства: checked, defaultChecked, form, name, type, value.

Методы: blur, click, focus, handleEvent.

События: onblur, onclick, onfocus.

Поддерживается N начиная с 2.0; свойство type, методы blur и focus, события onblur и onfocus добавлены в 3.0; метод handleEvent добавлен в 4.0.

<INPUT TYPE="file"> (IE)

Помещает на страницу поле ввода имени файла и кнопку вызова стандартного диалогового окна открытия файла.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dataFld, dataSrc, defaultValue, dir, disabled, firstChild, form, hideFocus, id, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, name, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, size, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, type, uniqueID, value, width.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, select, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: attributes, behaviorUrns, filters.

Поддерживается IE начиная с 4.0.

<INPUT TYPE="file"> (N)

Помещает на страницу поле ввода имени файла и кнопку вызова стандартного диалогового окна открытия файла.

Свойства: form, name, type, value.

Методы: blur, focus, handleEvent, select.

События: onblur, onclick, onfocus.

Поддерживается N начиная с 2.0; свойство type добавлено в 3.0; метод handleEvent добавлен в 4.0.

<INPUT TYPE="hidden"> (IE)

Помещает на страницу скрытое поле.

Свойства: canHaveHTML, className, dataFld, dataSrc, defaultValue, form, hideFocus, id, isContentEditable, isDisabled, isTextEdit, lang, language, name, nextSibling, nodeName, nodeType, nodeValue, offsetParent, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, sourceIndex, STYLE, tagName, tagUrn, type, uniqueID, value.

Методы: addBehavior, applyElement, attachEvent, clearAttributes, cloneNode, componentFromPoint, createTextRange, detachEvent, fireEvent, getAdjacentText, getAttribute, getExpression, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, replaceAdjacentText, replaceChild, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforedeactivate, onbeforeeditfocus, oncontrolselect, ondeactivate, onfocus, onlosecapture, onpropertychange, onreadystatechange, onresizeend, onresizestart.

Объекты: runtimeStyle, style.

Коллекции: attributes, behaviorUrns.

Поддерживается IE начиная с 3.0.

<INPUT TYPE="hidden"> (N)

Помещает на страницу скрытое поле.

Свойства: form, name, type, value.

Поддерживается N начиная с 2.0; свойство type добавлено в 3.0.

<INPUT TYPE="image">

Помещает на страницу изображение.

Свойства: accessKey, align, alt, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, complete, dataFld, dataSrc, defaultValue, dir, disabled, dynsrc, firstChild, form, hideFocus, hspace, id, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, loop, lowsrc, name, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth,

size, sourceIndex, src, start, STYLE, tabIndex, tagName, tagUrn, title, type, uniqueID, vspace, width.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, select, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: attributes, behaviorUrns, filters.

Поддерживается IE начиная с 3.0.

<INPUT TYPE="password"> (IE)

Помещает на страницу поле ввода пароля.

Свойства: accessKey, autocomplete, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dataFld, dataSrc, defaultValue, dir, disabled, firstChild, form, hideFocus, id, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, maxLength, name, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readOnly, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, size, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, type, uniqueID, value, vcard_name, width.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, createTextRange, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture,

removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, select, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: attributes, behaviorUrns, filters.

Поддерживается IE начиная с 3.0.

<INPUT TYPE="password"> (N)

Помещает на страницу поле ввода пароля.

Свойства: defaultValue, form, name, type, value.

Методы: blur, focus, handleEvent, select.

События: onblur, onfocus.

Поддерживается N начиная с 2.0; свойство type, события onblur и onfocus добавлены в 3.0; метод handleEvent добавлен в 4.0.

<INPUT TYPE="radio"> (IE)

Помещает на страницу радиокнопку.

Свойства: accessKey, canHaveChildren, canHaveHTML, checked, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dataFld, dataSrc, defaultChecked, defaultValue, dir, disabled, firstChild, form, hideFocus, id, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, name, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, size, sourceIndex, status, STYLE, tabIndex, tagName, tagUrn, title, type, uniqueID, value, width.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getExpression, hasChildNodes,

insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, select, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: attributes, behaviorUrns, filters.

Поддерживается IE начиная с 3.0.

<INPUT TYPE="radio"> (N)

Помещает на страницу радиокнопку.

Свойства: checked, defaultChecked, form, name, type, value.

Методы: blur, click, focus, handleEvent.

События: onblur, onclick, onfocus.

Поддерживается N начиная с 2.0; свойство type, методы blur и focus добавлены в 3.0; метод handleEvent добавлен в 4.0.

<INPUT TYPE="reset"> (IE)

Помещает на страницу командную кнопку сброса reset.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dataFld, dataSrc, defaultValue, dir, disabled, firstChild, form, hideFocus, id, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, name, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, size, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, type, uniqueID, value, width.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains,

createTextRange, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, select, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: attributes, behaviorUrns, filters.

Поддерживается IE начиная с 3.0.

<INPUT TYPE="reset"> (N)

Помещает на страницу командную кнопку сброса **reset**.

Свойства: form, name, type, value.

Методы: blur, click, focus, handleEvent.

События: onblur, onclick, onfocus.

Поддерживается N начиная с 2.0; свойство type, методы blur и focus, события onblur и onfocus добавлены в 3.0; метод handleEvent добавлен в 4.0.

<INPUT TYPE="submit"> (IE)

Помещает на страницу командную кнопку отправки данных submit.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dataFld, dataSrc, defaultValue, dir, disabled, firstChild, form, hideFocus, id, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, name, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, size, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, type, uniqueID, value, width.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, createTextRange, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, select, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: attributes, behaviorUrns, filters.

Поддерживается IE начиная с 3.0.

<INPUT TYPE="submit"> (N)

Помещает на страницу командную кнопку отправки данных submit.

Свойства: form, name, type, value.

Методы: blur, click, focus, handleEvent.

События: onblur, onclick, onfocus.

Поддерживается N начиная с 2.0; свойство type, методы blur и focus, события onblur и onfocus добавлены в 3.0; метод handleEvent добавлен в 4.0.

<INPUT TYPE="text"> (IE)

Помещает на страницу обычное поле ввода.

Свойства: accessKey, autocomplete, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dataFld, dataSrc, defaultValue, dir, disabled, firstChild, form, hideFocus, id, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, maxLength, name, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readOnly, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop,

scrollWidth, size, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, type, uniqueID, value, vcard_name, width.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, createTextRange, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, select, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onafterupdate, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onbeforeupdate, onblur, onchange, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onerrorupdate, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselect, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: attributes, behaviorUrns, filters.

Поддерживается IE начиная с 3.0.

<INPUT TYPE="text"> (N)

Помещает на страницу обычное поле ввода.

Свойства: defaultValue, form, name, type, value.

Методы: blur, focus, handleEvent, select.

События: onblur, onchange, onfocus, onselect.

Поддерживается N начиная с 2.0; свойство type добавлено в 3.0; метод handleEvent добавлен в 4.0.

<INS>

Помечает вновь добавленный в документ текст.

Свойства: accessKey, action, blockDirection, canHaveHTML, className, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight,

offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, cloneNode, componentFromPoint, detachEvent, fireEvent, focus, getAdjacentText, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertBefore, mergeAttributes, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, setActive, setExpression, swapNode.

События: onactivate, onbeforedeactivate, onblur, oncontrolselect, ondeactivate, onfocus, onreadystatechange, onresizeend, onresizestart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<ISINDEX>

Указывает Web-обозревателю вывести в окне поле ввода, в котором пользователь должен будет что-то ввести.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, disabled, hideFocus, id, isContentEditable, isDisabled, lang, language, parentElement, readyState, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, STYLE, tabIndex, tagUrn.

Методы: addBehavior, blur, componentFromPoint, dragDrop, fireEvent, focus, getBoundingClientRect, getClientRects, removeBehavior, setActive.

События: onactivate, onbeforedeactivate, onblur, oncontrolselect, ondeactivate, onfocus, onreadystatechange, onresize, onresizeend, onresizestart.

Объект: currentStyle.

Коллекция: behaviorUrns.

Поддерживается IE начиная с 4.0.

<KBD>

Выводит текст моноширинным шрифтом.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled,

firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<LABEL>

Задает текстовую метку элемента управления.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dataFld, dataFormatAs, dataSrc, dir, disabled, firstChild, hideFocus, htmlFor, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent,

dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<LAYER> и <ILAYER>

Помещает на страницу слой.

Свойства: above, background, below, bgColor, clip.bottom, clip.height, clip.right, clip.top, clip.width, document, left, name, pageX, pageY, parentLayer, siblingAbove, siblingBelow, src, top, visibility, window, x, y, zIndex.

Методы: captureEvents, handleEvent, load, moveAbove, moveBelow, moveBy, moveTo, moveToAbsolute, releaseEvents, resizeBy, resizeTo, routeEvent.

События: onblur, onfocus, onload, onmouseover, onmouseout.

Поддерживается N начиная с 4.0.

<LEGEND>

Задает текстовую метку группе элемента управления <FIELDSET>.

Свойства: accessKey, align, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<LI>

Помечает позицию списка.

Свойства: accessKey, blockDirection, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, type, uniqueID, value.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlayoutcomplete, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<LINK>

Устанавливает связь текущей страницы с другими файлами.

Свойства: canHaveHTML, disabled, firstChild, href, id, isContentEditable, isDisabled, isTextEdit, lastChild, name, nextSibling, nodeName, nodeType, nodeValue, parentElement, parentNode, parentTextEdit, previousSibling, readyState, rel, rev, scopeName, sourceIndex, tagName, tagUrn, type, uniqueID.

Методы: addBehavior, applyElement, attachEvent, clearAttributes, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, hasChildNodes, insertAdjacentElement, mergeAttributes, removeAttribute, removeBehavior, replaceAdjacentText, setAttribute, swapNode.

События: onload, onreadystatechange.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<LISTING>

Выводит блок текста моноширинным шрифтом.

Свойства: accessKey, blockDirection, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 3.0.

location

Содержит информацию об интернет-адресе.

Свойства: hash, host, hostname, href, pathname, port, protocol, search.

Методы: assign, reload, replace.

Поддерживается IE начиная с 3.02. Поддерживается N начиная с 2.0; методы reload и replace добавлены в 3.0; метод assign не поддерживается.

<MAP>

Задает список "горячих" областей карты-изображения.

Свойства: canHaveChildren, canHaveHTML, className, dir, firstChild, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, name, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, sourceIndex, STYLE, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, hasChildNodes, insertAdjacentElement,

insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setAttribute, setCapture, swapNode.

События: onbeforecut, onbeforepaste, onclick, oncut, ondblclick, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onscroll, onselectstart.

Коллекции: all, areas, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<MARQUEE>

Помещает на страницу прокручивающийся текст.

Свойства: accessKey, behavior, bgColor, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dataFld, dataFormatAs, dataSrc, dir, direction, disabled, firstChild, height, hideFocus, hspace, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, loop, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollAmount, scrollDelay, scrollHeight, scrollLeft, scrollTop, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, trueSpeed, uniqueID, vspace, width.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, start, stop, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onbounce, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfilterchange, onfinish, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onscroll, onselectstart, onstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children, filters.

Поддерживается IE начиная с 4.0.

<MENU>

Используется для создания неупорядоченного списка.

Свойства: accessKey, blockDirection, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<META>

Служит для размещения дополнительной информации о Web-странице: описания, списка ключевых слов для поисковых машин, сведений о программе, в которой была создана Web-страница, и т. п.

Свойства: `clientHeight`, `clientLeft`, `clientTop`, `clientWidth`, `content`, `httpEquiv`, `isTextEdit`, `name`, `parentTextEdit`, `scrollHeight`, `scrollLeft`, `scrollTop`, `scrollWidth`, `sourceIndex`, `tagName`.

Методы: `contains`, `dragDrop`, `getAttribute`, `removeAttribute`, `setAttribute`.

События: `onlayoutcomplete`.

Поддерживается IE начиная с 4.0.

MimeType

Представляет тип данных MIME, поддерживаемый Web-обозревателем. Доступен как элемент коллекции `mimeTypes`.

Свойства: `description`, `enabledPlugin`, `suffixes`, `type`.

Поддерживается N начиная с 3.0.

namespace

Предоставляет доступ к свойствам объекта поведения.

Свойства: `name`, `readyState`, `urn`.

Методы: `attachEvent`, `detachEvent`, `doImport`.

Событие: `onreadystatechange`.

Поддерживается IE начиная с 5.5.

navigator (IE)

Представляет сведения о программе Web-обозревателя.

Свойства: `appCodeName`, `appMinorVersion`, `appName`, `appVersion`, `browserLanguage`, `cookieEnabled`, `cpuClass`, `onLine`, `platform`, `systemLanguage`, `userAgent`, `userLanguage`.

Методы: `javaEnabled`, `taintEnabled`.

Коллекция: `plugins`.

Объект: `userProfile`.

Поддерживается IE начиная с 3.02.

navigator (N)

Представляет сведения о программе Web-обозревателя.

Свойства: `appCodeName`, `appName`, `appVersion`, `language`, `platform`, `userAgent`.

Методы: `javaEnabled`, `preference`, `taintEnabled`, `savePreference`.

Коллекции: mimeTypes, plugins.

Поддерживается N начиная с 2.0; коллекции mimeTypes и plugins, методы javaEnabled и taintEnabled добавлены в 3.0; свойства language и platform, методы preference и savePreferences добавлены в 4.0.

<NOBR>

Выводит текст в одну строку без разрывов.

Свойства: canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, lastChild, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentTextEdit, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, tagName, tagUrn, uniqueID.

Методы: addBehavior, attachEvent, click, componentFromPoint, contains, detachEvent, fireEvent, getAttribute, getBoundingClientRect, getClientRects, getExpression, insertAdjacentHTML, insertAdjacentText, releaseCapture, removeAttribute, removeBehavior, removeExpression, scrollIntoView, setAttribute, setCapture, setExpression.

События: onbeforecopy, onbeforecut, onbeforepaste, onclick, oncontextmenu, oncopy, oncut, ondblclick, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmouseenter, onmouseleave, onmousemove, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекция: behaviorUrns.

Поддерживается IE начиная с 4.0.

<NOFRAMES>

Задает текст, который будет выводиться Web-обозревателями, не поддерживающими фреймы.

Свойства: canHaveHTML, id, isContentEditable, isDisabled, parentElement, readyState, scopeName, tagUrn.

Методы: addBehavior, componentFromPoint, fireEvent, removeBehavior.

Событие: onreadystatechange.

Объекты: currentStyle, runtimeStyle, style.

Коллекция: behaviorUrns.

Поддерживается IE начиная с 4.0.

<NOSCRIPT>

Задает текст, который будет выводиться Web-обозревателями, не поддерживающими скрипты.

Свойства: canHaveHTML, id, isContentEditable, isDisabled, parentElement, readyState, scopeName, tagUrn.

Методы: addBehavior, componentFromPoint, fireEvent, removeBehavior.

Событие: onreadystatechange.

Объекты: currentStyle, runtimeStyle, style.

Коллекция: behaviorUrns.

Поддерживается IE начиная с 4.0.

<OBJECT>

Помещает на страницу внедренный объект: элемент ActiveX или Java-класс.

Свойства: accessKey, align, altHTML, BaseHref, canHaveChildren, canHaveHTML, classid, className, clientHeight, clientLeft, clientTop, clientWidth, code, codeBase, codeType, data, dataFld, dataSrc, dir, form, height, hideFocus, hspace, id, isContentEditable, isDisabled, isTextEdit, lang, language, name, nextSibling, nodeName, nodeType, nodeValue, object, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, recordset, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, type, uniqueID, vspace, width.

Методы: addBehavior, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getExpression, insertAdjacentElement, insertBefore, mergeAttributes, namedRecordset, releaseCapture, removeAttribute, removeBehavior, removeExpression, removeNode, replaceAdjacentText, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforedeactivate, onbeforeeditfocus, onblur, oncellchange, onclick, oncontrolselect, ondataavailable, ondatasetchanged, ondatasetcomplete, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onerror, onfocus, onkeydown, onkeypress, onkeyup, onlosecapture, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onrowenter, onrowexit, onrowsdelete, onrowsinserted, onscroll, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns.

Поддерживается IE начиная с 4.0.

<OL>

Используется для создания нумерованного списка.

Свойства: accessKey, blockDirection, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, compact, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, start, STYLE, tabIndex, tagName, tagUrn, title, type, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlayoutcomplete, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<OPTION> (IE)

Задает пункт списка или всплывающего меню <SELECT>.

Свойства: canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, defaultSelected, dir, firstChild, form, id, index,

innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, selected, tagName, tagUrn, uniqueID, value.

Методы: addBehavior, appendChild, applyElement, attachEvent, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, setAttribute, setCapture, setExpression, swapNode.

События: onlayoutcomplete, onlosecapture, onpropertychange, onreadystatechange, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 3.0.

<OPTION> (N)

Задает пункт списка или всплывающего меню <SELECT>.

Свойства: defaultSelected, index, length, selected, text, value.

Поддерживается N начиная с 2.0; свойство defaultSelected было добавлено в 3.0.

<P>

Определяет текстовый абзац.

Свойства: accessKey, align, blockDirection, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent,

dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlayoutcomplete, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

page

Представляет правило @page в таблице стиля.

Свойства: pseudoClass, selector.

Поддерживается IE начиная с 5.5.

<PLAINTEXT>

Выводит блок текста моноширинным шрифтом без обработки HTML-тегов.

Свойства: accessKey, blockDirection, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getElementsByTagName, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertBefore,

mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

plugin

Представляет инсталлированное расширение Web-обозревателя. Доступен как элемент коллекции navigator.plugins.

Свойства: description, filename, length, name.

Поддерживается N начиная с 3.0.

popup

Представляет всплывающее окно Web-обозревателя. Всплывающее окно — это особый тип окна, используемый для диалогов и предупреждений.

Свойства: document, isOpen.

Методы: hide, show.

Поддерживается IE начиная с 5.5.

<PRE>

Выводит блок текста моноширинным шрифтом с сохранением всего форматирования.

Свойства: accessKey, blockDirection, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState,

recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID, wrap.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<Q>

Форматирует текст как цитату.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, cloneNode, componentFromPoint, detachEvent, fireEvent, focus, getAdjacentText, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertBefore, mergeAttributes, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, setActive, setExpression, swapNode.

События: onactivate, onbeforedeactivate, onblur, oncontrolselect, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onkeydown, onkeypress, onkeyup, onreadystatechange, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<RT>

Задает текст аннотации для элемента <RUBY>.

Свойства: accessKey, canHaveHTML, className, contentEditable, dir, disabled, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, lang, language, name, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, readyState, scopeName, STYLE, tabIndex, tagName, tagUrn, title.

Методы: addBehavior, blur, componentFromPoint, fireEvent, focus, getExpression, removeBehavior, removeExpression, setActive, setExpression.

События: onactivate, onafterupdate, onbeforecut, onbeforedeactivate, onbeforepaste, onbeforeupdate, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondragstart, onerrorupdate, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onreadystatechange, onresizeend, onresizestart, onselectstart.

Коллекции: behaviorUrns, children, filters.

Поддерживается IE начиная с 5.0.

<RUBY>

Применяется для создания аннотации к какому-либо фрагменту текста.

Свойства: accessKey, canHaveHTML, className, contentEditable, dir, disabled, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, lang, language, name, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, padding, parentElement, readyState, recordNumber, scopeName, STYLE, tabIndex, tagName, tagUrn, title, unicodeBidi.

Методы: addBehavior, blur, componentFromPoint, fireEvent, focus, getExpression, removeBehavior, removeExpression, setActive, setExpression.

События: onactivate, onafterupdate, onbeforecut, onbeforedeactivate, onbeforepaste, onbeforeupdate, onblur, onclick, oncontextmenu,

oncontrolselect, oncut, ondblclick, ondeactivate, ondragstart, onerrorupdate, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onreadystatechange, onresizeend, onresizestart, onselectstart.

Коллекции: behaviorUrns, children, filters.

Поддерживается IE начиная с 5.0.

rule

Представляет описание стиля в таблице стилей, сделанное для каких-либо тегов. Доступен как элемент коллекции rules.

Свойства: readOnly, selectorText.

Объект: style.

Поддерживается IE начиная с 5.0.

runtimeStyle

Представляет доступ к текущему стилю элемента страницы, включающему установки, заданные в таблицах, встроенных стилях и атрибутах. При присвоении новых значений его свойствам значения аналогичных свойств объекта style не изменяются.

Свойства: accelerator, background, backgroundAttachment, backgroundColor, backgroundImage, backgroundPosition, backgroundPositionX, backgroundPositionY, backgroundRepeat, behavior, blockDirection, border, borderBottom, borderBottomColor, borderBottomStyle, borderBottomWidth, borderCollapse, borderColor, borderLeft, borderLeftColor, borderLeftStyle, borderLeftWidth, borderRight, borderRightColor, borderRightStyle, borderRightWidth, borderStyle, borderTop, borderTopColor, borderTopStyle, borderTopWidth, borderWidth, bottom, clear, clip, color, cssText, cursor, direction, font, fontFamily, fontSize, fontStyle, fontVariant, fontWeight, height, imeMode, layoutFlow, layoutGrid, layoutGridChar, layoutGridLine, layoutGridMode, layoutGridType, left, letterSpacing, lineBreak, lineHeight, listStyle, listStyleImage, listStylePosition, listStyleType, margin, marginBottom, marginLeft, marginRight, marginTop, overflow, overflowX, overflowY, padding, paddingBottom, paddingLeft, paddingRight, paddingTop, pageBreakAfter, pageBreakBefore, pixelBottom, pixelHeight, pixelLeft, pixelRight, pixelTop, pixelWidth, posBottom, posHeight, posLeft, position, posRight, posTop, posWidth, right, rubyAlign, rubyOverhang, rubyPosition, scrollbar3dLightColor, scrollbarArrowColor, scrollbarBaseColor,

scrollbarDarkShadowColor, scrollbarFaceColor, scrollbarHighlightColor, scrollbarShadowColor, scrollbarTrackColor, styleFloat, tableLayout, textAlign, textAlignLast, textAutospace, textDecoration, textDecorationBlink, textDecorationLineThrough, textDecorationNone, textDecorationOverline, textDecorationUnderline, textIndent, textJustify, textKashidaSpace, textTransform, textUnderlinePosition, top, unicodeBidi, verticalAlign, visibility, width, wordBreak, wordSpacing, wordWrap, writingMode, zIndex, zoom.

Методы: getAttribute, getExpression, removeAttribute, removeExpression, setAttribute, setExpression.

Поддерживается IE начиная с 5.0.

<S>

Выводит текст зачеркнутым.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<SAMP>

Форматирует текст как пример исходного кода программы.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

screen (IE)

Содержит информацию о видеоподсистеме клиентского компьютера.

Свойства: availHeight, availWidth, bufferDepth, colorDepth, fontSmoothingEnabled, height, updateInterval, width.

Поддерживается IE начиная с 4.0.

screen (N)

Содержит информацию о видеоподсистеме клиентского компьютера.

Свойства: availHeight, availLeft, availTop, availWidth, colorDepth, height, pixelDepth, width.

Поддерживается N начиная с 4.0.

<SCRIPT>

Используется для помещения на страницу скриптовой программы, исполняемой Web-обозревателем.

Свойства: canHaveHTML, clientHeight, clientLeft, clientTop, clientWidth, defer, event, firstChild, htmlFor, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, src, tagName, tagUrn, text, type, uniqueID.

Методы: addBehavior, applyElement, attachEvent, clearAttributes, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, getAdjacentText, getAttribute, getElementsByTagName, hasChildNodes, insertAdjacentElement, mergeAttributes, removeAttribute, removeBehavior, replaceAdjacentText, setAttribute, swapNode.

События: onload, onpropertychange, onreadystatechange.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<SELECT> (IE)

Помещает на страницу всплывающее меню или список с набором пунктов.

Свойства: accessKey, align, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, dataFld, dataSrc, dir, disabled, firstChild, form, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, length, multiple, name, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, selectedIndex, size, sourceIndex, STYLE, tabIndex, tagName, tagUrn, type, uniqueID, value.

Методы: addBehavior, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode, urns.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onchange, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondragenter, ondragleave, ondragover, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children, filters, options.

Поддерживается IE начиная с 3.0.

<SELECT> (N)

Помещает на страницу всплывающее меню или список с набором пунктов.

Свойства: form, length, name, selectedIndex, type.

Методы: blur, focus, handleEvent.

События: onblur, onchange, onfocus.

Коллекция: options.

Поддерживается N начиная с 2.0; свойство type добавлено в 3.0; метод handleEvent добавлен в 4.0.

selection

Предоставляет доступ к выделенному фрагменту Web-страницы.

Свойства: type, typeDetail.

Методы: clear, createRange, createRangeCollection, empty.

Коллекции: коллекция объектов TextRange.

Поддерживается IE начиная с 4.0.

<SMALL>

Задает отображение текста мелким шрифтом.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<SPAN>

Определяет встроенный элемент страницы, например, фрагмент текстового абзаца.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dataFld, dataFormatAs, dataSrc, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft,

offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, doScroll, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children, filters.

Поддерживается IE начиная с 4.0.

<STRIKE>

Выводит текст зачеркнутым. Аналогичен <S>.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes,

insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<STRONG>

Используется для выделения текста.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress,

onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<STYLE>

Задает таблицу стиля для Web-страницы.

Свойства: disabled, innerHTML.

Методы: addBehavior, dragDrop, removeBehavior, removeNode.

События: onerror, onreadystatechange.

Коллекция: behaviorUrns.

Поддерживается IE начиная с 4.0.

style (IE)

Представляет встроенный стиль элемента страницы.

Свойства: accelerator, background, backgroundAttachment, backgroundColor, backgroundImage, backgroundPosition, backgroundPositionX, backgroundPositionY, backgroundRepeat, behavior, blockDirection, border, borderBottom, borderBottomColor, borderBottomStyle, borderBottomWidth, borderCollapse, borderColor, borderLeft, borderLeftColor, borderLeftStyle, borderLeftWidth, borderRight, borderRightColor, borderRightStyle, borderRightWidth, borderStyle, borderTop, borderTopColor, borderTopStyle, borderTopWidth, borderWidth, bottom, clear, clip, color, cssText, cursor, direction, display, font, fontFamily, fontSize, fontStyle, fontVariant, fontWeight, height, imeMode, layoutFlow, layoutGrid, layoutGridChar, layoutGridLine, layoutGridMode, layoutGridType, left, letterSpacing, lineBreak, lineHeight, listStyle, listStyleImage, listStylePosition, listStyleType, margin, marginBottom, marginLeft, marginRight, marginTop, overflow, overflowX, overflowY, padding, paddingBottom, paddingLeft, paddingRight, paddingTop, pageBreakAfter, pageBreakBefore, pixelBottom, pixelHeight, pixelLeft, pixelRight, pixelTop, pixelWidth, posBottom, posHeight, posLeft, position, posRight, posTop, posWidth, right, rubyAlign, rubyOverhang, rubyPosition, scrollbar3dLightColor, scrollbarArrowColor, scrollbarBaseColor, scrollbarDarkShadowColor, scrollbarFaceColor, scrollbarHighlightColor, scrollbarShadowColor, scrollbarTrackColor, styleFloat, tableLayout, textAlign, textAlignLast, textAutospace,

textDecoration, textDecorationBlink, textDecorationLineThrough, textDecorationNone, textDecorationOverline, textDecorationUnderline, textIndent, textJustify, textKashidaSpace, textTransform, textUnderlinePosition, top, unicodeBidi, verticalAlign, visibility, width, wordBreak, wordSpacing, wordWrap, writingMode, zIndex, zoom.

Методы: getAttribute, getExpression, removeAttribute, removeExpression, setAttribute, setExpression.

Поддерживается IE начиная с 4.0.

style (N)

Представляет стиль элемента страницы. Доступен как элемент коллекций classes, contextual, ids и tags.

Свойства: align, backgroundColor, backgroundImage, borderBottomWidth, borderColor, borderLeftWidth, borderRightWidth, borderStyle, borderTopWidth, clear, color, display, fontFamily, fontSize, fontStyle, fontWeight, lineHeight, listStyleType, marginBottom, marginLeft, marginRight, marginTop, paddingBottom, paddingLeft, paddingRight, paddingTop, textAlign, textDecoration, whiteSpace, width.

Методы: borderWidth, margins, paddings.

Поддерживается N начиная с 4.0.

styleSheet

Представляет таблицу стилей.

Свойства: canHaveHTML, cssText, disabled, href, id, isContentEditable, idDisabled, media, owningElement, parentStyleSheet, readOnly, title, type.

Методы: addImport, addPageRule, addRule, fireEvent, hasFocus, removeRule.

Коллекции: imports, pages, rules.

Объект: page.

Поддерживается IE начиная с 4.0.

<SUB>

Отображает текст нижним индексом.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop,

offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<SUP>

Отображает текст верхним индексом.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText,

insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<TABLE>

Помещает на Web-страницу таблицу.

Свойства: accessKey, align, background, bgColor, border, borderColor, borderColorDark, borderColorLight, canHaveChildren, canHaveHTML, caption, cellPadding, cellSpacing, className, clientHeight, clientLeft, clientTop, clientWidth, cols, dataFld, dataPageSize, dataSrc, dir, firstChild, frame, height, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, rules, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, tFoot, tHead, title, uniqueID, width.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, createCaption, createTFoot, createTHead, deleteCaption, detachEvent, dragDrop, fireEvent, firstPage, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertBefore, insertRow, lastPage, mergeAttributes, moveRow, nextPage, previousPage, refresh, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut,

ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onscroll, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, cells, childNodes, children, filters, rows, tBodies.

Поддерживается IE начиная с 4.0.

<TBODY>

Используется для обозначения тела таблицы.

Свойства: accessKey, align, bgColor, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, dir, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID, vAlign.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, deleteRow, deleteTFoot, deleteTHead, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertBefore, insertRow, mergeAttributes, moveRow, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children, rows.

Поддерживается IE начиная с 4.0.

<TD>

Задает ячейку таблицы и ее содержимое.

Свойства: align, background, bgColor, borderColor, borderColorDark, borderColorLight, canHaveChildren, canHaveHTML, cellIndex, className, clientHeight, clientLeft, clientTop, clientWidth, colSpan, dir, firstChild, height, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, noWrap, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, rowSpan, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID, vAlign, width.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children, filters.

Поддерживается IE начиная с 4.0.

<TEXTAREA> (IE)

Помещает на страницу область редактирования текста.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, cols, contentEditable, dataFld, dataSrc,

defaultValue, dir, disabled, firstChild, form, hideFocus, id, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, name, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readOnly, readyState, recordNumber, rows, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, size, sourceIndex, status, STYLE, tabIndex, tagName, tagUrn, title, type, uniqueID, value, wrap.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, createTextRange, detachEvent, doScroll, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, select, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onafterupdate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onbeforeupdate, onblur, onchange, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onerrorupdate, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onscroll, onselect, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children, filters.

Поддерживается IE начиная с 3.0.

<TEXTAREA> (N)

Помещает на страницу область редактирования текста.

Свойства: defaultValue, form, name, type, value.

Методы: blur, focus, handleEvent, select.

События: onblur, onchange, onfocus, onkeydown, onkeypress, onkeyup, onselect.

Поддерживается N начиная с 2.0; свойство type добавлено в 3.0; метод handleEvent добавлен в 4.0.

TextNode

Представляет строку текста как объект в DOM.

Свойства: `data`, `length`, `nextSibling`, `nodeName`, `nodeType`, `nodeValue`, `parentNode`, `previousSibling`.

Метод: `splitText`.

Поддерживается IE начиная с 5.0.

TextRange

Представляет некоторый текстовый фрагмент Web-страницы.

Свойства: `boundingHeight`, `boundingLeft`, `boundingTop`, `boundingWidth`, `htmlText`, `offsetLeft`, `offsetTop`, `text`.

Методы: `collapse`, `compareEndPoints`, `duplicate`, `execCommand`, `expand`, `findText`, `getBookmark`, `getBoundingClientRect`, `getClientRects`, `inRange`, `isEqual`, `move`, `moveEnd`, `moveStart`, `moveToBookmark`, `moveToElementText`, `moveToPoint`, `parentElement`, `pasteHTML`, `queryCommandEnabled`, `queryCommandIndeterm`, `queryCommandState`, `queryCommandSupported`, `queryCommandValue`, `scrollIntoView`, `select`, `setEndPoint`.

Поддерживается IE начиная с 4.0.

TextRectangle

Задает прямоугольник, ограничивающий строку текста.

Свойства: `bottom`, `left`, `right`, `top`.

Поддерживается IE начиная с 5.0.

<TFOOT>

Используется для обозначения основания таблицы.

Свойства: `accessKey`, `align`, `bgColor`, `canHaveChildren`, `canHaveHTML`, `className`, `clientHeight`, `clientLeft`, `clientTop`, `clientWidth`, `dir`, `firstChild`, `hideFocus`, `id`, `innerHTML`, `innerText`, `isContentEditable`, `isDisabled`, `isTextEdit`, `lang`, `language`, `lastChild`, `nextSibling`, `nodeName`, `nodeType`, `nodeValue`, `offsetHeight`, `offsetLeft`, `offsetParent`, `offsetTop`, `offsetWidth`, `outerHTML`, `outerText`, `parentElement`, `parentNode`, `parentTextEdit`, `previousSibling`, `readyState`, `recordNumber`, `scopeName`, `scrollHeight`, `scrollLeft`, `scrollTop`, `scrollWidth`, `sourceIndex`, `STYLE`, `tabIndex`, `tagName`, `tagUrn`, `title`, `uniqueID`, `vAlign`.

Методы: `addBehavior`, `appendChild`, `applyElement`, `attachEvent`, `blur`, `clearAttributes`, `click`, `cloneNode`, `componentFromPoint`, `contains`, `deleteRow`,

detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertBefore, insertRow, mergeAttributes, moveRow, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondragenter, ondragstart, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children, rows.

Поддерживается IE начиная с 4.0.

<TH>

Задает ячейку заголовка таблицы.

Свойства: align, background, bgColor, borderColor, borderColorDark, borderColorLight, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, colSpan, dir, firstChild, height, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, noWrap, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, rowSpan, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID, vAlign, width.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy,

oncut, ondblclick, ondeactivate, ondragenter, ondragstart, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, filters.

Поддерживается IE начиная с 4.0.

<THEAD>

Используется для обозначения заголовка таблицы.

Свойства: accessKey, align, bgColor, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, dir, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID, vAlign.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, deleteRow, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertBefore, insertRow, mergeAttributes, moveRow, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondragenter, ondragstart, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children, rows.

Поддерживается IE начиная с 4.0.

<TITLE>

Определяет заголовок Web-страницы.

Свойства: canHaveHTML, firstChild, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, lastChild, nextSibling, nodeName, nodeType, nodeValue, parentElement, parentNode, parentTextEdit, previousSibling, readyState, scopeName, sourceIndex, tagName, tagUrn, text, uniqueID.

Методы: addBehavior, applyElement, attachEvent, clearAttributes, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, getAdjacentText, getAttribute, getElementsByTagName, hasChildNodes, insertAdjacentElement, mergeAttributes, removeAttribute, removeBehavior, replaceAdjacentText, setAttribute, swapNode.

События: onlayoutcomplete, onreadystatechange.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<TR>

Задает строку таблицы.

Свойства: accessKey, align, bgColor, borderColor, borderColorDark, borderColorLight, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, dir, firstChild, height, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, rowIndex, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sectionRowIndex, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID, vAlign, width.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, deleteCell, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertBefore, insertCell, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onblur, onclick, oncontextmenu,

oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfilterchange, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, cells, childNodes, children.

Поддерживается IE начиная с 4.0.

<TT>

Выводит текст моноширинным шрифтом.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<U>

Выводит текст подчеркнутым.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<UL>

Используется для создания маркированного списка.

Свойства: accessKey, blockDirection, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, compact, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText,

parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, type, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, dragDrop, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlayoutcomplete, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

userProfile

Позволяет получить доступ к информации о пользователе.

Методы: addRedRequest, clearRequest, doReadRequest, getAttribute, setAttribute.

Поддерживается IE начиная с 4.0.

<VAR>

Используется для форматирования имен переменных и функций языков программирования.

Свойства: accessKey, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit,

previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, getExpression, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertAdjacentText, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeExpression, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, setExpression, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекция: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

<WBR>

Задает "мягкий" разрыв строки.

Свойства: canHaveHTML, id, isContentEditable, isDisabled, outerHTML, outerText, parentElement, scopeName, tagUrn.

Методы: addBehavior, componentFromPoint, fireEvent, getAttribute, removeAttribute, removeBehavior, scrollIntoView, setAttribute.

Объект: currentStyle.

Коллекция: behaviorUrns.

Поддерживается IE начиная с 4.0.

window (IE)

Представляет открытое окно Web-обозревателя или фрейм.

Свойства: closed, defaultStatus, dialogArguments, dialogHeight, dialogLeft, dialogTop, dialogWidth, frameElement, length, name, offscreenBuffering, opener, parent, returnValue, screenLeft, screenTop, self, status, top.

Методы: alert, attachEvent, blur, clearInterval, clearTimeout, close, confirm, createPopup, detachEvent, execScript, focus, moveBy, moveTo, navigate, open, print, prompt, resizeBy, resizeTo, scroll, scrollBy, scrollTo, setActive, setInterval, setTimeout, showHelp, showModalDialog, showModelessDialog.

События: onactivate, onafterprint, onbeforedeactivate, onbeforeprint, onbeforeunload, onblur, oncontrolselect, ondeactivate, onerror, onfocus, onhelp, onload, onresize, onresizeend, onresizestart, onscroll, onunload.

Коллекция: frames.

Объекты: clientInformation, clipboardData, document, event, external, history, location, navigator, screen.

Поддерживается IE начиная с 3.02.

window (N)

Представляет открытое окно Web-обозревателя или фрейм.

Свойства: closed, innerHeight, innerWidth, length, locationbar, menubar, name, offscreenBuffering, opener, outerHeight, outerWidth, pageXOffset, pageYOffset, parent, personalbar, screenX, screenY, scrollbars, self, status, statusbar, toolbar, top, window.

Методы: alert, atob, back, blur, btoa, captureEvents, clearInterval, clearTimeout, close, confirm, disableExternalCapture, enableExternalCapture, find, focus, forward, handleEvent, home, moveBy, moveTo, open, print, prompt, releaseEvents, resizeBy, resizeTo, routeEvent, scroll, scrollBy, scrollTo, setHotKeys, setInterval, setResizable, setTimeout, setZOptions, stop.

События: onblur, ondragdrop, onerror, onfocus, onload, onmove, onresize, onunload.

Коллекция: frames.

Объекты: crypto, document, history, location, navigator, screen.

Поддерживается N начиная с 2.0.

<XML>

Помещает фрагмент XML-кода в HTML-код.

Свойства: canHaveHTML, id, isContentEditable, isDisabled, parentElement, readyState, recordset, scopeName, scrollHeight, src, tagUrn, XMLDocument.

Методы: addBehavior, componentFromPoint, fireEvent, removeBehavior.

События: ondataavailable, ondatasetchanged, ondatasetcomplete, onreadystatechange, onrowenter, onrowexit, onrowsdelete, onrowsinserted.

Коллекция: behaviorUrns.

Поддерживается IE начиная с 5.0.

<XMP>

Выводит блок текста моноширинным шрифтом с сохранением всего форматирования и без учета HTML-тегов.

Свойства: accessKey, blockDirection, canHaveChildren, canHaveHTML, className, clientHeight, clientLeft, clientTop, clientWidth, contentEditable, dir, disabled, firstChild, hideFocus, id, innerHTML, innerText, isContentEditable, isDisabled, isTextEdit, lang, language, lastChild, nextSibling, nodeName, nodeType, nodeValue, offsetHeight, offsetLeft, offsetParent, offsetTop, offsetWidth, outerHTML, outerText, parentElement, parentNode, parentTextEdit, previousSibling, readyState, recordNumber, scopeName, scrollHeight, scrollLeft, scrollTop, scrollWidth, sourceIndex, STYLE, tabIndex, tagName, tagUrn, title, uniqueID.

Методы: addBehavior, appendChild, applyElement, attachEvent, blur, clearAttributes, click, cloneNode, componentFromPoint, contains, detachEvent, fireEvent, focus, getAdjacentText, getAttribute, getBoundingClientRect, getClientRects, getElementsByTagName, hasChildNodes, insertAdjacentElement, insertAdjacentHTML, insertBefore, mergeAttributes, releaseCapture, removeAttribute, removeBehavior, removeChild, removeNode, replaceAdjacentText, replaceChild, replaceNode, scrollIntoView, setActive, setAttribute, setCapture, swapNode.

События: onactivate, onbeforecut, onbeforedeactivate, onbeforepaste, onblur, onclick, oncontextmenu, oncontrolselect, oncut, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onfocus, onhelp, onkeydown, onkeypress, onkeyup, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onpaste, onpropertychange, onreadystatechange, onresize, onresizeend, onresizestart, onselectstart.

Объекты: currentStyle, runtimeStyle, style.

Коллекции: all, attributes, behaviorUrns, childNodes, children.

Поддерживается IE начиная с 4.0.

Коллекции

Коллекции — это объекты, являющиеся набором объектов. Ниже приводится алфавитный перечень коллекций DOM с пояснением их семантики, описанием формального синтаксиса, перечнем свойств, методов и сведениями о поддержке наиболее распространенными Web-обозревателями.

all

Содержит все дочерние элементы.

```
{Объект}.all({Первичный индекс}[, {Вторичный индекс}]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать элементы, у которых значения атрибутов ID или NAME совпадают с этой строкой, и будет возвращена коллекция найденных элементов. Для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс.

Свойство: length.

Методы: item, tags, urns.

Поддерживается IE начиная с 4.0.

anchors

Содержит все "якоря".

```
document.anchors({Первичный индекс}[, {Вторичный индекс}]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать "якорь", у которого значения атрибутов ID или NAME совпадают с этой строкой. Если таких "якорей" несколько, то IE вернет коллекцию найденных "якорей"; для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс. N в этом случае вернет первый найденный "якорь".

Свойство: length.

Методы (N не поддерживаются): item, tags, urns.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

applets

Содержит все внедренные элементы <APPLET>.

```
document.applets({Первичный индекс}[, {Вторичный индекс}]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать элемент, у которого значения атрибутов ID или NAME совпадают с этой строкой. Если таких элементов несколько, то IE вернет коллекцию найденных элементов; для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс. N в данном случае вернет первый найденный элемент.

Свойство: length.

Методы (N не поддерживаются): item, tags, urns.

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

areas

Содержит все "горячие" области <AREA> в списке областей <MAP>.

```
{Объект}.areas([Первичный индекс][, {Вторичный индекс}]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать области, у которых значения атрибутов ID или NAME совпадают с этой строкой, и будет возвращена коллекция найденных областей. Для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс.

Свойство: length.

Методы: add, item, remove, tags, urns.

Поддерживается IE начиная с 4.0.

attributes

Содержит все атрибуты тега элемента страницы.

```
{Объект}.attributes({Индекс});
```

Свойство: length.

Метод: item.

Поддерживается IE начиная с 5.0.

behaviorUrns

Содержит все поведения, привязанные к текущему элементу страницы.

```
{Объект}.behaviorUrns({Индекс});
```

Свойство: length.

Метод: item.

Поддерживается IE начиная с 5.0.

bookmarks

Содержит все "закладки", сделанные в базе данных.

```
event.bookmarks({Индекс});
```

Свойство: length.

Метод: item.

Поддерживается IE начиная с 4.0.

boundElements

Содержит все элементы страницы, привязанные к данным.

```
document.boundElements([Первичный индекс][, [Вторичный индекс]]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать элементы, у которых значения атрибутов ID или NAME совпадают с этой строкой, и будет возвращена коллекция найденных элементов. Для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс.

Свойство: length.

Методы: item, tags, urns.

Поддерживается IE начиная с 5.0.

cells

Содержит все ячейки строки таблицы или всей таблицы.

```
[Объект].cells([Первичный индекс][, [Вторичный индекс]]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать ячейки, у которых значения атрибутов ID или NAME совпадают с этой строкой, и будет возвращена коллекция найденных ячеек. Для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс.

Свойство: length.

Методы: item, tags, urns.

Поддерживается IE начиная с 4.0 для <TR> и начиная с 5.0 для <TABLE>.

childNodes

Содержит все элементы страницы и объекты TextNode, являющиеся дочерними для текущего элемента страницы.

```
[Объект].childNodes([Первичный индекс][, [Вторичный индекс]]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать элементы, у которых значения атрибутов ID или NAME совпадают с этой строкой, и будет возвращена коллекция найденных элементов. Для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс.

Свойство: length.

Методы: item, urns.

Поддерживается IE начиная с 5.0.

children

Содержит все элементы страницы, являющиеся дочерними для текущего элемента.

```
{Объект}.children({Первичный индекс}[, {Вторичный индекс}]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать элементы, у которых значения атрибутов ID или NAME совпадают с этой строкой, и будет возвращена коллекция найденных элементов. Для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс.

Свойство: length.

Методы: item, tags, urns.

Поддерживается IE начиная с 4.0.

classes

Позволяет задать стиль всех элементов страницы, имеющих заданный идентификатор класса.

```
document.classes.{Имя идентификатора класса}.all|{Имя тега}.{Свойство
↳стиля}[ = "{Значение свойства}"];
```

Пример использования коллекции classes:

```
document.classes.samplestyle.all.color = "green";
```

Задает зеленый цвет текста для всех элементов страницы с идентификатором класса samplestyle.

```
document.classes.redheader.H1.color = "red";
```

Задает красный цвет текста для всех элементов страницы <H1> с идентификатором класса redheader.

Поддерживается N начиная с 4.0.

contextual

Позволяет задать стиль всех элементов страницы, появляющихся в нужном контексте.

```
document.contextual({Список }).{Свойство стиля})[ = "{Значение
↳свойства}"];
```

Пример использования коллекции contextual:

```
document.contextual(document.tags.H1, document.tags.B).color = "blue";
```

Задает синий цвет текста для всех элементов страницы <B>, находящихся внутри <H1>.

Поддерживается N начиная с 4.0.

controlRange

Возвращается методами createControlRange и createRange.

Свойство: length.

Методы: add, execCommand, item, queryCommandEnabled, queryCommandIndeterm, queryCommandState, queryCommandSupported, queryCommandValue, remove, scrollIntoView, select.

Поддерживается IE начиная с 5.0.

elements

Содержит все элементы управления, находящиеся в форме.

```
{Объект формы}.elements({Первичный индекс}[, {Вторичный индекс}]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать элемент, у которого значения атрибутов ID или NAME совпадают с этой строкой. Если таких элементов несколько, то IE вернет коллекцию найденных элементов; для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс. N в данном случае вернет первый найденный элемент.

Свойство: length.

Методы (N не поддерживаются): item, tags, urns.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

embeds

Содержит все внедренные элементы <EMBED>.

```
document.embeds({Первичный индекс}[, {Вторичный индекс}]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать элемент, у которого значения атрибутов ID или NAME совпадают с этой строкой. Если таких элементов несколько, то IE вернет коллекцию найденных элементов; для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс. N в этом случае вернет первый найденный элемент.

Свойство: length.

Методы (N не поддерживаются): item, tags, urns.

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

filters

Содержит все визуальные фильтры и преобразования, примененные к текущему элементу страницы.

```
{Объект}.filters({Первичный индекс}[, {Вторичный индекс}]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать элементы, у которых значения атрибутов ID или NAME совпадают с этой строкой, и будет возвращена коллекция найденных элементов. Для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс.

Свойство: `length`.

Метод: `item`.

Поддерживается IE начиная с 4.0.

forms

Содержит все формы.

```
document.forms({Первичный индекс}[, {Вторичный индекс}]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать форму, у которой значения атрибутов ID или NAME совпадают с этой строкой. Если таких форм несколько, то IE вернет коллекцию найденных форм; для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс. N в этом случае вернет первую найденную форму.

Свойство: `length`.

Методы (N не поддерживаются): `item`, `tags`, `urns`.

Поддерживается IE начиная с 3.02 и N начиная с 3.0.

frames

Содержит все объекты `window`, соответствующие фреймам в наборе.

```
document.frames({Первичный индекс}[, {Вторичный индекс}]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать фрейм, у которого значения атрибутов ID или NAME совпадают с этой строкой. Если таких фреймов несколько, то IE вернет коллекцию найденных форм; для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс. N в этом случае вернет первый найденный фрейм.

Свойство: `length`.

Метод (N не поддерживается): item.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

ids

Позволяет задать стиль всех элементов страницы, имеющих заданное значение атрибута ID.

```
document.ids.{Значение атрибута ID}.all|{Имя тега}.{Свойство
↳стиля}[ = "{Значение свойства}"];
```

Примеры использования коллекции ids.

```
document.ids.samplestyle.all.color = "green";
```

Задает зеленый цвет текста для всех элементов страницы, значение ID которых установлено в samplestyle.

```
document.classes.redheader.H1.color = "red";
```

Задает красный цвет текста для всех элементов страницы <H1>, значение ID которых установлено в redheader.

Поддерживается N начиная с 4.0.

images

Содержит все графические изображения.

```
document.images({Первичный индекс}[, {Вторичный индекс}]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать изображение, у которого значения атрибутов ID или NAME совпадают с этой строкой. Если таких изображений несколько, то IE вернет коллекцию найденных изображений; для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс. N в этом случае вернет первое найденное изображение.

Свойство: length.

Методы (N не поддерживаются): item, tags, urns.

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

imports

Содержит все импортированные стили в текущей таблице стилей.

```
{Объект таблицы стилей}.imports({Индекс});
```

Свойство: length.

Метод: item.

Поддерживается IE начиная с 4.0.

layers

Содержит все слои.

```
document.layers([Индекс]);
```

Индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать слой, у которого значения атрибута NAME совпадает с этой строкой. Если таких слоев несколько, N вернет первый найденный.

Свойство: `length`.

Поддерживается N начиная с 4.0.

links

Содержит все гиперссылки. В N также включает все "горячие" области <AREA>.

```
document.links([Индекс]);
```

Свойство: `length`.

Методы (N не поддерживаются): `item`, `tags`, `urns`.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

mimeTypes

Содержит все типы данных MIME, поддерживаемые Web-обозревателем.

```
navigator.mimeTypes([Индекс]);
```

Индекс может быть как числовым, так и строковым. В последнем случае он представляет собой тип данных MIME.

Свойство: `length`.

Поддерживается N начиная с 3.0.

namespaces

Содержит все пространства имен.

```
document.namespaces([Первичный индекс][, [Вторичный индекс]]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать элементы, у которых значения атрибутов ID или NAME совпадают с этой строкой, и будет возвращена коллекция найденных элементов. Для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс.

Свойство: `length`.

Методы: add, item.

Поддерживается IE начиная с 5.5.

options

Содержит все пункты <OPTION> списка <SELECT>.

```
{Объект списка}.options({Первичный индекс}[, {Вторичный индекс}]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать пункт, у которого значения атрибутов ID или NAME совпадают с этой строкой. Если таких пунктов несколько, то IE вернет коллекцию найденных пунктов; для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс. N в этом случае вернет первый найденный пункт.

Свойство: length.

Методы (N не поддерживаются): add, item, remove, tags, urns.

Поддерживается IE начиная с 3.02 и N начиная с 2.0.

pages

Содержит все правила @page в текущей таблице стилей.

```
{Объект таблицы стилей}.pages({Индекс});
```

Свойство: length.

Метод: item.

Поддерживается IE начиная с 5.5.

plugins (document)

Содержит все внедренные элементы <EMBED>.

```
document.plugins({Индекс});
```

Свойство: length.

Методы (N не поддерживаются): item, tags.

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

plugins (navigator)

Содержит все расширения, установленные в программе Web-обозревателя.

```
navigator.plugins({Индекс});
```

Свойство: length.

Метод: `refresh`.

Поддерживается IE начиная с 4.0 и N начиная с 3.0.

rows

Содержит все строки таблицы или ее раздела.

```
{Объект}.rows({Первичный индекс}[, {Вторичный индекс}]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать строки, у которых значения атрибутов ID или NAME совпадают с этой строкой, и будет возвращена коллекция найденных строк. Для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс.

Свойство: `length`.

Методы: `item`, `tags`, `urns`.

Поддерживается IE начиная с 4.0.

rules

Содержит все описания стилей в текущей таблице стилей.

```
{Объект таблицы стилей}.rules({Индекс});
```

Свойство: `length`.

Метод: `item`.

Поддерживается IE начиная с 4.0.

scripts

Содержит все скрипты <SCRIPT>.

```
document.scripts({Первичный индекс}[, {Вторичный индекс}]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать скрипты, у которых значения атрибутов ID или NAME совпадают с этой строкой, и будет возвращена коллекция найденных скриптов. Для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс.

Свойство: `length`.

Методы: `item`, `tags`, `urns`.

Поддерживается IE начиная с 4.0.

styleSheets

Содержит все таблицы стилей.

```
document.styleSheets([Первичный индекс][, [Вторичный индекс]]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать таблицы стилей, у которых значения атрибутов ID или NAME совпадают с этой строкой, и будет возвращена коллекция найденных таблиц стилей. Для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс.

Свойство: length.

Методы: item, urns.

Поддерживается IE начиная с 4.0.

tags

Позволяет задать стиль всех элементов страницы, созданных с помощью тега с указанным именем.

```
document.tags.[Имя тега].[Свойство стиля][ = "[Значение свойства]"];
```

Пример использования коллекции tags:

```
document.ids.H1.color = "blue";
```

Задает синий цвет текста для всех элементов страницы <H1>.

Поддерживается N начиная с 4.0.

tBodies

Содержит все разделы <TBODY> в текущей таблице.

```
[Объект таблицы].tBodies([Первичный индекс][, [Вторичный индекс]]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать разделы, у которых значения атрибутов ID или NAME совпадают с этой строкой, и будет возвращена коллекция найденных секций. Для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс.

Свойство: length.

Методы: item, tags, urns.

Поддерживается IE начиная с 4.0.

TextRange

Коллекция объектов TextRange.

```
selection.TextRange([Первичный индекс][, [Вторичный индекс]]);
```

Первичный индекс может быть либо числом, либо строкой; во втором случае он позволяет выбрать объекты, у которых значения атрибутов ID или NAME совпадают с этой строкой, и будет возвращена коллекция найденных объектов. Для выбора нужного элемента из этой результирующей коллекции можно использовать числовой вторичный индекс.

Свойство: `length`.

Метод: `item`.

Поддерживается IE начиная с 5.5.

TextRectangle

Возвращается методом `getClientRects`.

Свойство: `length`.

Метод: `item`.

Поддерживается IE начиная с 5.0.

ПРИЛОЖЕНИЕ 4

Примеры JavaScript-сценариев

В этом приложении приведены несколько несложных JavaScript-сценариев, выложенных для свободного использования на сайте http://www.sources.ru. Каждый скрипт приводится с минимальными изменениями и минимальным описанием; начинающим JavaScript-программистам предлагается разобраться в их работе самостоятельно.

Первые несколько скриптов будут представлять собой обычные украшательства.

Движущийся фон страницы

Этот скрипт создает очень красивый эффект движения фона Web-страницы. Работает только в Internet Explorer.

```
<BODY BACKGROUND="somepic.gif">
<SCRIPT LANGUAGE="JavaScript">
var backgroundOffset = 0;
var bgObject = eval('document.body');
function scrollBG(maxSize) {
  backgroundOffset = backgroundOffset + 1;
  if (backgroundOffset > maxSize) backgroundOffset = 0;
  bgObject.style.backgroundPosition = "0 " + backgroundOffset; }
var ScrollTimer = window.setInterval("scrollBG(307)", 64);
</SCRIPT>
```

Автор скрипта Roy Sinclair (roysinclair@email.msn.com).

"Выползающий" заголовок

Этот скрипт позволит заголовку окна Web-обозревателя постепенно "выползать" слева. Это выглядит забавно.

```
<HEAD>
. . .
<SCRIPT>
var tit = document.title;
var c = 0;

function writetitle() {
  document.title = tit.substring(0, c);
  if(c == tit.length) {
    c = 0;
    setTimeout("writetitle()", 3000) }
  else {
    c++;
    setTimeout("writetitle()", 200) } }

writetitle();
</SCRIPT>
. . .
</HEAD>
```

Автор скрипта неизвестен.

"Дрожащая" страница

Этот скрипт позволит вам позабавить посетителя сайта. При наведении курсора мыши на картинку или какой-либо иной элемент страницы окно Web-обозревателя "вздрагивает".

```
<HEAD>
. . .
<SCRIPT language=JavaScript1.2>
function boom(n) {
  if (window.top.moveBy) {
    for (i = 10; i > 0; i--) {
      for (j = n; j > 0; j--) {
        window.top.moveBy(0,i);
        window.top.moveBy(i,0);
        window.top.moveBy(0,-i);
        window.top.moveBy(-i,0); } } } }
</SCRIPT>
. . .
</HEAD>
```

Эта часть кода помещается в HTML-заголовке страницы и представляет собой определение функции, обеспечивающей "дрожание" окна.

```
<BODY>
. . .
<A NAME="boom" onmouseover="boom(1);"><IMG SRC="thrill.gif"></A>
. . .
</BODY>
```

А это был приведен пример ее вызова.

Автор скрипта неизвестен.

Теперь — более сложный скрипт.

Переливающиеся гиперссылки

Этот скрипт плавно изменяет цвет гиперссылок во всей Web-странице. Работает только в Internet Explorer.

```
<HEAD>
. . .
<SCRIPT LANGUAGE="JavaScript">
function initArray() {
  for (var i = 0; i < initArray.arguments.length; i++) {
    this[i] = initArray.arguments[i]; }
  this.length = initArray.arguments.length; }

function linkDance() {
  link = (link + 1) % colors.length;
  vlink = (vlink + 1) % colors.length;
  document.linkColor = colors[link];
  document.vlinkColor = colors[vlink];
  setTimeout("linkDance()", delay * 1000); }

var colors = new initArray(
"red",
"blue",
"green",
"purple",
"black",
"tan",
"red");
delay = .5; // seconds
link = 0;
vlink = 2;
linkDance();
</SCRIPT>
. . .
<HEAD>
```

Автор скрипта неизвестен.

Следующий наш скрипт принесет большую практическую пользу. Поскольку он очень невелик, вы можете просто запомнить его и в дальнейшем вставлять в свои страницы.

Полноэкранное окно

Этот небольшой скрипт открывает Web-страницу в новом окне, причем оно занимает весь экран компьютера и не содержит ни строки меню, ни строки состояния, ни панелей инструментов. Такое окно может быть использовано для показа видеоклипов, больших картин или даже в качестве начальной страницы сайта "концептуального" дизайна.

```
<HEAD>
. . .
<SCRIPT LANGUAGE="JavaScript">
function fullScreen(theURL) {
  window.open(theURL, '', 'fullscreen=yes, scrollbars=auto'); }
</SCRIPT>
. . .
</HEAD>
```

Эта часть кода помещается в HTML-заголовке страницы и представляет собой определение функции, выводящей заданную Web-страницу в новом окне.

```
<BODY>
. . .
<A HREF="javascript:void(0);" onclick="fullScreen('somepage.html');">
Открываем окно на весь экран</A>
. . .
</BODY>
```

А это был небольшой пример вызова функции `fullScreen`.

Автор скрипта Paul Deron (trombonepaul@yahoo.com).

И в заключение рассмотрим даже не скрипт, а довольно сложный пример Web-страницы.

Иерархический список разделов

Иерархический список разделов аналогичен стандартному иерархическому списку Windows, который вы можете видеть, например, в левой панели Проводника. Он состоит из разделов и вложенных в них подразделов, организованных в виде "дерева"; пользователь может сворачивать и разворачивать его "ветви", чтобы увидеть всю структуру сайта. Каждый пункт такого

списка представляет собой гиперссылку, щелкнув по которой пользователь перейдет на нужную страницу.

Ниже приведен небольшой пример организации такого иерархического списка разделов. Работает он опять же только в Internet Explorer.

```
<HTML>
<HEAD>
<TITLE>Иерархический список разделов</TITLE>
<STYLE TYPE="text/css">
  .sub{ visibility: hidden;
        display: none; }
</STYLE>
<SCRIPT LANGUAGE=JAVASCRIPT>
function subDisplay(subName) {
  if (eval(subName).style.visibility == "hidden"){
    eval(subName).style.visibility = "visible";
    eval(subName).style.display = "list-item";
    eval("img" + subName).src = "opened.gif" }
  else {
    eval(subName).style.visibility = "hidden";
    eval(subName).style.display = "none";
    eval("img" + subName).src = "closed.gif" } }
</SCRIPT>
</HEAD>
<BODY>
<UL>
  <IMG SRC="closed.gif" ID="imgsub1">
  <SPAN onclick="subDisplay('sub1')">Избранное</SPAN>
  <UL CLASS="sub" ID="sub1">
    <IMG SRC="closed.gif" ID="imgsub2">
    <SPAN onclick="subDisplay('sub2')">Музыка</SPAN><BR>
    <UL CLASS="sub" ID="sub2">
      <IMG SRC="topic.gif">
      <A HREF="http://www.napster.com">Napster</A><BR>
      <IMG SRC="topic.gif">
      <A HREF="http://www.mp3.com">MP3.com</A><BR>
      <IMG SRC="topic.gif">
      <A HREF="http://www.scour.net">Scour</A><BR>
    </UL>
    <IMG SRC="closed.gif" ID="imgsub3">
    <SPAN onclick="subDisplay('sub3')">Исходные тексты</SPAN><BR>
    <UL CLASS="sub" ID="sub3">
      <IMG SRC="topic.gif">
      <A HREF="http://www.sources.ru">ИСХОДНИКИ.RU</A><BR>
    </UL>
```

```
    <IMG SRC="topic.gif">
    <A HREF="http://www.gothic.ru">Русские готические страницы</A>
  </UL>
</UL>
</BODY>
</HTML>
```

Автор скрипта Tim Fischer (tim@bbs-la.com).

Предметный указатель

А

Б

В

Г

Д

З

И

К

П

С

Т

У

Ф