Самая популярная серия компьютерных книг в США!

PRENTICE HALL PTR

Строим Web-сайты

GARAGE

ДИЗАЙН • HTML • CSS

МАРК КЭМПБЕЛ

Издательство
ТРИУМФ

Серия GARAGE

Марк Кэмпбел

Строим Web-сайты

ДИЗАЙН • HTML • CSS

Перевод с английского языка

PRENTICE HALL PTR

«Издательство Триумф»
Москва

WEB DESIGN GARAGE

Marc Campbell

Prentice Hall Professional Technical reference
Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

УДК 004.738.5
ББК 32.973.202
К98

Кэмпбел, Марк.

К98 Строим Web-сайты. Дизайн • HTML • CSS. GARAGE : пер. с англ. яз. / Марк Кэмпбел ; [пер. с англ. Александр Горлач, Александр Климович]. — М. : Изд-во ТРИУМФ, [2006]. — 480 с. : ил. — (Серия «Garage»). — Доп. тит. л. англ. — **ISBN 5-89392-134-8**.

Агентство CIP РГБ

Если вы – начинающий Web-мастер, занятый построением собственного Web-сайта, неважно, с какой целью – для рекламы своего бизнеса, для онлайновой торговли или просто так, для самовыражения, – то эта книга для вас. С ее помощью вы станете настоящим профессионалом в области Web-дизайна.

Вашим гидом будет признанный разработчик Web-сайтов и преподаватель Марк Кэмпбел. Его книга отличается редким достоинством: она не требует от вас нудного чтения «от корки до корки». Скорее, книга приспособлена для быстрого поиска нужной темы, получения ответа на вопрос «как сделать это?» и практического применения полученных знаний. И можно смело утверждать: нет таких вопросов Web-дизайна, на которые вы не найдете ответа в этой книге!

Данное издание особенно интересно тем, что оно выпускается в серии **GARAGE**, которая пользуется большой популярностью у американских читателей. Ведь, как известно, многие передовые технологии современной цивилизации, в том числе и компьютер, создавались именно в гараже.

Посетите наш Интернет-магазин «Три ступеньки®»: www.3st.ru
E-mail: post@triumph.ru

Authorized translation from the English language edition, entitled WEB DESIGN GARAGE, 1st Edition, ISBN 0131481991, by CAMPBELL, MARC, published by Pearson Education, Inc, publishing as Prentice Hall PTR, Copyright © 2005 Pearson Education Inc.

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means – electronic or mechanical, including photocopying, recording or by any information storage retrieval systems – without the written permission from Pearson Education, Inc. RUSSIAN language edition published by Triumph Publishing, Copyright © 2006.

Авторизованный перевод англоязычного издания, под названием WEB DESIGN GARAGE, 1st Edition, ISBN 0131481991, by CAMPBELL, MARC, изданного Pearson Education, Inc, publishing as Prentice Hall PTR, Copyright © 2005 Pearson Education Inc.

Все права защищены. Никакая часть данной книги не может быть переделана или изменена в какой-либо форме, электронной или механической, включая фотокопирование, переписи на носители информации без разрешения Pearson Education, Inc. Русскоязычная версия, изданная ООО «ИЗДАТЕЛЬСТВО ТРИУМФ», Copyright © 2006 г.

ISBN 5-89392-134-8
ISBN 0-13-148199-1 (США)

© Обложка, серия, оформление
ООО «Издательство ТРИУМФ», 2006

Оглавление

Введение 11

ЧАСТЬ 1. Дизайн и удобство использования 13

ГЛАВА 1. Разработка дизайна, облегчающего навигацию 14

ГЛАВА 2. Группирование схожих функций 20

ГЛАВА 3. Разработка рисунков для щелчков 28

ГЛАВА 4. Достижение сбалансированности 36

ГЛАВА 5. Выбор цветов 43

Выбор цветов текста и фона 43

Выбор цветов ссылок 47

Выбор цветов интерфейса 48

ГЛАВА 6. Разработка специальных возможностей 51

Обеспечение доступности изображений 52

Обеспечение доступности мультимедиа 54

Управление использованием цвета 55

ГЛАВА 7. Удобство навигации 57

ГЛАВА 8. Использование меню переходов 62

ГЛАВА 9. Управление всплывающими окнами 68

Создание всплывающих окон 70

Перемещение всплывающего окна на передний план 73

Добавление ссылки Close 74

ГЛАВА 10. Создание открывающихся меню 75

Понятие слоев 76

Размещение открывающихся меню 78

ГЛАВА 11. Обеспечение путей отхода 89

ГЛАВА 12. Работа со скриптами и таблицами стилей 92

Внедрение скриптов 92

Внедрение таблиц стилей 94

Присоединение скриптов 95

Присоединение таблиц стилей 96

Дневник: Внедрение или присоединение? 97

ГЛАВА 13. Запуск скриптов при помощи ссылок 98

Дневник: JavaScript-ссылки или onClick-ссылки? 99

ЧАСТЬ 2. Вопросы, касающиеся макетов 100

ГЛАВА 14. Создание макетов с фиксированной шириной при помощи таблиц 101

Дневник: для создания макета использовать HTML- или CSS-код? 107

ГЛАВА 15. Создание макетов с фиксированной шириной при помощи таблиц CSS 108

ГЛАВА 16. Центрирование CSS-макетов с фиксированной шириной 113

ГЛАВА 17. Создание текучих макетов при помощи таблиц 116

ГЛАВА 18. Создание текучих макетов при помощи таблиц CSS 122

ГЛАВА 19. Разделение изображений на срезы для макетных таблиц 127

ГЛАВА 20. Создание многоколонных макетов при помощи таблиц 132

ГЛАВА 21. Создание многоколонных макетов при помощи таблиц CSS 136

ГЛАВА 22. Предотвращение разрыва навигационной панели 139

ГЛАВА 23. Раскрашивание ячеек таблицы 146

ГЛАВА 24. Раскрашивание и форматирование CSS-разделов 151

ГЛАВА 25. Управление пространством дизайна в языке HTML 158

ГЛАВА 26. Управление пространством дизайна при помощи таблиц CSS 159

ЧАСТЬ 3. Вопросы, касающиеся изображений 161

ГЛАВА 27. Выбор правильного типа файла изображения 162

ГЛАВА 28. Оптимизация Web-изображений 164

Оптимизация разрешения 165

Оптимизация размера изображения 167

Оптимизация палитр файлов в форматах GIF и PNG 167

Оптимизации сжатия изображений в формате JPEG 168

ГЛАВА 29. Трансформация изображений с использованием атрибутов языка HTML 169

ГЛАВА 30. Размещение внутристрочных изображений при помощи языка HTML 171

ГЛАВА 31. Плавающие внутристрочные изображения с использованием таблиц CSS 173

ГЛАВА 32. Разработка состояний кнопок 177

ГЛАВА 33. Создание ролловерной графики 180

ГЛАВА 34. Повышение доступности изображений 184

ГЛАВА 35. Создание карт ссылок 186

ГЛАВА 36. Повышение доступности карт ссылок 190

ГЛАВА 37. Использование фоновых изображений в ячейках таблицы 193
ГЛАВА 38. Использование фоновых изображений в CSS-разделах 196
ГЛАВА 39. Использование фоновых изображений на Web-страницах 199

ЧАСТЬ 4. Текст 203

ГЛАВА 40. Правильное использование текстовых элементов 204
Работа с тегами заголовков 205
Маркировка абзацев 207
Использование элементов цитат 207
Отображение адреса 208
ГЛАВА 41. Правильное использование элементов фраз 210
ГЛАВА 42. Добавление каскадных таблиц стилей 213
Написание каскадных таблиц стилей 214
Написание таблицы стилей 214
Внедрение таблиц стилей 220
Импорт таблиц стилей 221
Использование HTML-атрибута стиля 223
ГЛАВА 43. Замена тегов форматирования каскадными таблицами стилей 225
ГЛАВА 44. Создание специальных стилей форматирования 228
ГЛАВА 45. Изменение внешнего вида текстового элемента 232
Определение контекстно-зависимых селекторов 232
Определение дочерних селекторов 235
Определение братских селекторов 237
Смешивание селекторов разных типов 238
ГЛАВА 46. Определение стилей классов 240
Формирование привилегированного клуба 243
ГЛАВА 47. Управление шрифтом 245
Устранение ошибок, связанных со шрифтами 246
Выбор наилучшей гарнитуры 248
ГЛАВА 48. Управление размером шрифта 249
Использование тега <font> 249
Использование атрибута font-size 250
Использование констант длины в каскадных таблицах стилей 252
Дневник: способ изменения размера шрифта 253
ГЛАВА 49. Управление интервалами 254
Управление пространством вокруг элементов 254
Управление пространством внутри элементов 256

Управление междустрочными интервалами 256
Управление зазорами между словами 257
Управление зазором между буквами 258
ГЛАВА 50. Управление выравниванием текста 260
ГЛАВА 51. Управление отступами 263
ГЛАВА 52. Создание списков 269
Вложенные списки в HTML 271
Настройка свойств списка с помощью каскадных таблиц стилей 273
Замена маркеров изображением 273
Изменение положения начального символа 274
Переопределение типа списка, используемого браузером по умолчанию 277
ГЛАВА 53. Выделение текста с помощью цвета 280
ГЛАВА 54. Реабилитация горизонтальных линий 283
Создание знаков конца 286
ГЛАВА 55. Добавление строки даты и метки времени 287
ГЛАВА 56. Разработка таблиц данных 295
Использование большинства тегов группировки 299
Работа с атрибутами colspan и rowspan 300
Выбор значений атрибутов cellpadding и cellspacing 305
Изменение границы 308
Повышение доступности 312
ГЛАВА 57. Использование псевдоэлементов 318
ГЛАВА 58. Определение стилей идентификаторов 323
ЧАСТЬ 5. Ссылки 325
ГЛАВА 59. Применение стилей к гиперссылкам 326
Дневник: в защиту подчеркивания ссылок 331
ГЛАВА 60. Создание ролловерных гиперссылок 332
ГЛАВА 61. Выбор надлежащего пути 334
Использование абсолютных путей 334
Использование относительных путей, указываемых относительно документа 336
Использование относительных путей, указываемых относительно корня 338
ГЛАВА 62. Изменение цвета одиночной гиперссылки 340
Изменение цвета, используемого по умолчанию для всех состояний ссылок 340
Изменение стандартного цвета для отдельных состояний ссылок 341
ГЛАВА 63. Выбор подходящего текста для гиперссылки 343
ГЛАВА 64. Создание всплывающих подсказок для гиперссылок 349
ГЛАВА 65. Открытие нового окна браузера 351

ГЛАВА 66. Использование именованных анкеров 352
Задание именованных анкеров 352
Привязка к именованным анкерам 353
ГЛАВА 67. Создание ссылок электронной почты 355

ЧАСТЬ 6. Формы 357

ГЛАВА 68. Работа с элементами управления форм 358
Работа с кнопками общего назначения 359
Работа с флажками 360
Работа с полями файлов 361
Работа со скрытыми полями 363
Работа с полями изображений 364
Замена кнопки Reset (Сброс) *365*
Замена кнопки общего назначения *366*
Работа со списками 366
Работа с меню 368
Работа с полями паролей 370
Работа с переключателями 371
Работа с кнопками сброса 373
Работа с кнопками Submit (Передать) 374
Работа с текстовыми областями 375
Работа с текстовыми полями 377
ГЛАВА 69. Применение стилей к текстовым элементам управления 379
Изменение текстового стиля 379
Изменение цвета и границы 382
ГЛАВА 70. Проверка достоверности входных данных формы 385
ГЛАВА 71. Навигация с помощью клавиши Tab 393
ГЛАВА 72. Работа с наборами полей 397
ГЛАВА 73. Работа с метками 405

ЧАСТЬ 7. Специальные приемы 409

ГЛАВА 74. Встраивание мультимедиа 410
ГЛАВА 75. Автоматическое обновление страницы 415
ГЛАВА 76. Автоматическая переадресация браузера 419
ГЛАВА 77. Отображение содержимого, выбираемого случайным образом 422
ГЛАВА 78. Вывод предупреждения браузера 433
ГЛАВА 79. Отображение всплывающего окна «Вы покидаете сайт» 439

ЧАСТЬ 8. Базовое обучение 441

ГЛАВА 80. Создание HTML-документов 442

Организация Web-сайта 447

ГЛАВА 81. Добавление ключевых слов и описаний страниц 452

ГЛАВА 82. Блокирование частей сайта для поисковых систем 454

ГЛАВА 83. Тестирование сайта 458

ГЛАВА 84. Выбор Web-хоста 461

Поиск Web-хоста 461

Предоставляемые возможности 464

ГЛАВА 85. Загрузка сайта на сервер 466

ГЛАВА 86. Проверка правильности исходного кода 468

Глоссарий 470

Введение

Это случается всегда? Чтобы попытаться объяснить что-то, вам требуется пять сотен дополнительных рукописных страниц, около девяноста тысяч слов, около двухсот тридцати рисунков и шесть месяцев жизни, а затем один лихой человек неожиданно приходит и обобщает все это одним предложением.

Это случилось со мной, когда я отправил эту скромную книгу на одобрение Эбену Хьюиту, редактору книг серии Garage Series. Он ответил так – цитирую: «Кэмпбел выполнил прекрасную работу для демонстрации того, что важен не только дизайн, но и удобство использования – они даже могли бы быть единым». Когда я прочитал это, то чуть не упал со стула. «Правильно!» – закричал я. – «Вот, что я пытался сказать!»

Но затем я подумал – кто еще мог бы прочитать это? Кто в моем издательстве мог бы сопоставить резюме из двадцати слов стопке рукописных бумаг и думать о времени, усилиях и сбережениях, сохраненных в результате выпуска брошюры или информационного листа, или даже искусного лозунга вместо полной книги? Как мог невероятно проницающий комментарий Хьюита повлиять на условия моего соглашения? Мое повышение? Это были сугубо деловые отношения, поскольку я уже потратил все.

К счастью для меня, гром не грянул. Эта книга спокойно отправилась в печать, и кажется никто не заметил, что сделал Эбен Хьюит – Java-программист, не меньше. А теперь и подавно.

Или, может быть, привлекательная экономия заключалась в том, что пока Хьюит может извергать все кратко изложенные, важные тезисы, которые ему нравятся, книга перед вами, не достигнув уровня четкости мышления Эбена Хьюита, предоставляет подробности в виде утешения. В книге в виде множества примеров демонстрируется, как дизайн и удобство использования неумолимо переплетаются. В ней даже делается попытка объяснить, почему так происходит.

Поэтому, если вы генератор идей, высоко концептуальный проповедник с интеллектом шестого уровня, Эбен Хьюит, с вашего позволения не читайте дальше. Редактор серии просто дал все, что должна предоставить данная книга. Но если вы похожи на меня, больше похожи на пожарный гидрант, чем на проповедника, с интеллектом, не стоящим даже рядом с шестым уровнем, вам нравятся рисунки, примеры и фрагменты кода и есть кто-то для объяснения всего этого: добро пожаловать в книгу Web Design Garage.

Возможно, пригодится пара слов об организации книги. Книга Web Design Garage немного похожа на объектно-ориентированное программирование, в котором происходит погружение в библиотеку (или книгу, как это было); найдите то, что необходимо; смешивайте, сопоставляйте и объединяйте; и возвращайтесь к работе. Вместо традиционных глав используются темы: восемьдесят шесть, если быть точным. Темы могут быть короткими, кусочками, размером со статью, и размышлениями, размером с главу, в зависимости от обсуждаемого предмета. Эти темы разделены на восемь общих категорий или частей, чтобы вы смогли найти необходимую информацию максимально быстро.

Для наилучших результатов не читайте эту книгу от корки до корки – вы можете обнаружить, что многое можно пропустить. Иногда материал повторяется, чтобы сделать темы максимально независимыми.

Наилучший подход заключается в обращении к тем темам, которые помогут решить определенные проблемы, возникшие с вашим сайтом, и построить курс, основываясь на этих темах. Если в любой данной теме есть ссылка на методы дизайна, например, группирование или технология, наподобие Каскадных Таблиц Стилей (Cascading Style Sheets), с которыми вы не знакомы, вы можете перейти к темам, в которых описываются данные методы, и расширить свой кругозор.

Разные дороги стремятся к объединению, особенно в случае с Web-дизайном, и все они ведут к одному и тому же месту (более или менее), поэтому вы в конце концов пройдете по всей книге. Однако вы сделаете это, следуя своим собственным путем, а не малопонятным лабиринтом ума автора. Надеюсь, что вы после прочтения книги Web Design Garage уйдете с чем-то стремящимся к нелинейному опыту, сильно напоминающим серфинг по Интернету.

Что касается меня, то мне нравится думать, что книга Web Design Garage – это нечто вроде вымышленного приключенческого рассказа, одного из тех – «выбери свою собственную линию рисования» – типов. Вы, герой, пытаетесь связать вместе части волшебного заклинания, которое спасет королевство. Чтобы сделать это, вам придется сосредоточенно изучать скудно документированные древние формулы. Есть даже периодически появляющийся злодей: отдел по продажам, который появляется тогда, когда ожидаешь его менее всего, чтобы отнять ваш сайт и использовать его для промывки мозгов ваших посетителей с целью покупки вещей. Так или иначе, весь этот технический разговор – просто метафора. Не обращайте на это внимание, и вы обнаружите настоящую интересную книгу.

Если вы читаете эти слова и если я прав относительно того, кому понравится эта книга, то вы являетесь человеком с очевидным вкусом и с некоторыми априорными знаниями о том, как строить Web-сайт. Возможно, вы единственный доморощенный Web-дизайнер/Web-разработчик в вашем месте работы. Возможно, вы занимаетесь собственным бизнесом и вам необходимо создать Web-сайт для вашего бизнеса. Возможно, вы являетесь артистом и окружение Интернета вас заинтересовало. Возможно, вы просто любите технологии и на выходных вам нравится испытывать возможности, не подвластные человеку. Какой бы ни была ваша ситуация, в большинстве тем делается предположение, что вы в основном знакомы с необходимыми процедурами, например, создание HTML-документов и работы со скриптами. Если вы уже знаете немного, эта книга поможет вам узнать гораздо больше.

Однако, если я не прав, что вполне вероятно, пожалуйста, начните рассмотрение книги Web Design Garage с части 8: Темы основного обучения. Когда я писал эту часть, я притворялся, что не знаю совсем ничего, и пытался объяснить основные понятия ремесла самому себе наиболее понятным способом. Идея заключается в том, что если вы до этого никогда не видели букв HTML, кроме данного предложения, то, начав с части 8, вы сможете продолжить рассмотрение оставшихся частей книги.

И, какого черта, если даже вы знаете одну или две вещи, то в любом случае можете взглянуть на часть 8: просто, чтобы освежить память. Множество Web-дизайнеров, включая меня, являются самоучками. Мы все имеем эти небольшие пробелы в образовании. Возможно, часть 8 прояснит для вас некоторые вещи, как Барни Мариспини (Barney Marispini) и Роб Стритер (Rob Streeter) (технические редакторы данной книги) помогли мне закрыть несколько выбоин в моем собственном образовании. Я обязан вам, парни. На самом деле, я обязан вам по отдельности.

Находясь вдалеке от них, я был бы недобросовестным, если бы не отметил Джона Нейдхарта (John Neidhart) из компании Barbs, Джона Фуллера (John Fuller), Ракель Каплан (Raquel Kaplan), Робина О'Брайена (Robin O'Brien), Катлида Аддиса (Kathleed Addis), Джули Нагил (Julie Nahil), Дмитрия Коржа (Dmitri Korzh) и всех из издательства Prentice Hall PTR, кто принимал участие в процессе публикации книги. Также выражаю благодарность обычным подозреваемым компании Studio B, как старым, так и новым: Нэйл Салкайнд (Neil Salkind), Линн Холлер (Lynn Haller), Стэйси Барон (Stacey Barone), Катрина Хиллстен (Katrina Hillsten) и Джэки Коде (Jackie Coder).

Особая признательность и чувство долга исходит непосредственно от меня тем людям, с которыми я не имел удовольствия встретиться виртуально или реально, чьих имен я не знаю, но чьи усилия содействовали заботе, питанию и выпуску данной книги и ее автора, напоминающих супергероев, анонимно защищающих город.

ЧАСТЬ 1.
Дизайн и удобство использования

- Разработка дизайна, облегчающего навигацию
- Группирование схожих функций
- Разработка рисунков для щелчков
- Достижение сбалансированности
- Выбор цветов
- Разработка для специальных возможностей
- Оставление «хлебных крошек»
- Использование меню переходов
- Управление всплывающими окнами
- Создание открывающихся меню
- Обеспечение путей отхода
- Работа со скриптами и таблицами стилей
- Дневник: Внедрение или присоединение?
- Запуск скриптов при помощи ссылок
- Дневник: JavaScript-ссылки или onClick-ссылки?

ГЛАВА 1.
Разработка дизайна, облегчающего навигацию

Дизайн в Интернете на самом деле не ограничивается предоставлением посетителям чего-то привлекательного, на что приятно посмотреть. Он играет более важную роль: создает ощущение места. Хороший дизайн гармонично объединяет страницы сайта. Другими словами, дизайн обозначает «место размещения» сайта. Он проводит границу вокруг определенного места в Интернете и сообщает: «Здесь все гармонично».

Это ощущение места достигается при помощи единообразного использования элементов дизайна по всему сайту. Такие элементы страниц, как макет, цветовая схема и стиль текста, должны быть зрительно связаны. Если страницы сайта используют различные подходы к дизайну, независимо от того, насколько эффективным может быть дизайн этих отдельных страниц, будет создано ощущение места сродни торговому центру. Каждая сверкающая витрина магазина соревнуется с другими витринами без видимого смысла или цели. Это визуальная катастрофа. Однако когда страница за страницей применяется некоторый единообразный дизайн, получается нечто наподобие высококлассного универсального магазина. Поднимается ваш уровень. Привлекаются более серьезные клиенты. Когда посетители перемещаются по сайту, им кажется, что они ходят с этажа на этаж и из отдела в отдел, которые находятся под одной крышей. Более того, способ использования дизайна может помочь посетителям понять, в каком отделе они находятся, не ища планы этажей около лифтов.

Вот как это работает. Содержимое большинства сайтов организовано в соответствии с определенной **иерархией** или логической структурой; это означает, что разработчики сайта разбили представляемую информацию на категории и подкатегории, например:

- Фильмы
  - Боевик
  - Комедия
  - Драма
  - Мелодрама
  - Мистика/Приключения
  - Ужасы
  - Научная фантастика
- Музыка
  - Популярная
  - Джаз
  - Классическая
- Игры
  - Персональный компьютер
  - Приставка
- О нас

Часто задаваемые вопросы

Как насчет сайтов, на которых посетители находят нужную информацию, осуществляя поиск вместо использования навигации? Имеют ли данные сайты иерархическую структуру?

Да, почти всегда. Даже содержимое огромных сайтов, например, Amazon и eBay, организовано иерархически. Пока большинство посетителей использует поиск для нахождения необходимой информации, отважные люди, желающие просмотреть категории, могут сделать это.

При разработке такого сайта иерархия выполняет большую часть работы. Элементы списка самого высокого уровня становятся элементами выбора в навигации, как показано на рисунке 1.1. Это этажи в вашем высококлассном универсальном магазине. Теперь уже известно, что для каждой из основных категорий необходима страница на сайте. Как насчет подкатегорий? Это отделы в вашем универмаге; на каждом этаже здания может быть расположено несколько отделов. Для каждой из подкатегорий также нужна страница на сайте. Возможно, также понадобятся под-подкатегории или под-под-подкатегории, в зависимости от глубины и детализации иерархии сайта. Без проблем. Считайте, что для каждого элемента списка необходима одна страница.

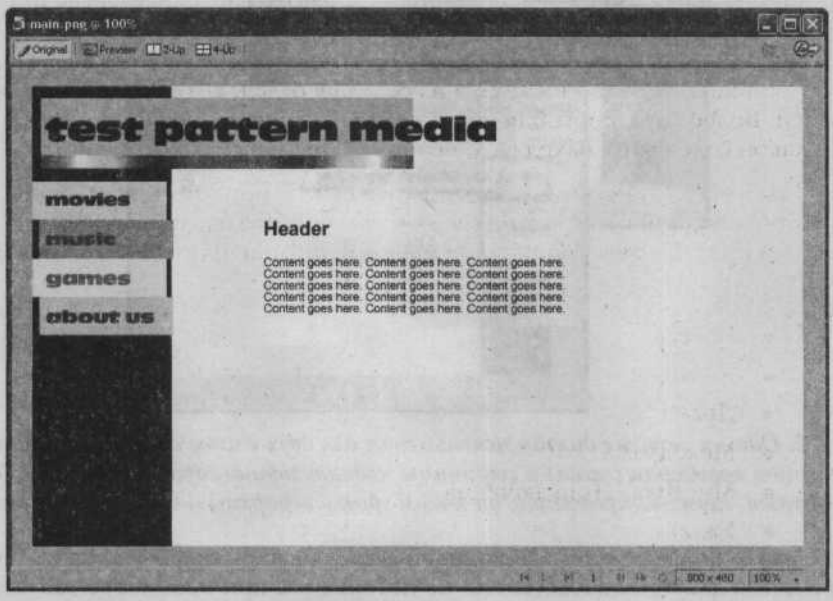


Рис. 1.1. Элементы верхнего уровня иерархии сайта представляют категории для навигационной панели (Navigation bar)

Преимущество иерархической структуры заключается в том, что она проводит посетителей именно туда, куда они желают попасть. Навигация доставляет их на необходимый этаж, а подкатегории направляют их к необходимому отделу на этом этаже. Это быстро, просто и наглядно, как при перемещении по реальному зданию. Ваша задача, как дизайнера, заключается в предельном улучшении данного эффекта. Вам нужно, чтобы графическое представление сайта максимально соответствовало организационной структуре. Хитрость заключается в том, чтобы сделать страницы

для каждого уровня иерархии отличающимися визуально, при этом придерживаясь общего единообразного вида и функциональности.

Определение

Уровень (Level) – это общее название для каждого раздела иерархии. Например, если сайт углубляется до под-подкатегорий, в иерархии присутствует три уровня (основная категория, подкатегория и под-подкатегория).

Посмотрите на дизайны страницы, представленные на рисунке 1.2. Они унифицированы и абсолютно одинаковы. Однако обратите внимание, что левая является страницей верхнего уровня – этажом здания – в то время, как правая является страницей подкатегории – отделом на этом этаже. В дизайне нет даже намека на положение страниц в иерархии. В одном случае дизайн определяет основную категорию. В другом случае дизайн определяет подкатегорию.

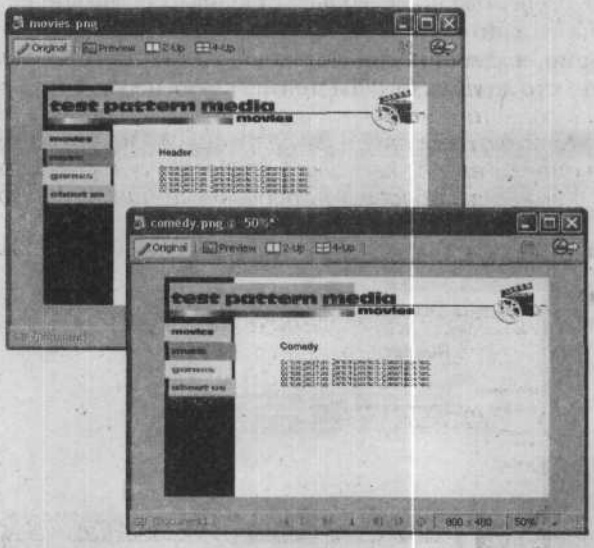


Рис. 1.2. Один и тот же дизайн используется для двух типов страниц: страницы основной категории (слева) и страницы подкатегории (справа). Если дизайн единообразен, сложно определить, на каком уровне иерархии находится посетитель

Серьезное дело, не правда ли? По крайней мере, дизайн унифицирован. Навигация понятна и легко доступна. Посетители не потеряются при просмотре сайта. Ну, скорее всего. Если посетители вошли через переднюю дверь, т.е. если они появились на первой странице сайта и перемещались по категориям и подкатегориям – возможно, все будет в порядке. Однако Web-сайт – это не реальное место. Оно является виртуальным. Это место обладает свойствами, которыми реальные места не обладают. Так, посетители не обязаны входить через переднюю дверь. Благодаря поисковым системам, закладкам и обмену ссылками между друзьями по электронной почте, посетители могут материализоваться в дальних уголках вашего сайта, не имея никакого понятия о том, где они находятся по отношению ко всему остальному. Однако если ваш дизайн намекает на определенное положение и общий смысл места, это может помочь посетителям получить представление о сайте, независимо от того, как они туда попали.

Совет

Для каждого уровня иерархии сайта создавайте особый, но при этом схожий, дизайн страницы.

Сравните предыдущие дизайны с дизайном, представленным на рисунке 1.3. На этом рисунке дизайн сам усиливает логическую структуру сайта. Страница подкатегории, расположенная справа, имеет признаки страницы основной категории, расположенной слева, однако ее дизайн достаточно отличается, чтобы понять, что страница подкатегории является другим видом страницы. Придайте всем страницам подкатегорий подобный вид, используя при этом дизайн основной категории для страниц основных категорий, и можете быть уверены в успехе.

Совет

Даже не думайте о разработке сайта, предварительно не набросав его структуру. Не обязательно знать каждую отдельную категорию или подкатегорию. В любом случае, они, вероятно, будут изменяться множество раз в процессе разработки. Однако следует знать, хотя бы приблизительно, насколько глубокой будет иерархия, чтобы иметь возможность обеспечить особый, но при этом схожий, дизайн для страниц каждого уровня.

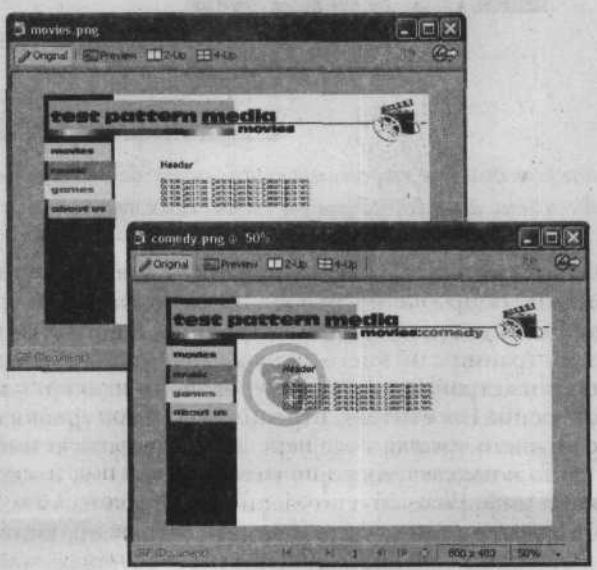


Рис. 1.3. Два различных, но при этом схожих, образца дизайна – один для страницы основной категории (слева), а второй – для страницы подкатегории (справа), – помогают посетителям понять, где они находятся при навигации по сайту

Чтобы придать отличительные особенности дизайну, можно пойти несколькими различными путями. Страницы, представленные на рисунке 1.4, при перемещении вглубь сайта становятся более загруженными, и на них используется больше графических элементов. Данная схема отлично подходит, когда иерархия спускается из общих категорий к все более и более конкретной информации, как при перемещении из общей категории, например, «Книги», к жанру книги, скажем, фантастике, а затем к конкретному названию в этом жанре, например, The Lord of the Rings (Властелин колец). Все более усложняющийся дизайн наводит посетителей на мысль, что с каждым шагом они становятся ближе и ближе к требуемой информации, которая находится на страницах нижнего уровня.

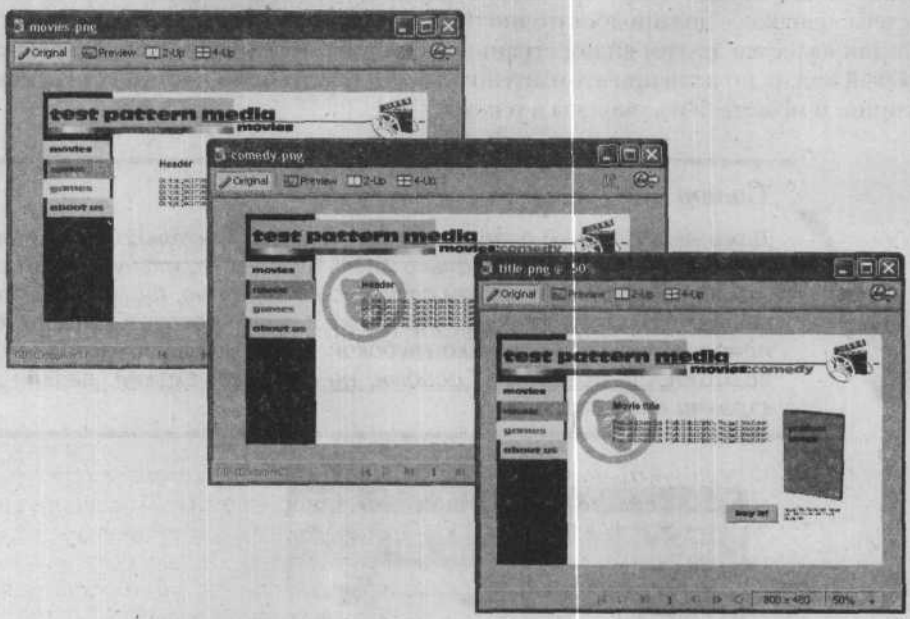


Рис. 1.4. В данном дизайне страницы становятся более загруженными при перемещении вглубь сайта, а информация на страницах становится более конкретной

Если ваш сайт сосредотачивается на информации, расположенной на страницах верхнего уровня, можно попробовать дизайн, представленный на рисунке 1.5. В данном случае при перемещении вглубь сайта страницы становятся менее сложными. Предполагается, что страницы нижнего уровня уточняют или расширяют информацию, представленную на странице верхнего уровня, но являются практически независимыми. С точки зрения посетителя, страница верхнего уровня предоставляет товары, а страницы нижнего уровня содержат дополнительную информацию. Такой тип структуры отлично подходит для корпоративных сайтов, поскольку такие сайты предоставляют общую информацию о компании или организации. Однако эта схема не подходит для сайтов электронной коммерции, например, книжных магазинов и магазинов, торгующих аудио- или видеозаписями, для которых наиболее важны страницы нижнего уровня – определенные продукты для продажи.

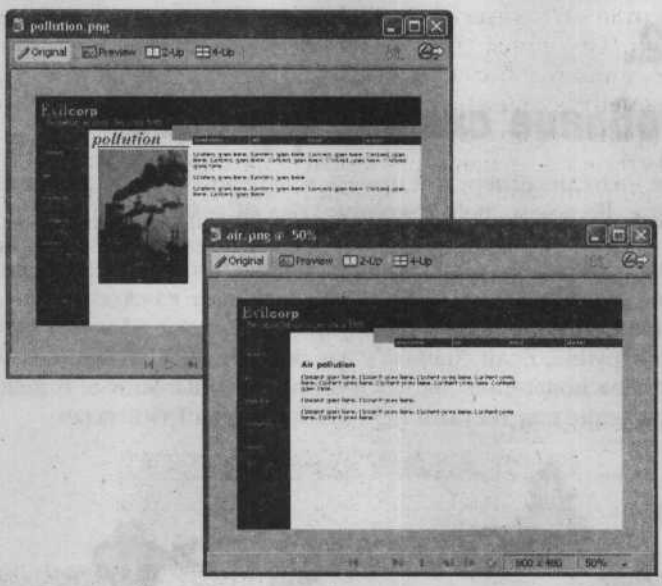


Рис. 1.5. Если желаете сфокусировать внимание на странице верхнего уровня, используйте менее сложный дизайн на страницах нижнего уровня

Совет

Используйте наилучший дизайн на страницах, к которым хотите привлечь максимальное внимание. Если наиболее важны страницы нижнего уровня, как, например, на сайте электронной коммерции, стройте дизайн по направлению к нижнему уровню. Однако если наиболее важны страницы верхнего уровня, как, например, на корпоративном сайте, сделайте акцент на страницах верхнего уровня.

ГЛАВА 2.
Группирование схожих функций

Группирование – это дизайнерский прием, в котором схожие элементы помещаются друг возле друга. Возьмем любое количество объектов, например, набор на рисунке 2.1. Если их расположить равномерно по всему пространству дизайна, они будут выглядеть, как случайные объекты – ни больше, ни меньше. Однако если их сгруппировать, как показано на рисунке 2.2, глаз выделяет каждую группу как отдельную единицу, и мозг пытается понять, почему эти отдельные объекты принадлежат данной кон-кретной группе. Если сначала у нас была группа случайных, бессмысленных объектов, то теперь появились две логические группы: живые и неживые объекты. Люди, размышляющие над тестами I.Q., любят данный тип задач.

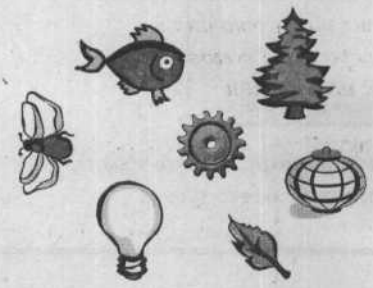


Рис. 2.1. Возьмите любой набор случайных, бессмысленных объектов

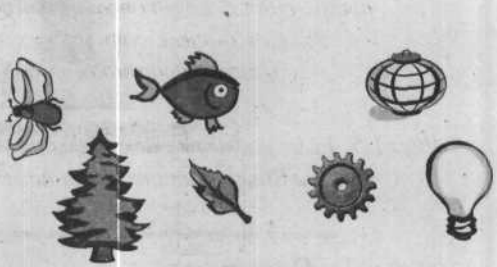


Рис. 2.2. Соберите объекты в группы, и мозг свяжет их логически

Группирование является простым и интуитивно понятным. Оно работает по принципу ассоциаций и необыкновенно эффективно. И его можно использовать в Web-дизайне.

На самом деле Web-сайт является одним из наилучших мест для применения группирования. Почему? Чтобы не заходить далеко, возьмем простую гиперссылку, являющуюся основным элементом интерфейса компьютер-человек. Основным способом взаимодействия посетителей с сайтом являются щелчки на гиперссылках. Однако сами гиперссылки имеют, по крайней мере, три различных назначения. Щелчок часто приводит к загрузке новой страницы, но не всегда. Иногда щелчок на гиперссылке заставляет открыться окно программы для работы с электронной почтой. В другой раз щелчок запускает скрипт на языке JavaScript или серверное приложение. Все зависит от того, как запрограммировать ссылку.

Забудьте теперь об электронной почте и скриптах. Просто рассмотрим основное назначение гиперссылки, которое заключается в загрузке новых страниц, и вы по-прежнему найдете множество вариаций. Иногда страница загружается в текущем окне браузера. Иногда страница загружается в новом окне браузера. Иногда посетитель переходит на другую страницу сайта. Иногда посетитель переходит на совершенно другой сайт. И ссылки, инициирующие выполнение всех этих разнообразных действий, выглядят совершенно одинаково, как показано на рисунке 2.3. Только одна из этих ссылок открывает окно программы для работы с электронной почтой, и единственный способ ее найти – угадать.

К счастью, есть дизайнерский прием, позволяющий ассоциировать объекты логически, как показано на рисунке 2.4. Обратите внимание, как группирование намного

упростило выбор ссылки для открытия окна для работы с электронной почтой, хотя все гиперссылки выглядят одинаково и используют один и тот же способ взаимодействия – одиночный щелчок мышью. Если у вас есть объекты, которые выглядят и действуют одинаково, но выполняют различные функции, группирование необходимо для их различения.

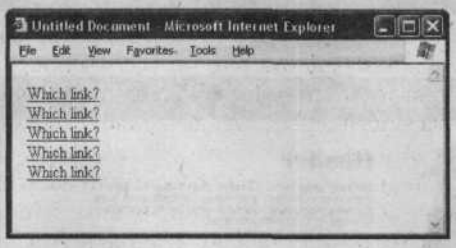


Рис. 2.3. Отвечайте быстро! Какие из приведенных гиперссылок заставляют открыться окно почтовой программы? Нельзя узнать это иначе, кроме как щелкнуть на них, поскольку гиперссылки всегда выглядят одинаково, независимо от их назначения

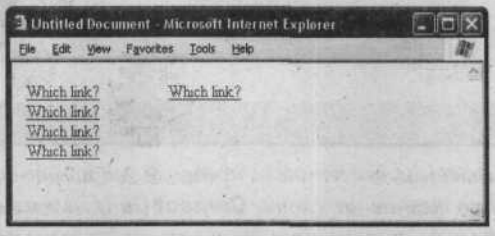


Рис. 2.4. Примените прием группирования к этому бессистемному набору объектов, вы, возможно, сможете догадаться, что ссылка, расположенная справа, открывает окно почтовой программы

Можно использовать возможности группирования для оптимизации интерфейса вашего Web-сайта. Взгляните на навигационную панель, показанную на рисунке 2.5. Названия на кнопках могут быть ясными и простыми для понимания, однако сами кнопки сгруппированы не самым эффективным способом. Щелчок на первых четырех кнопках приводит к загрузке новых страниц, тогда как щелчок на пятой кнопке открывает окно программы для работы с электронной почтой. Это обрывает логику вашего дизайна, поскольку посетители ожидают, что после щелчка на кнопке **Contact Us** (Свяжитесь с нами) загрузится страница.

Лучше отделить кнопку **Contact Us** (Свяжитесь с нами) от остальных кнопок и перенести ее в какое-либо другое место интерфейса, как показано на рисунке 2.6. Еще лучше изменить целиком внешний вид кнопки **Contact Us** (Свяжитесь с нами), как показано на рисунке 2.7. Это усилит мысль, что кнопка **Contact Us** (Свяжитесь с нами) имеет другое назначение, не связанное с загрузкой страницы.

На навигационной панели остаются четыре кнопки. Что произойдет, если первые три кнопки связаны со страницами на текущем сайте, а четвертая кнопка открывает новое окно браузера и переходит на другой сайт, где посетители могут проследить за выполнением своих заказов? И хотя все четыре кнопки загружают новые страницы, вы, возможно, пожелаете отделить четвертую кнопку от трех остальных просто

потому, что она функционирует немного по-другому. Но где поместить четвертую кнопку? Нежелательно группировать ее с кнопкой **Contact Us** (Свяжитесь с нами), как показано на рисунке 2.8, поскольку посетители предположат, что эти кнопки имеют сходное назначение, хотя на самом деле это не так.

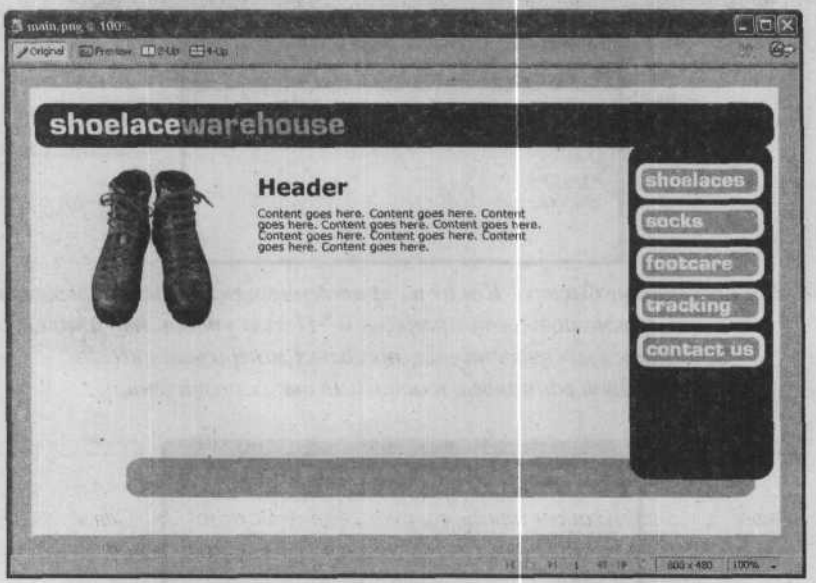


Рис. 2.5. Из-за группирования посетители предполагают, что все пять кнопок имеют одно назначение. Однако щелчок на кнопке ***Contact Us*** *(Свяжитесь с нами) открывает окно почтовой программы, а щелчок на других кнопках загружает новые страницы*

Часто задаваемые вопросы

Группирование – это просто дизайнерский прием?

Безусловно, нет. Это фундаментальный психологический принцип. В наших мозгах зашита логика поиска смысла в группах.

В начале президентской кампании 2004 года Республиканская партия издала фотографию, на которой юный Джон Керри (John Kerry) стоит на несколько рядов ниже печально известного либерала Джэйн Фонда (Jane Fonda) на митинге в защиту мира в период войны во Вьетнаме. На фотографии мистер Керри и госпожа Фонда не разговаривают. Они не вступают в сговор против правительства, не планируют встречи на Кубе и не делают ничего, что взбесило бы консерваторов. Так зачем же публиковать фотографию? Все просто – группирование. Ассоциация. Показав Джона Керри рядом с Джэйн Фонда, республиканцы надеялись опорочить его образ.

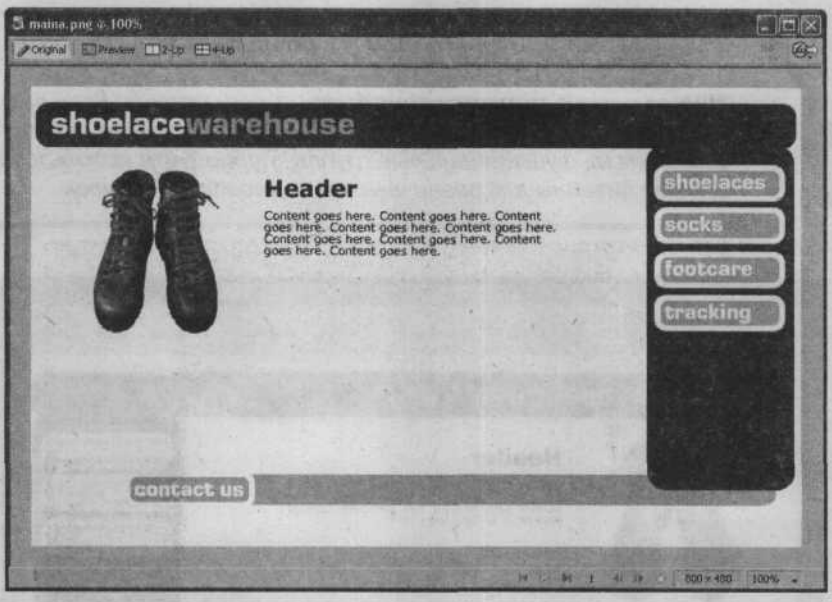


Рис. 2.6. Отделите кнопку **Contact Us** *(Свяжитесь с нами), и вашим посетителям станет понятно, что кнопка* **Contact Us** *(Свяжитесь с нами) имеет другое назначение, нежели остальные кнопки*

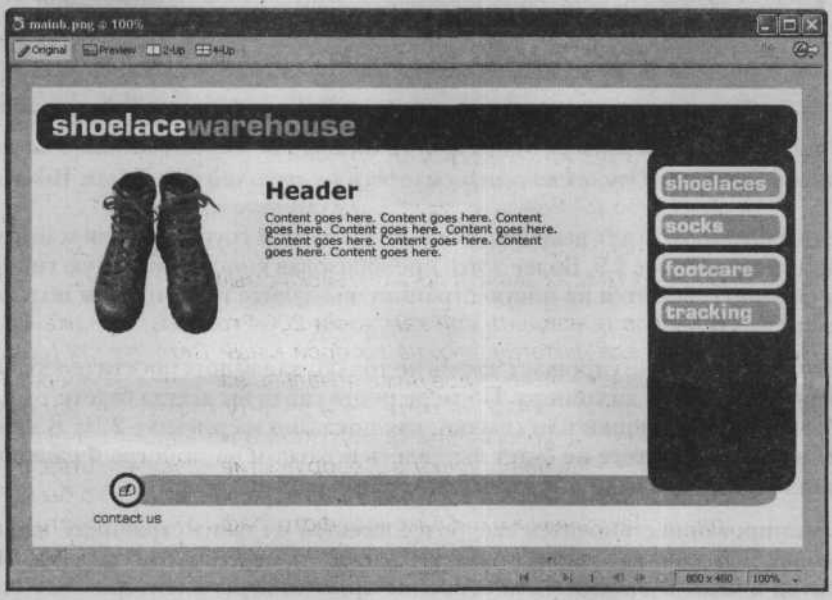


Рис. 2.7. При изменении внешнего вида кнопки **Contact Us** *(Свяжитесь с нами) усиливается мысль, что кнопка выполняет другой тип действий*

Совет

Иногда одного группирования бывает недостаточно. Если ваш интерфейс все еще выглядит неуклюже после разделения элементов управления на функциональные группы, попробуйте использовать различные дизайны для различных типов кнопок или ссылок.

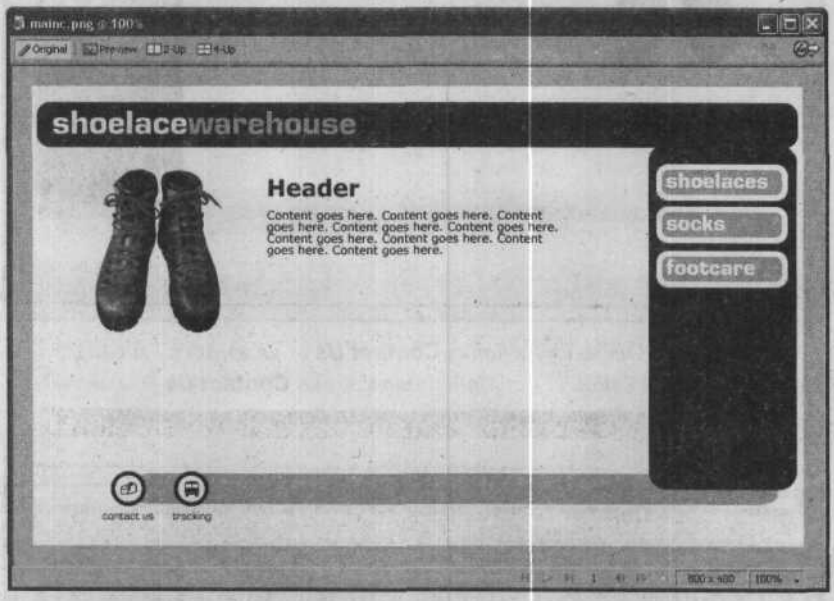


Рис. 2.8. Не добавляйте кнопки в одну группу, если они не выполняют схожие функции. Кнопка ***Tracking*** *(Отслеживание) ссылается на страницу на другом Web-сайте*

Наилучшим выбором будет выделение места для новой группы в вашем интерфейсе, как показано на рисунке 2.9. Более того, преобразовав кнопку в простую гиперссылку, которая обычно ссылается на новую страницу, вы дадите посетителям подсказку о ее назначении.

Хорошее группирование упрощает жизнь не только для ваших посетителей. Она упрощает жизнь для вас как дизайнера. По мере роста сайта вы всегда будете точно знать, куда добавить новые кнопки или ссылки, как показано на рисунке 2.10. В этом случае ваш сайт в конечном итоге не будет выглядеть похожим на монстра Франкенштейна. Сам дизайн исключает излишнюю мыслительную работу.

Метод группирования становится еще более важным на таких страницах, как корзины электронных магазинов со множеством элементов управления со ссылками. Помните, что удобство в использовании – это основное правило игры в Интернете. Попробуйте поработать с интерфейсом корзины, который представлен на рисунке 2.11. Дизайн делает использование корзины мучительным. В данном интерфейсе ничто не сгруппировано в соответствии с назначением, поэтому посетители вынуждены читать название каждой кнопки, чтобы понять, что им делать. Чем больше вы заставляете их останавливаться и думать о том, что они делают, тем меньше вероятность, что они сделают покупку.

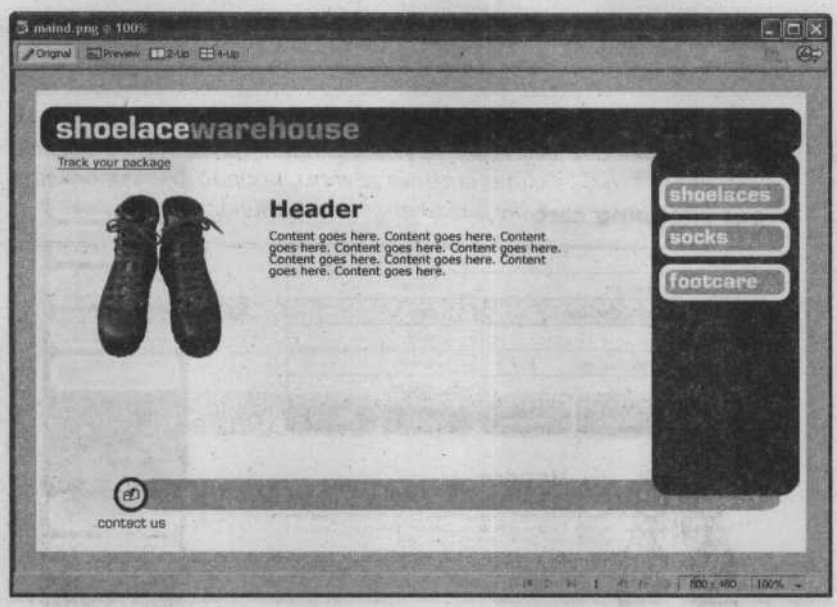


Рис. 2.9. Вместо этого создавайте отдельные группы для отдельных функций

Рис. 2.10. Благодаря группированию, вы всегда точно знаете, куда добавить новые кнопки или ссылки

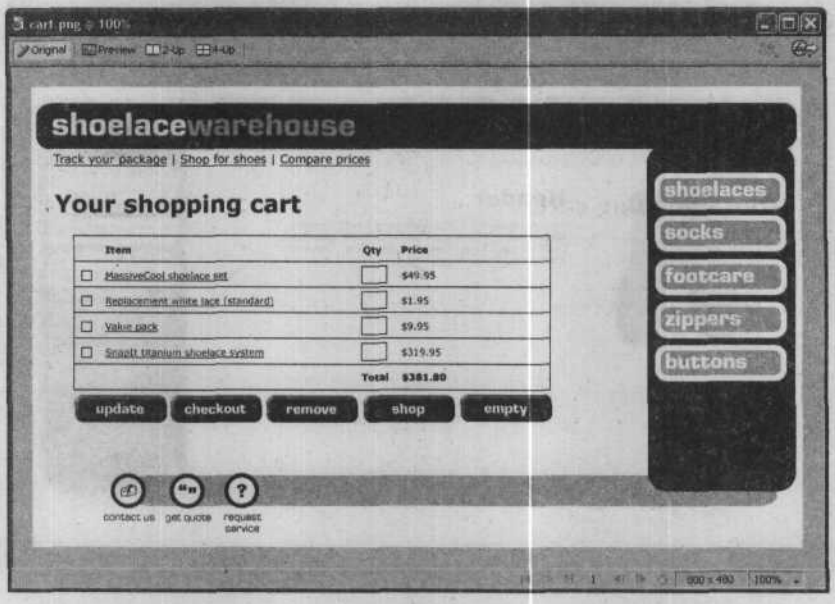


Рис. 2.11. Посетителям тяжело использовать эту корзину, поэтому они не захотят ее использовать вообще. Прощайте, продажи

Небольшое группирование улучшает этот интерфейс, как показано на рисунке 2.12. Организация кнопок в соответствии с их назначением делает работу с корзиной проще и более интуитивно понятной. А еще лучше доработать дизайн кнопок, чтобы усилить мысль об их различных назначениях, как показано на рисунке 2.13.

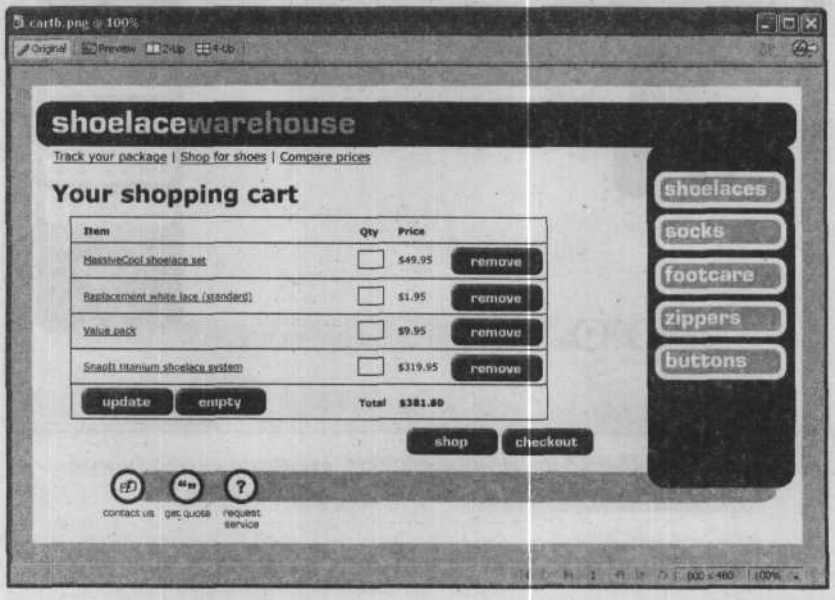


Рис. 2.12. Корзина стала более понятной после организации кнопок в соответствии с их назначением

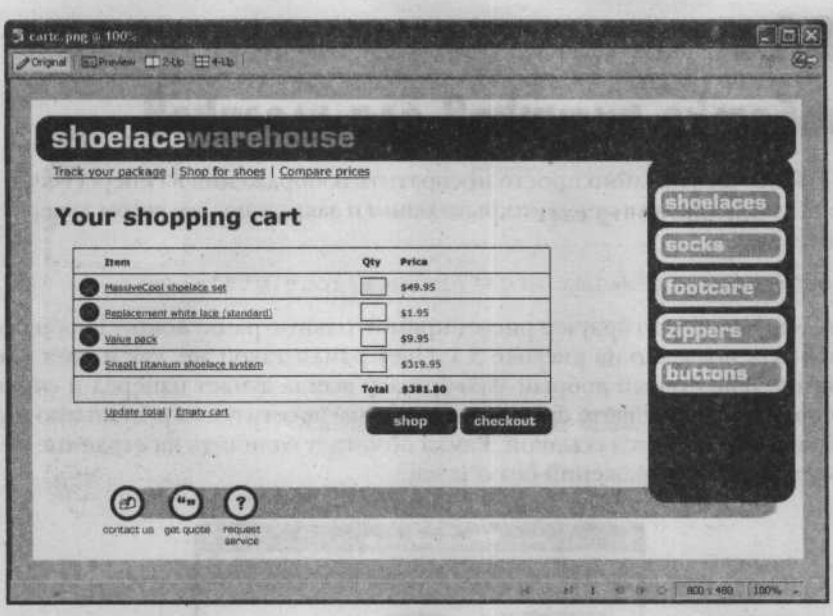


Рис. 2.13. Еще лучше использовать различный дизайн для различных типов кнопок

ГЛАВА 3.
Разработка рисунков для щелчков

В языке HTML чрезвычайно просто превратить изображения в гиперссылки. Просто вставьте тег изображения между открывающим и закрывающим тегом анкера, наподобие следующего:

```
<a href="index.htm"><img src="/images/logo.gif"></a>
```

Обратите внимание, что браузер рисует прямоугольную рамку вокруг изображения гиперссылки, как показано на рисунке 3.1. Цвет рамки такой же, как и цвет текстовых гиперссылок. Ваш старый добрый Web-браузер всегда думает наперед и справедливо предполагает, что вы желаете предоставить своим посетителям визуальную подсказку, что изображение является ссылкой. Рамка помогает отличить на странице изображения со ссылкой от изображений без ссылки.

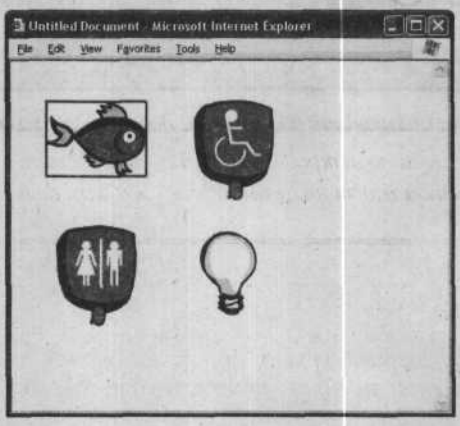


Рис. 3.1. После превращения изображения в гиперссылку, браузер нарисует рамку вокруг него, сообщая посетителям, что данное изображение имеет ссылку. Обратите внимание, как рамка выделяет изображение с гиперссылкой из других изображений

Совет

Имеет смысл использовать рамку гиперссылки тогда, когда ваше изображение ссылается на полноразмерную версию графического файла.

Но на всех сайтах, которые вы посещали в Интернете, сколько изображений со ссылкой, использующих встроенную рамку, вы видели? Не очень много. Рамка является неуклюжей и безобразной и стремится повлиять на поток зрительной информации страницы. Вот почему большинство Web-дизайнеров предпочитает отключать рамку, как в следующем примере:

```
<a href="index.htm"><img src="/images/logo.gif" border="0"></a>
```

Присваивание атрибуту `border` значения 0 решает одну проблему, но при этом создает другую, как продемонстрировано на рисунке 3.2. На изображении с гиперссылкой все еще можно щелкать, однако без рамки, привлекающей внимание, посетители не имеют никакого понятия о том, какое изображение содержит ссылку, пока не переместят указатель мыши над всеми изображениями. Ваши посетители не простят вам этого. Все, что заставляет их останавливаться и думать, является приглашением покинуть ваш сайт.

Рис. 3.2. Можно отключить встроенную рамку гиперссылки, однако в этом случае изображение со ссылкой будет выглядеть так же, как остальные изображения

Совет

Атрибуту `border` можно присвоить любое значение, а не только 0. Чем большее значение используется, тем толще становится рамка.

Хитрость здесь заключается в следующем: необходимо сделать так, чтобы изображения со ссылками выглядели как ссылки, не прибегая при этом к помощи безобразной рамки. В этом отношении отлично подойдут две методики: контекст и намеки.

Размещение изображений со ссылками в правильном контексте на странице значительно способствует их восприятию как элементов со ссылками. Рассмотрим макет, представленный на рисунке 3.3. Из пяти представленных изображений три изображения со ссылками, а два – без ссылок. Если подумать несколько секунд, какие изображения со ссылками, а какие – без ссылок, вы, возможно, догадаетесь, однако ваши посетители не будут такими же терпеливыми, как вы. Чтобы поддерживать их счастливое погружение в ваш сайт, попробуйте использовать более разумный макет, показанный на рисунке 3.4. Отделив изображения со ссылками от изображений без ссылок, вы создаете на странице интерфейс миниатюрной панели управления, которая наводит на мысль о ссылках.

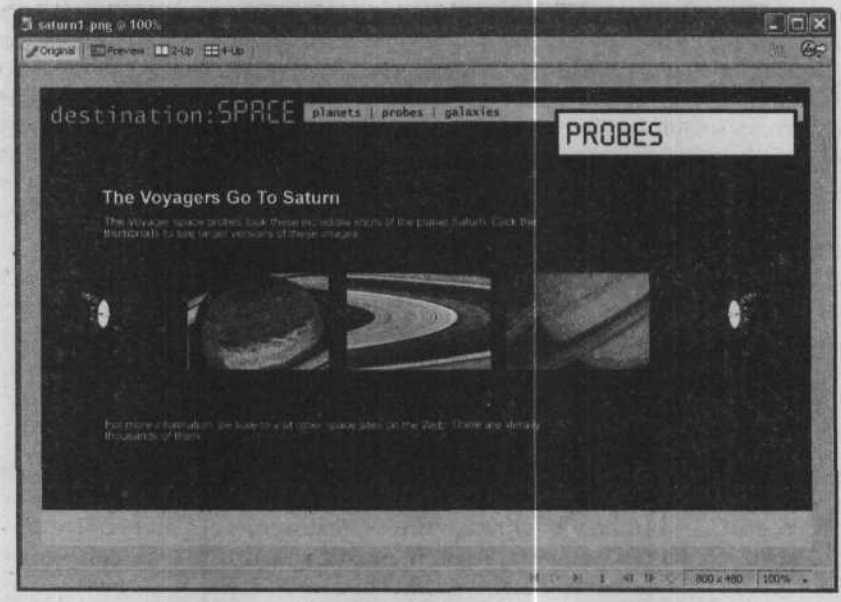


Рис. 3.3. Над этим макетом необходимо немного поработать. Три изображения со ссылками не просто отличить от двух изображений без ссылок

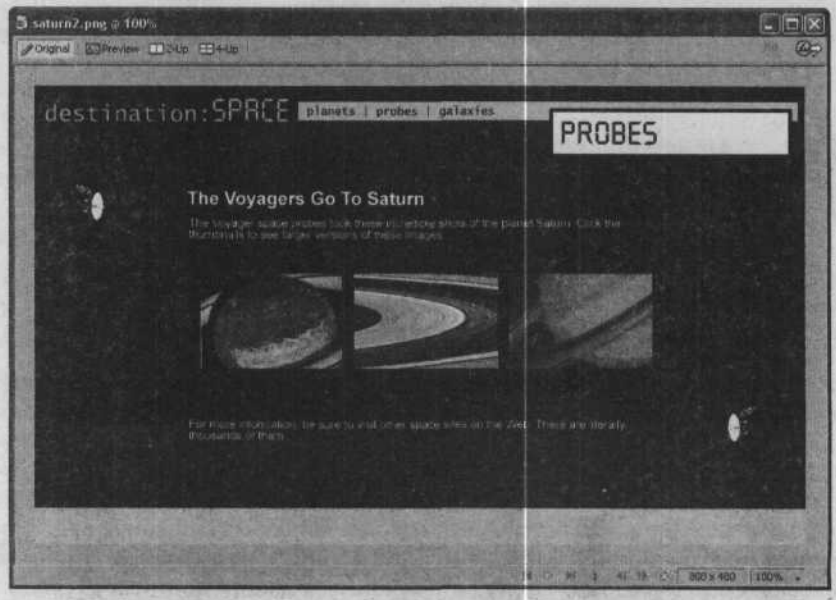


Рис. 3.4. О, гораздо лучше. Изображения со ссылками выделяются в данном контексте

Совет

Задать значение атрибута `border` *можно и для изображения без ссылки. Программа Netscape рисует рамку, используя цвет текста на странице по умолчанию. Программа Internet Explorer рисует рамку, используя черный цвет.*

Иногда правильного контекста бывает недостаточно, особенно когда дизайн перегружен изображениями. Для подобных ситуаций пригодятся намеки. **Намеки** – это визуальные подсказки в самих элементах дизайна, которые подсказывают или намекают на назначение элемента. Встроенная в браузер рамка гиперссылки, может быть, и не очень приятна с эстетической точки зрения, однако создает прекрасный намек, поскольку она визуально подсказывает посетителям, что данное изображение содержит ссылку.

Одним из наиболее распространенных намеков, используемых в Интернете, является моделирование кнопок. Оно так распространено, что многие дизайнеры равнозначно используют термины **кнопка** и **изображение со ссылкой**. Сделав так, что изображения со ссылками будут выглядеть как кнопки – объекты, на которые нажимают в реальном мире, – вы ассоциативно связываете действие по нажатию с вашим изображением. Следовательно, изображение привлекает щелчки мышью.

Посмотрим, что произойдет после замены уменьшенных изображений на кнопки, как показано на рисунке 3.5. Обратите внимание, как быстро исчезли все недавние сомнения относительно того, какие изображения на странице имеют ссылки. Дизайн кнопок помогает поместить их отдельно от изображений без ссылок, а явная форма кнопки способствует щелчкам.

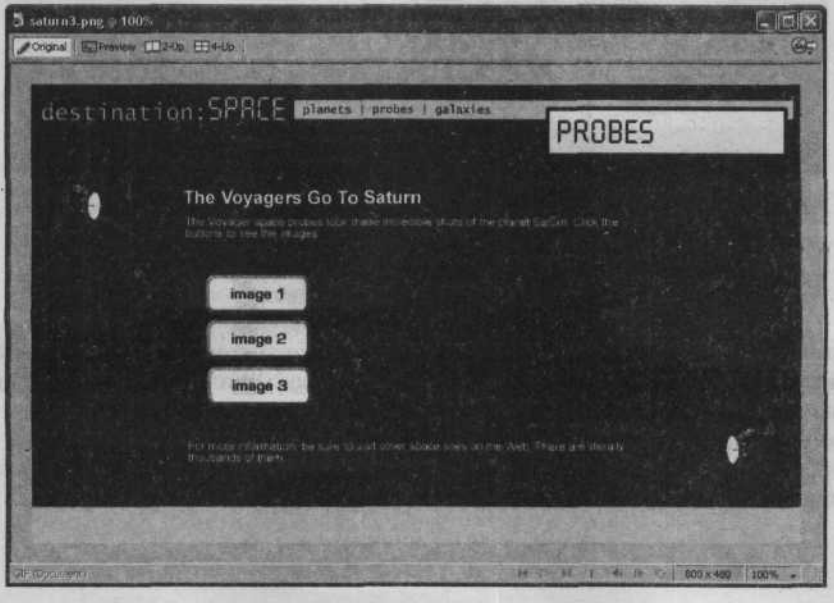


Рис. 3.5. Разработка изображений со ссылками, похожих на кнопки, – это верный способ привлечения щелчков

Определение

Намек – это визуальная подсказка, указывающая на назначение элемента дизайна.

Хотя кнопки и являются мощным инструментом, не считайте это само собой разумеющимся. Кнопки не сильно вам помогут, если посетители не смогут их найти, как показано на рисунке 3.6. Всегда старайтесь размещать кнопки так, чтобы они были заметны, как показано на рисунке 3.7. Желательно, чтобы ваши посетители ощущали, что ваш интерфейс находится под их управлением.

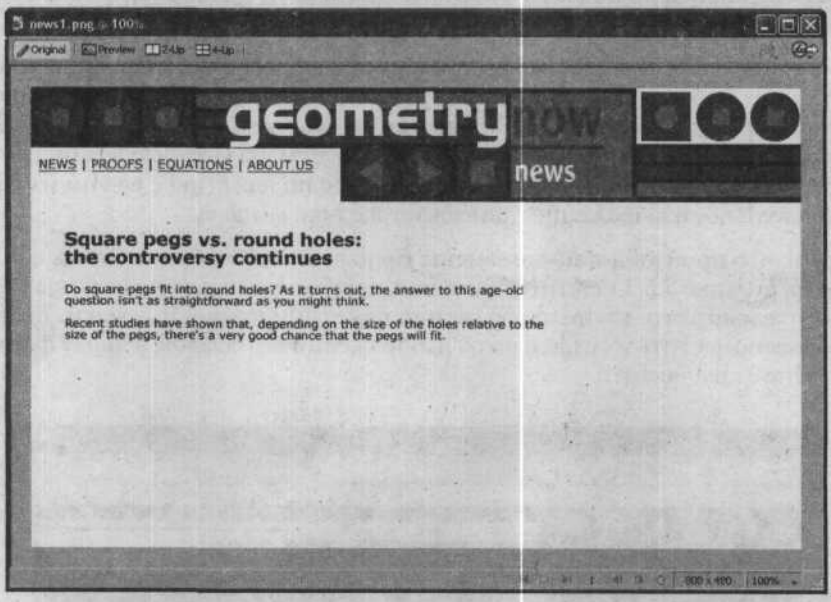


Рис. 3.6. Посетители предполагают, что переход по газетным материалам осуществляется с использованием кнопок со стрелками. Однако их цвет, дизайн и соседство с другими элементами дизайна затрудняют поиск этих кнопок

На рисунке 3.7 также показано, что использование значков или символов в качестве изображений со ссылками является еще одним эффективным намеком. В данном дизайне значки стрелок более непосредственны и интуитивно понятны, чем кнопки с надписями **Previous** (Предыдущий) и **Next** (Следующий). Вы можете заметить, однако, что значки обладают некоторыми характерными особенностями кнопок. Они выглядят трехмерно, что помогает им выделиться на странице.

Совет

Убедитесь, что кнопки без ссылок не похожи на кнопки, иначе ваши посетители будут тщетно щелкать на них, удивляясь, почему ничего не происходит.

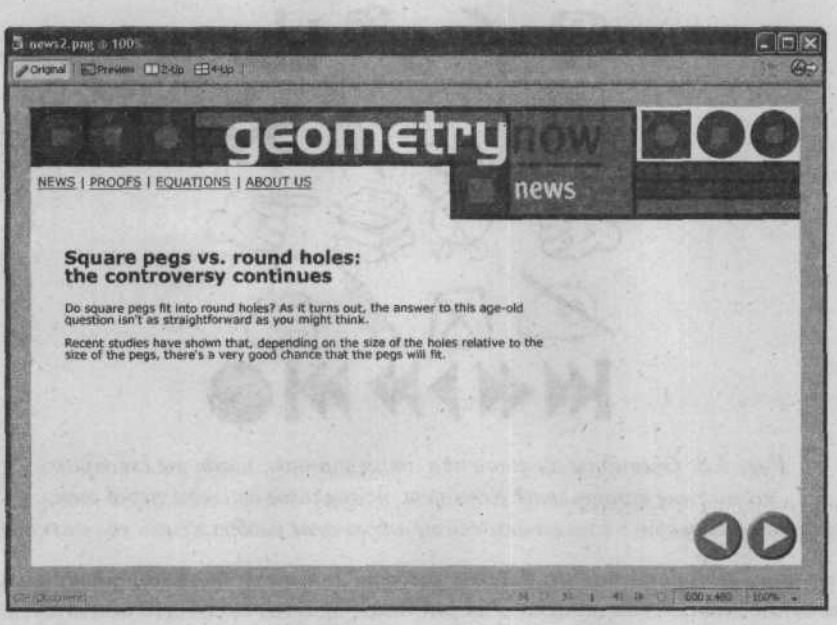


Рис. 3.7. Аккуратно располагайте кнопки. Контекст существенно меняет дело, даже если вы используете мощный намек, например, моделирование кнопок

Значки и символы работают лучше всего, когда их значение очевидно и общепринято. Символы в стиле DVD или CD для функций, например, **Play** (Проиграть), **Fast Forward** (Перемотка вперед), **Stop** (Остановить) и остальные, в основном, достаточно понятны. Это касается и использования значка корзины для функции **Delete** (Удалить) или значка с изображением листочка бумаги с отогнутым углом для функции **New Document** (Новый документ). Однако остальные значки, представленные на рисунке 3.8, могут иметь более вольную интерпретацию. Значок карандаша на некоторых сайтах может обозначать функцию `Create` (Создать), а на других сайтах – функцию `Edit` (Правка). Значок земного шара может обозначать функцию выхода в Интернет, но в другом случае может и не обозначать. Эффективность значков на вашем сайте зависит от того, как вы их используете, но хорошее практическое правило заключается в следующем: если существует какое-либо сомнение насчет значения определенного значка, не используйте его.

Рис. 3.8. Очевидны ли значения этих значков, когда вы смотрите на них вне контекста? Если нет, подумайте дважды перед тем, как использовать их на вашем сайте в качестве изображений со ссылками

Одним из способов решения проблемы двусмысленности является включение текста в изображение значка, как показано на рисунке 3.9. Этот подход используется в операционной системе Microsoft Windows и работает достаточно хорошо, когда речь идет о простоте использования. В операционной системе Windows все имеет значок для щелчка, однако у каждого значка есть название – на всякий случай. Если желаете, можете поэкспериментировать с таким подходом на вашем Web-сайте, однако если ваши значки работают только вместе с сопровождающими надписями, возможно, лучше не использовать только кнопки. Пользователи ожидают от изображений в рабочей среде, как Windows, одного, и совсем другого в информационной среде, такой, как Интернет. То, что пользуется успехом в одной среде, не всегда может быть перенесено в другую. Люди полагают, что можно щелкать на значках, расположенных на их рабочем столе, поскольку так работает операционная система Windows. Они не думают, что щелкать можно и на значках, расположенных на Web-сайте, где изображения часто используются только для информационных целей.

Определение

Метка – это текст на кнопке.

Совет

Распространенной практикой является создание корпоративных логотипов со ссылкой, особенно в том случае, когда логотип располагается на навигационной панели или где-либо в основном интерфейсе сайта. Логотип обычно связан с первой страницей сайта. Можно предположить, что посетители знают это из своего собственного опыта, поэтому нет необходимости демонстрировать, что логотип имеет ссылку. С другой стороны, добавление ролловерного эффекта или чего-нибудь в этом роде не повредит и, безусловно, укрепит мысль о возможности щелчков.

Рис. 3.9. Одним из способов решения проблемы двусмысленности является включение текстовой надписи в изображение значка, однако данный метод лучше подходит для среды операционной системы. В Интернете лучше использовать кнопки, чем значки

ГЛАВА 4.
Достижение сбалансированности

Сбалансированность – это путь дзэна. Сбалансированность также является важной дизайнерской характеристикой в Интернете. Если макет не сбалансирован, кажется, что с ним что-то не в порядке. Он визуально отталкивает, что продемонстрировано на рисунке 4.1. Однако если добиться сбалансированности, как показано на рисунке 4.2, все элементы макета будут выглядеть гармонично. Они кажутся подходящими друг другу. Web-страница обретает внутреннее спокойствие и гармонию, и это успокаивает посетителей.

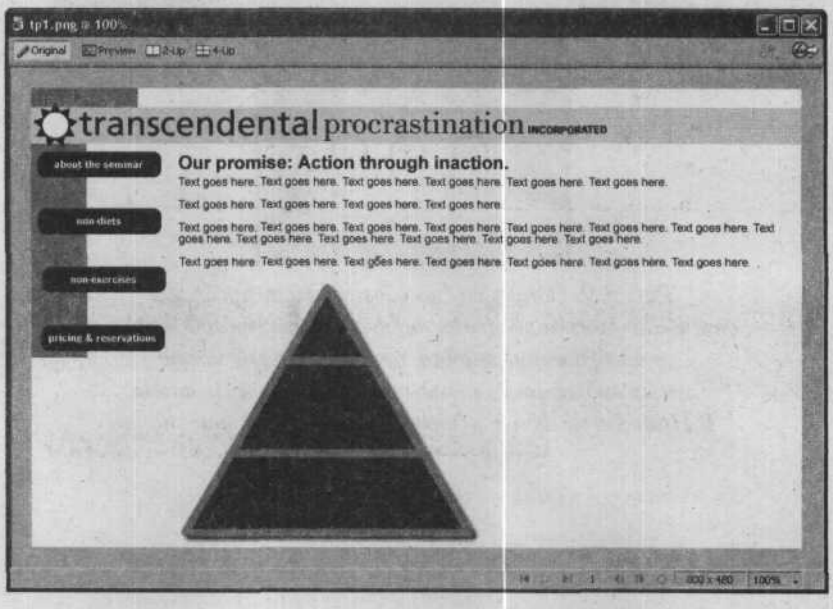


Рис. 4.1. Этот макет не кажется слишком хорошо сбалансированным

Сбалансированность дизайна предназначена не только для того, чтобы посетители чувствовали себя комфортно. Сбалансированный макет более удобен в использовании, чем несбалансированный макет. Он кажется более организованным, что помогает посетителям находить желаемую информацию. Взгляните на несбалансированные столбцы, показанные на рисунке 4.3. Поскольку столбец слева гораздо короче, чем столбец справа, посетители могут не подумать пролистать страницу, чтобы увидеть кнопки, расположенные внизу страницы, как показано на рисунке 4.4. Использование Web-страницы не должно напоминать игру в прятки. Если столбцы сбалансированы, как показано на рисунке 4.5, сам макет подсказывает, что продолжение содержимого страницы находится вне экрана.

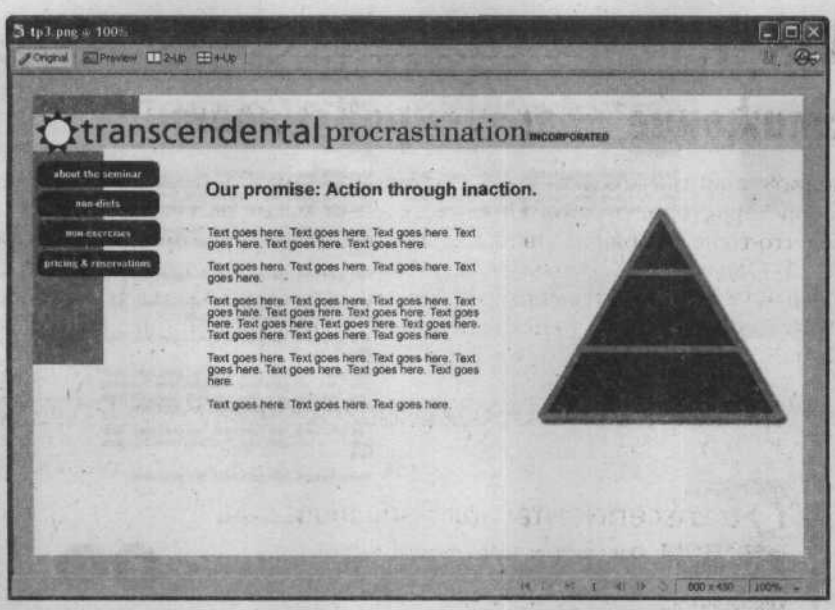


Рис. 4.2. Ощущения от сбалансированного дизайна приятнее

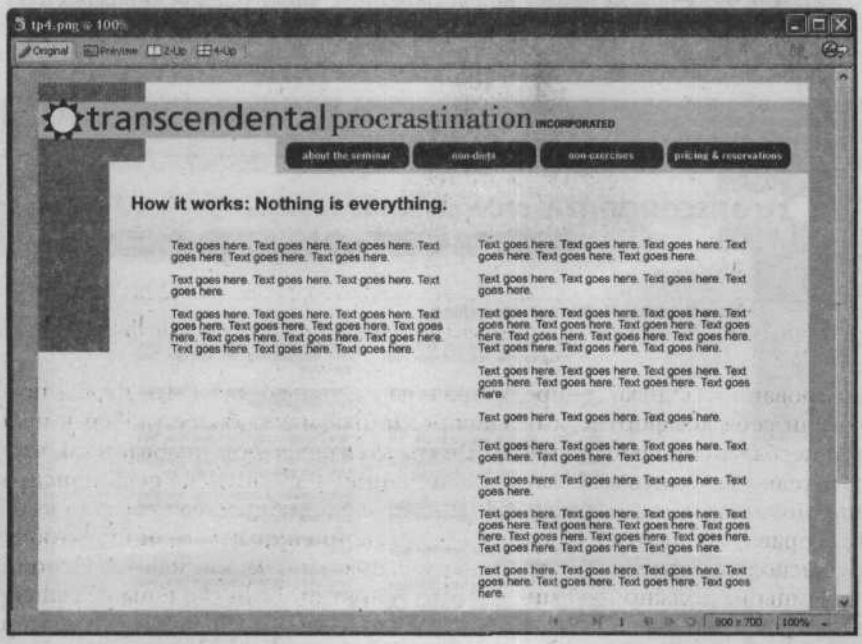


Рис. 4.3. Этот несбалансированный макет не способствует тому, чтобы посетители пролистали страницу

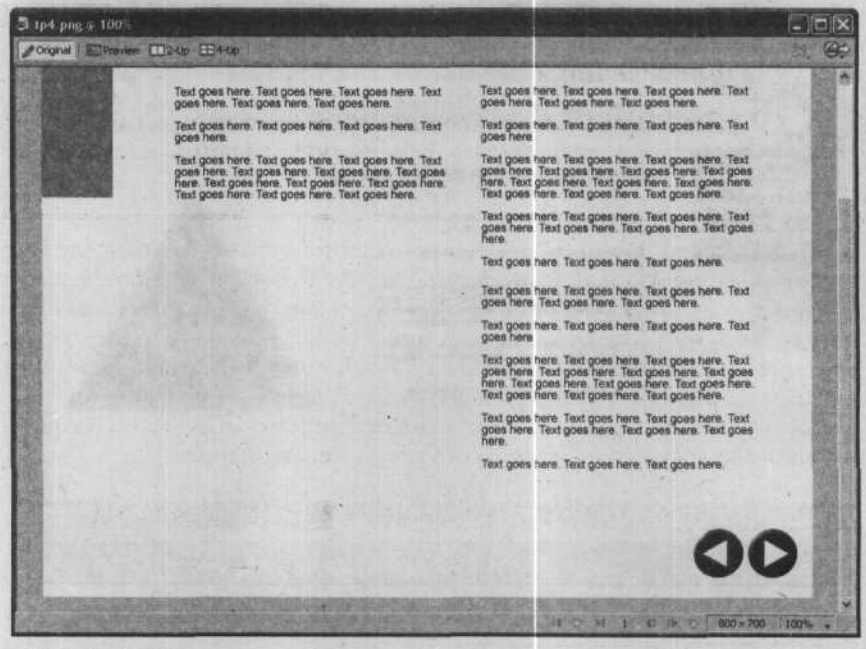


Рис. 4.4. Если они не пролистают страницу, то не увидят «примочки», расположенные внизу страницы

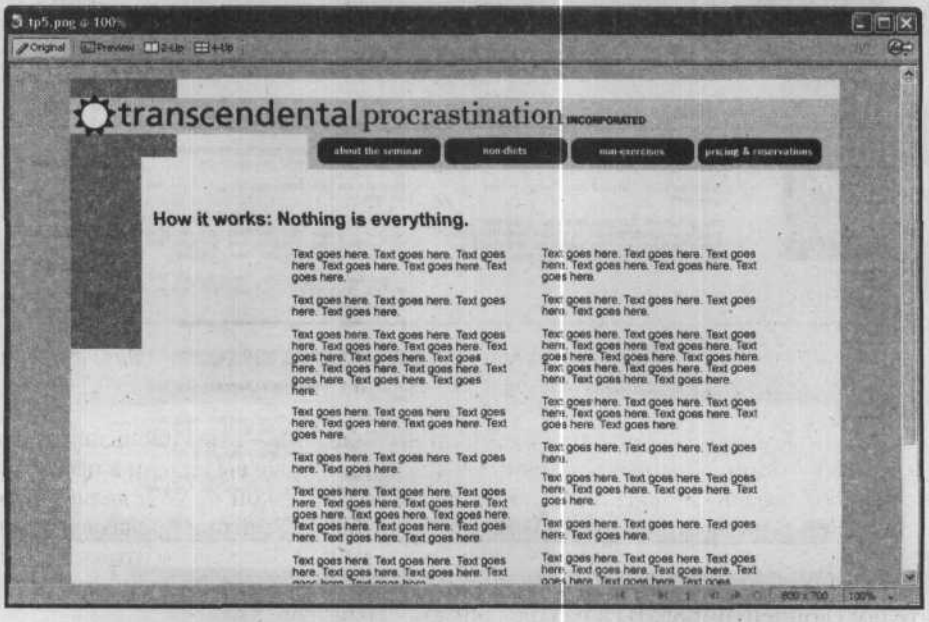


Рис. 4.5. Сбалансированные столбцы помогают укрепить мысль, что страница не заканчивается внизу окна браузера

Определение

Содержимое, находящееся вне экрана, – это информация, которую посетители могут увидеть, только пролистав страницу.

Процесс поиска сбалансированности является немного субъективным, где вы полагаетесь на свой дизайнерский инстинкт, чтобы определить, что работает, а что нет. Один хороший прием заключается в схематическом изображении дизайна с использованием сплошных фигур, как показано на рисунке 4.6. Более абстрактный взгляд на дизайн помогает сфокусироваться на сбалансированности макета, а не на самих элементах дизайна. Если одна часть дизайна кажется тяжелее, чем другая, просто меняйте фигуры местами, пока не добьетесь сбалансированного вида. Затем разместите соответствующие элементы содержимого обратно на своих местах.

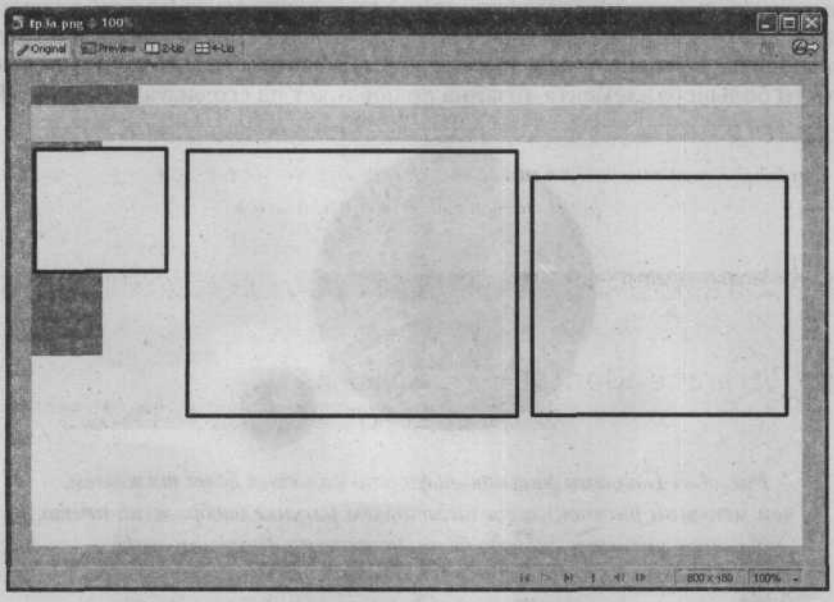


Рис. 4.6. Работа с макетом в абстрактной форме помогает сконцентрироваться на сбалансированности, а не на содержимом

Возможно, самый простой способ сбалансировать дизайн – это использовать почти такие же пропорции для всех элементов страницы, которые вы видели в предыдущих примерах. Это можно назвать классическим подходом. Такой подход является хорошим дизайнерским выбором, если вы желаете, чтобы вид сайта и впечатление от него были сдержанными, серьезными и информативными. Визуально это совершенно не привлекает внимание. Вместо этого дизайн отходит на задний план, позволяя посетителям сконцентрироваться на содержимом страницы.

Определение

Визуальный вес определяет, насколько тяжелым кажется элемент дизайна.

Классический подход не подойдет, если вы желаете, чтобы дизайн был захватывающим. В таком случае необходим более динамичный макет, который делает балансирование более сложным и субъективным процессом. Вам окажет содействие принятие во внимание визуального веса элементов дизайна.

Визуальный вес тяжело измерить, однако он включает в себя размер и форму рассматриваемого элемента. Визуальный вес имеет весьма отдаленное отношение к действительному весу объектов, если они существуют в реальном мире. Как показано на рисунке 4.7, большие элементы дизайна обычно бывают тяжелее, чем меньшие элементы, даже если последние представляют собой что-то, что весит больше в реальном мире. Квадрат кажется более тяжелым, чем прямоугольник, даже когда они имеют одинаковую площадь, как показано на рисунке 4.8. И даже вымывание или уменьшение насыщенности большего элемента дизайна не повлияет на его визуальный вес (Рис.4.9).

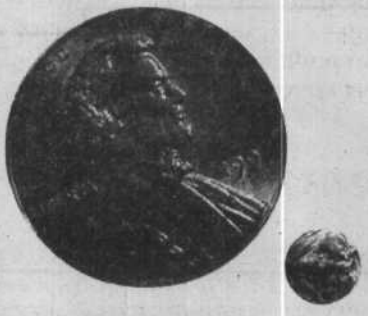


Рис. 4.7. Больший рисунок визуально кажется более тяжелым, чем меньший рисунок, хотя на меньшем рисунке изображено нечто, что является гораздо более тяжелым в реальном мире

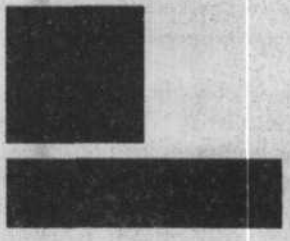


Рис. 4.8. Квадрат и прямоугольник могут иметь одинаковую площадь, но выглядящий массивно квадрат кажется тяжелее

Рис. 4.9. Можно уменьшить насыщенность большего элемента дизайна, но он не потеряет от этого свой визуальный вес

После того как определен визуальный вес используемых элементов дизайна, их можно расположить таким образом, чтобы получить нечто вроде гармонии или равновесия. Мы хотим, чтобы различные веса были сбалансированы, исходя из их положения на странице. Более легкие и более тяжелые элементы кажутся расположенными в беспорядке, как показано на рисунке 4.10. Однако, изменив положение элементов в дизайне, как показано на рисунке 4.11, мы распределили их веса лучше, что дает нам более сбалансированный, гармоничный эффект, в то же время сохраняя смысл движения и ощущение, которое вы получаете при использовании элементов с различными визуальными весами.

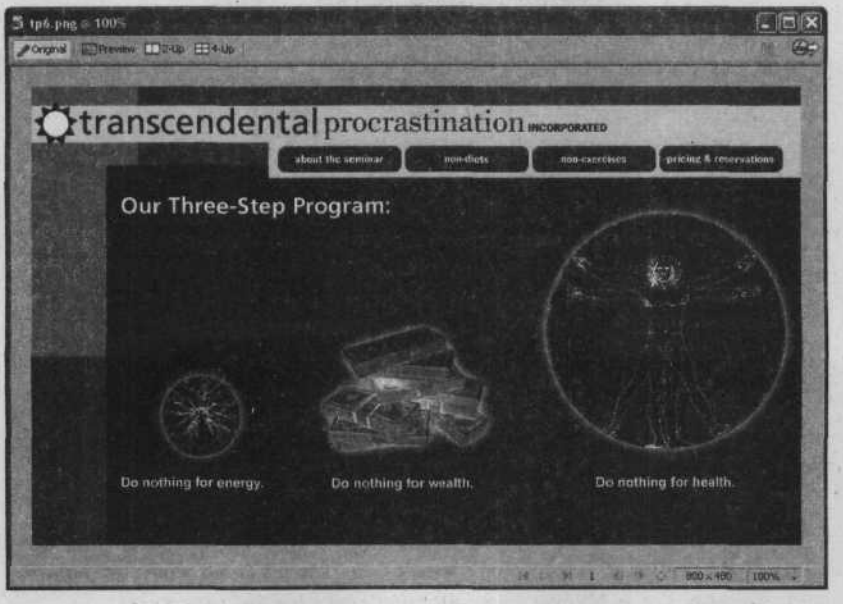


Рис. 4.10. Кажется, что эти элементы расположены в беспорядке

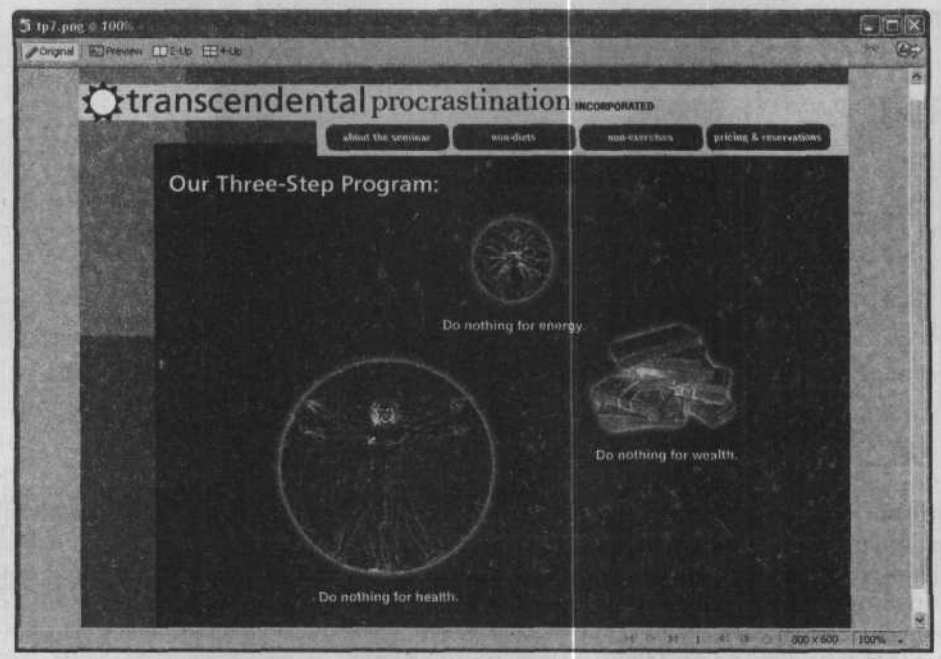


Рис. 4.11. Перераспределение их весов создает большее ощущение равновесия, не жертвуя при этом ощущением динамики

ГЛАВА 5.
Выбор цветов

Выбор эффективной цветовой схемы для Web-сайта является одним из самых больших фокусов в арсенале дизайнера. Каждый элемент дизайна на странице – включая пустое пространство – имеет цвет, и мы хотим убедиться, что все цвета работают сообща для укрепления настроения и назначения сайта.

Совет

Попробуйте следующее: используйте на вашем сайте цвета, большая часть которых принадлежит одному и тому же семейству: теплые и заманчивые, забавные и энергичные или спокойные и прохладные. Затем для одного или двух элементов дизайна, например, для ссылок или символов предупреждения (элементы дизайна, которые действительно должны выделяться), используйте цвет из абсолютно другого семейства цветов.

Работу по выбору цвета можно разбить на три общие области: цвета текста и фона, цвета ссылок и цвета интерфейса.

Выбор цветов текста и фона

Люди не читают текст на Web-странице. У них нет на это времени. Чтение чрезвычайно затягивает процесс. Вместо этого они любят просматривать страницу и выбирать частицы и кусочки информации, которая им интересна. Если хотите, можете винить в этом закат западной цивилизации или слишком большое количество телевизионных программ и видеоигр, но вредные привычки просматривать все бегло остаются фактом. Лучшее, что вы, как дизайнер, можете сделать с текстом на вашем сайте, – выполнить его простым для просмотра.

В этом отношении контраст является наиважнейшим инструментом. Нам требуется мягкое выделение текста на фоне, чтобы блуждающие взгляды могли просматривать, не щурясь и не замедляя скорость, и читать текст слово за словом. Если контраст недостаточен, как показано на рисунке 5.1, вашим посетителям будет тяжелее просматривать текст, что в свою очередь приведет к тому, что им будет тяжелее найти необходимую информацию.

В то же время, не менее важен и тип контраста. Разительный или яркий контраст, похожий на рисунке 5.2, исполняет две роли. Во-первых, он привлекает слишком большое внимание к тексту, который должен отходить на задний план, когда посетители не нуждаются в нем. А во-вторых, заставляет глаза посетителей работать усерднее, чем обычно, что приводит к головным болям и зрительному напряжению. Идеальный контраст между текстом и фоном понятен с точки зрения выделения текста, и спокоен в том смысле, что он не подавляет дизайн.

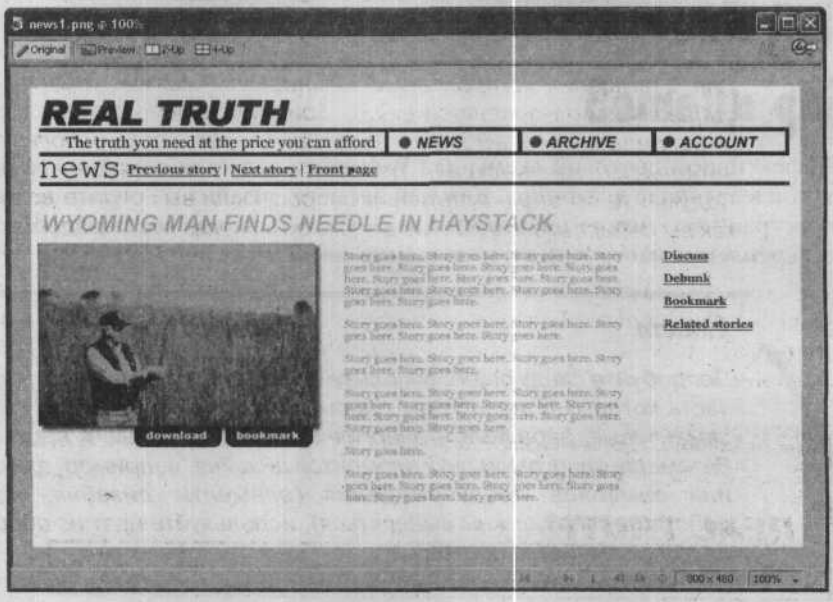


Рис. 5.1. Ваши посетители просматривают текст бегло, а текст с низким контрастом тяжело просматривать бегло

Рис. 5.2. Разительный контраст может быть легким для чтения, но заметьте, как он привлекает к себе слишком много внимания

Совет

Для повышения контраста на странице вы можете подвергнуться соблазну использования жирного шрифта для всего текста. Не стоит этого делать. Жирный шрифт подходит лучше, когда используется в небольших количествах, для выделения или акцентирования. Подумайте об этом, ответив на вопрос: если вы кричите все время, как вы собираетесь увеличить громкость, когда это понадобится?

Вы не ошибетесь, если используете черный текст на белом фоне, как показано на рисунке 5.3. Такое сочетание создает хороший контраст и не трудно для глаз. Если вы можете использовать данную цветовую схему в вашем дизайне, то это вообще будет наилучшим решением.

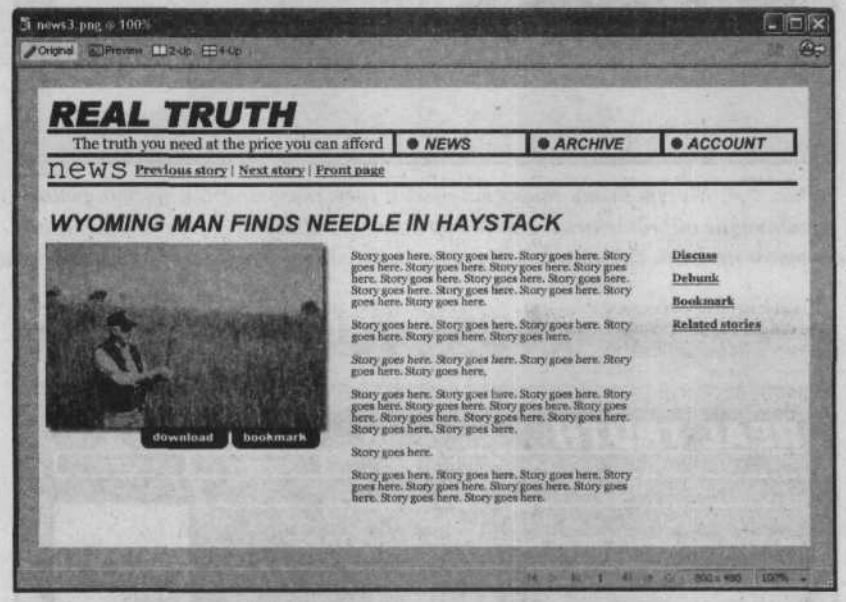


Рис. 5.3. Когда дело доходит до контраста, ничто не поколеблет черный текст на белом фоне

Однако не стоит замыкаться на этой схеме. Существует достаточно веских причин, чтобы испытать различные комбинации цветов. Бледные оттенки или оттенки с уменьшенной насыщенностью могут быть настолько же эффективны, как и белый цвет фона, как показано на рисунке 5.4. Можно поэкспериментировать со светлым текстом на темном фоне, как показано на рисунке 5.5, хотя эта схема, как правило, тяжелее для глаз. Если ваш сайт насыщен текстом, подобная схема может быть не лучшим выбором.

Даже если на вашем сайте мало текста, позаботьтесь о выборе цвета фона страниц. Фон должен оставаться «фоновым» настолько, насколько возможно. Вы же не хотите, чтобы он отвлек внимание, направленное на основное содержимое ваших страниц. Фон – это не место для ярких или необычных оттенков. Оставьте их для подсветок – элементов, которые используются специально для привлечения внимания, – и пусть содержимое страницы станет звездой представления.

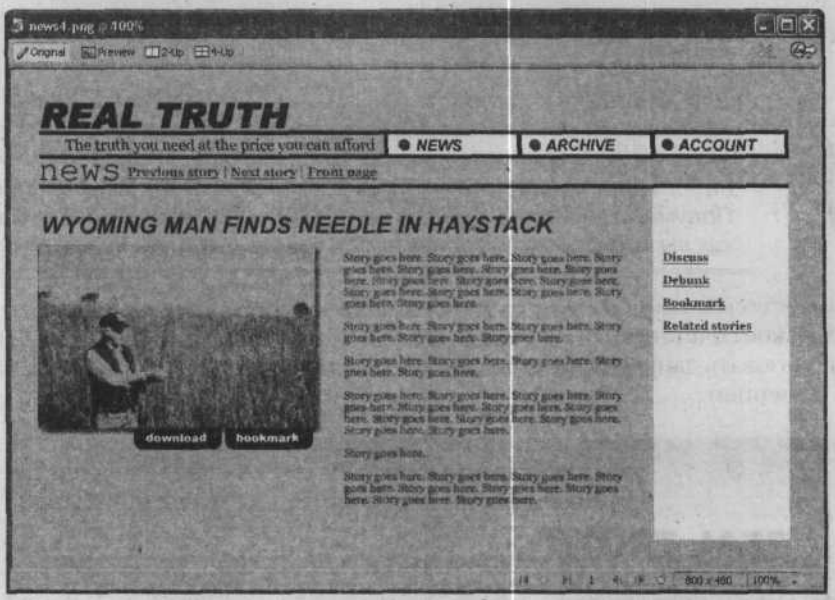


Рис. 5.4. Могут быть эффективными светлые оттенки цвета фона, придающие определенный характер вашему сайту. Обратите внимание, что в этом примере белый цвет отлично подходит в качестве цвета подсветки

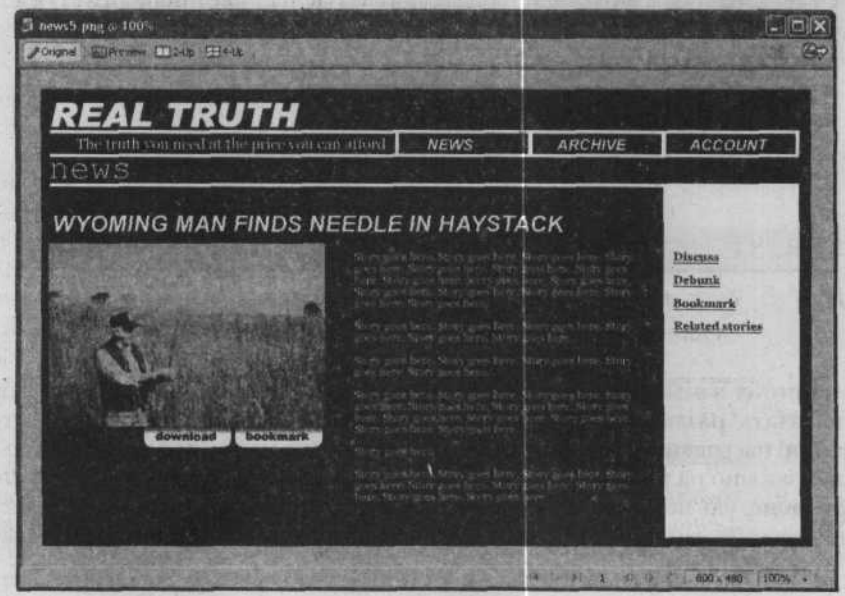


Рис. 5.5. Можно также попробовать светлый текст на темном фоне, однако это будет тяжелее для глаз

Выбор цветов ссылок

С годами появились два стандарта в отношении цветов гиперссылок. Они представлены в таблице 5.1. Изо всех сил старайтесь использовать эти цвета ссылок в вашем Web-дизайне. Посетители уже знают из своего опыта, что синий цвет обозначает непосещенную ссылку, поэтому, если они смогут использовать это знание, когда придут к вам на сайт, вы уберете еще одно препятствие в удобстве использования.

Совет

Приберегите слишком яркие оттенки для меток на графических кнопках или в качестве цветов гиперссылок. Эти элементы должны привлекать внимание посетителей, поэтому они лучше подходят для экзотических или необычных цветов.

Конечно, не всегда целесообразно или желательно использовать стандартные цвета гиперссылок. Возможно, вы захотите привести ссылки в соответствие с цветами компании или цветами, связанными с вашим бизнесом или группой. Возможно, синий и фиолетовый цвета не подходят в общем контексте вашего дизайна. В конце концов, отказаться от стандартов лучше, чем заставить их работать. Если вы отказываетесь от стандартов, отказывайтесь единообразно. Используйте одни и те же цвета для представления одних и тех же элементов, страница за страницей. Девизом вашего дизайна должно стать: «Ожидания оправдались». Это ключ к обучению посетителей тому, как работает ваш сайт. Если они стали ожидать, что красный цвет используется для непосещенных ссылок, используйте красный цвет для непосещенных ссылок абсолютно везде на вашем сайте.

Совет

Вы не обязаны следовать стандартам вплоть до точных шестнадцатеричных кодов цветов. Если рекомендуемый оттенок синего вам не подходит, попробуйте более темный или более светлый оттенок. До тех пор, пока посетители будут считать данный цвет синим, ваш сайт будет получать преимущество от традиционной цветовой ассоциации.

Совет

Как и в случае с большинством остальных жемчужин традиционной мудрости Интернета, стандарты цветов гиперссылок могут неожиданно привести к обратным результатам. Если вы решили не следовать стандартам, то полностью избегайте их. Вы создадите меньше путаницы, если вообще не будете использовать синий цвет.

Табл. 5.1. Стандартные цвета гиперссылок

Цвет	Обозначает	Код цвета (IE)	Код цвета (Netscape)
Синий	Непосещенные ссылки	#0000FF	#0000FF
Фиолетовый	Посещенные ссылки	#990099	#663399

Выбор цветов интерфейса

Часто задаваемые вопросы

Все определяет компания Microsoft?

Почти, но не полностью. В программе Internet Explorer для операционной системы Windows 98 стандартный фиолетовый цвет для посещенных ссылок был переопределен на зеленый. Однако к тому времени фиолетовый цвет уже просочился в коллективное подсознание Интернета в качестве правильного цвета для посещенных ссылок. Программа Internet Explorer для операционной системы Windows XP тихо вернулась обратно к стандартному фиолетовому цвету.

Графическая отделка сайта предоставляет вам наибольшую свободу в выборе цветов. Вы не обязаны столько же думать о вопросах удобства использования цвета. Вместе с тем, здесь задается зрительное ощущение сайта, и цвет играет немаловажную роль в формировании подходящего настроения.

Некоторая работа по цветовому дизайну уже может быть сделана. Если вы разрабатываете сайт для бизнеса или организации, в вашем дизайне должны выразительно использоваться корпоративные цвета. Корпоративные цвета – это лишь цвета, на которые департамент по маркетингу затратил много времени и ресурсов, чтобы ассоциировать их с организацией. Вспомните красные дуги компании Coca-Cola или золотые дуги компании McDonald's. Догадайтесь, какие цвета изобилуют на соответствующих корпоративных Web-сайтах. Использование корпоративных цветов – это способ **брендинга** или, другими словами, приведения сайта к общему корпоративному виду организации. Как только вы захотите использовать в дизайне логотип группы, вы также захотите включить другие признаки фирмы, например, корпоративные цвета.

Определение

Брендинг – это метод приведения сайта к общему корпоративному виду организации, используя элементы из маркетингового материала, например, логотипы и корпоративные цвета, в дизайне.

Часто задаваемые вопросы

Если я поработаю с отделом маркетинга, буду ли я уважать себя утром?

Работа с отделом маркетинга часто оставляет неприятное впечатление. Но просто подумайте об этом следующим образом. Бюджет на рекламу вашего сайта, возможно, равен нулю. Однако каждый доллар, потраченный на продвижение торговой марки компании в других источниках, например, по телевидению, можно считать долларом, потраченным на продвижение вашего сайта – пока вы эффективно используете торговую марку на сайте. Они тратят деньги, а вы получаете выгоду. Кто сказал, что отдел Web-разработок не смеется последним?

У цвета также есть психологическое измерение, которое, по крайней мере, стоит учесть при планировании подхода. Вы не задавались вопросом, почему у множества Web-сайтов в основном интерфейсе присутствует синий цвет? Взглянув на таблицу 5.2, вы увидите, что у людей синий цвет ассоциируется со стабильностью и доверием – два отличных качества для продвижения, если вы желаете удержать своих посетителей, особенно, если разрабатывается сайт для финансового или медицинского сектора.

Табл. 5.2. Общие психологические цветовые ассоциации

Цвет	Ассоциация (западная культура)	Примеры
Красный	Страсть, сила, жара, жестокость, предупреждение	Пожарная машина, стоп-сигнал, валентинка
Оранжевый	Тепло, пища	Камин, интерьер закусочной
Желтый	Игривость, счастье	Солнечный свет, цветы
Зеленый	Природа, рост, успокоение, изобилие; также ревность, болезнь	Леса, деньги; также плесень, тина
Синий	Доверие, надежность, стабильность, преданность	Униформа полицейских и военных
Фиолетовый	Роскошь, величие, исключительность	Королевские одеяния
Коричневый	Устойчивость, практичность	Земля, скалы
Белый	Чистота, невинность; также холодность, отчужденность, суровость	Машины скорой помощи, униформа медицинских работников, свадебные платья; также снег, мрамор, гипс
Черный	Формальность, власть, уныние; также мрачность, печаль	Смокинги, солнцезащитные очки; также смерть (старуха с косой)

К счастью, дизайн совсем не имеет отношения к науке. Невозможно просто превратить цвета в предугадываемые эмоциональные отклики, точно так же, как нельзя одеть сайт с неудачным дизайном в синие цвета и надеяться, что это будет способствовать атмосфере доверия. Цвет должен быть одной стороной общей стратегии для передачи настроения и назначения сайта. Если ваша цель – предоставить информацию, дополните консервативный дизайн успокаивающими цветами. Если ваша цель – развлечения, попробуйте использовать более динамичные дизайны и рискните, применив энергичные цвета.

Часто задаваемые вопросы

Все ли культуры имеют одинаковые ассоциации?

Нет. Цветовые ассоциации не универсальны. На самом деле, многие культуры имеют совершенно отличные интерпретации цветов. Трудно предположить, что невеста на Западе наденет красное свадебное платье – цвета пожарной машины – и тем самым не вызовет насмешек. Тем не менее, красный цвет является цветом непорочности в Индии и Китае. И в то время как зеленый цвет на Западе может обозначать богатство, это очень неудачный цвет на Востоке, где он обозначает супружескую измену. Если вы разрабатываете сайт для посетителей из разных стран, в достаточном объеме изучите их культуру.

ГЛАВА 6.
Разработка специальных возможностей

Вы являетесь дизайнером. Интернет же является такой графической средой, где легко забыть, что не каждый человек бродит по Интернету, используя свои глаза. Ведь Интернет – это мировое сообщество всех людей, включая людей с ограниченными возможностями. Инвалиды по зрению тоже желают посещать ваш сайт. То же хотят делать люди с когнитивными расстройствами или с затруднениями при обучении. Если бы эти люди появились перед вашей дверью, вы бы не захотели оскорбить их или сделать их визит неприятным. Точно также при разработке Web-сайта вы должны учитывать их особые ситуации.

Разработка специальных возможностей заключается в том, чтобы сделать содержимое Web-сайта доступным для каждого, независимо от инвалидности. Интернет не единственное место, где важны специальные возможности. На самом деле, Интернет является одним из последних мест, где рассматриваются вопросы специальных возможностей. Общественные здания должны иметь пандусы и лифты, чтобы обеспечить легкий доступ людям с физическими недостатками. Автомобильные стоянки выделяют лучшие места для инвалидов. Кинотеатры предоставляют специальные места и ряды, более удобные для инвалидов-колясочников. Телевизионные программы выходят с субтитрами для тех, кто плохо слышит. Только в конце 1990-х годов люди, работающие с компьютерами, начали думать так же. Теперь в Соединенных Штатах и многих других странах специальные возможности являются вопросом права. Правительству США запрещено давать подряды исполнителям – в том числе, разработчикам Web-сайтов – чьи продукты не отвечают нормам по специальным возможностям. Никто не собирается тащить вас в федеральную тюрьму, если вы не создали Web-сайт со специальными возможностями. Вы просто не сможете выгодно устроиться и получать от американских налогоплательщиков семь тысяч долларов за ссылку.

Определение

*Web-сайт обладает **специальными возможностями**, если его содержимое доступно для всех, включая людей с ограниченными возможностями.*

Простого упоминания словосочетания «специальные возможности» в обычном разговоре достаточно, чтобы причинить страдание некоторым дизайнерам, однако наделить Web-сайт специальными возможностями на самом деле не так уж и сложно. Вы не обязаны разрабатывать способ направления изображений прямо в мозг инвалида по зрению. Вы не обязаны пытаться передать каждому одинаковые впечатления от сайта. Вы просто должны предоставить эквивалентное впечатление, т.е. предоставить информацию на сайте таким образом, чтобы люди с инвалидностью имели тот же уровень доступности, что и нормальные люди.

Часто задаваемые вопросы

Где я могу узнать больше о законе по специальным возможностям в Соединенных Штатах?

*Текст закона Соединенных Штатов по специальным возможностям вы можете найти в 1998 поправке раздела 508 Закона о реабилитации (**www.section508.gov**).*

Ваши нетрудоспособные посетители даже пойдут на компромисс. У них на компьютерах установлено специальное программное обеспечение и оборудование, например, читатели экрана, преобразователи текст-речь и преобразователи текст-шрифт Брайля. Все, что вам необходимо сделать, – представить содержимое таким образом, чтобы эти устройства смогли его найти и работать с ним.

Определение

Текстовые эквиваленты – это текстовые описания нетекстового содержимого для использования читателями экрана или другими специальными средствами.

Обычный текст является исходным материалом, который используют инструменты специальных возможностей для поиска и преобразований. Поэтому, создавая сайт со специальными возможностями, желательно присвоить соответствующий текст каждому нетекстовому кусочку содержимого вашего сайта. Это и есть **текстовые эквиваленты**. Идея заключается в том, что если вы можете описать словами части вашего сайта, которые только визуальны (например, изображения), вы предоставите людям с инвалидностью эквивалентное впечатление о содержимом вашего сайта. Да, это разные впечатления, но все в порядке. Нетрудоспособные люди должны получить ту же информацию, что и другие люди, независимо от того, увидели они ее на картинке или прослушали текстовое описание этой картинки.

Обеспечение доступности изображений

Возможно, вы уже знаете, как добавить текстовые эквиваленты к изображениям, не подозревая об этом. Все, что для этого необходимо – атрибут `alt` тега `img`. Этот атрибут определяет текст всплывающей подсказки, которая появляется в окне браузера при наведении указателя мыши на изображение. Средства специальных возможностей используют атрибут `alt` для описания изображения инвалидам по зрению.

Итак, чтобы сделать изображение доступным, просто задайте ему подробное дублирующее описание, наподобие следующего:

```
<img src="saturn.jpg" alt="Фотография планеты Сатурн, полученная с
космического зонда Voyager, которая показывает гигантские кольца и
оранжевые полосы облаков планеты.">
```

Не экономьте на тексте, как в данном случае:

```
<img src="saturn.jpg" alt="Сатурн">
```

Такой тип значения атрибута `alt` полезен только тем людям, которые в нем не нуждаются. Помните, что вы хотите передать своим нетрудоспособным посетителям эквивалентное ощущение, поэтому слова должны быть настолько точными, насколько это возможно, чтобы выразить сущность изображения. Представьте, что вы вернулись на урок литературы и пишете сочинение. Предоставьте своим читателям достаточно информации, чтобы они смогли сформировать в уме четкую картину того, что вы описываете.

Для изображений со ссылками попытайтесь вставить в текст атрибута `alt` описание того, что произойдет при щелчке на изображении, как в следующем примере:

```
<img src="aboutus.jpg" alt="Переход в раздел сайта "О нас".">
```

или так:

```
<img src="macromedia.jpg" alt="Переход на сайт macromedia.com">
```

Если в тексте атрибута `alt` будет стоять просто «О нас» или «Macromedia», инвалиды по зрению, посещающие ваш сайт, не будут иметь понятия о том, для чего нужно и что делает изображение. Используя слово для описания результата, вы воспроизводите ненужный видимый контекст. Смысл или назначение изображения выясняется из отчетливости и понятности одного описания.

Часто задаваемые вопросы

Как насчет изображений, содержащих текст? Этот текст доступен?

Как таковой, нет. Иногда изображение содержит встроенный текст; например, метка на кнопке или легенда на диаграмме. Однако читатели экрана не воспринимают буквы и слова на изображении, поэтому они не могут воспроизвести этот текст в более доступном виде. Добавляйте текст для атрибута `alt` и для таких изображений.

Еще одно табу по специальным возможностям, касающееся изображений: не разрабатывайте мерцающие, мигающие, трясущиеся, со стробоскопическим эффектом и подобные раздражающие изображения, похожие на некоторые из наименее любимых рекламных баннеров. Делать этого не стоит не только потому, что эти изображения безобразны с эстетической точки зрения, психологически оскорбительны и тупы концептуально, но и потому, что они также способствуют возникновению зрительного напряжения и головных болей у людей с чувствительным зрением и могут привести к приступам у людей, страдающих эпилепсией. Если вы настолько безрассудны в оформлении рекламного объявления, что готовы создать физические трудности для некоторых ваших посетителей, возможно, вам бы больше подошла работа в отделе рекламы.

Обеспечение доступности мультимедиа

Обеспечить доступность мультимедиа, наподобие Flash-роликов, звуковых файлов и клипов QuickTime, тяжелее, поскольку мы работаем не просто с одним статическим изображением. По этой причине мы не можем просто обобщить информацию мультимедиа-файла простым текстовым описанием. Более того, средства специальных возможностей обычно не могут извлечь текст, находящийся внутри мультимедиа файла, для преобразования в произносимые слова или в шрифт Брайля.

К сожалению, не существует простого и быстрого способа сделать мультимедийные данные доступным. Консорциум World Wide Web, или W3C, предлагает несколько решений, однако все они являются трудоемкими и имеют громоздкий и трудный для понимания вид. Так, в соответствии с рекомендацией консорциума W3C, мультимедиа-презентация должна содержать синхронизированную аудиодорожку, комментирующую действие, происходящее на экране, что очень похоже на прямую радиотрансляцию бейсбольного матча, вместе с полными субтитрами. В качестве альтернативы можно представить содержимое мультимедиа в виде простого текста или создать ряд страниц, которые бы предоставляли ту же информацию в более доступном формате.

Правда заключается в том, что большинство мультимедиа-файлов в Интернете не обеспечивает специальных возможностей вообще. Чтобы сделать существующую презентацию доступной людям с ограниченными возможностями, зачастую приходится начинать все сначала и разрабатывать ее заново. Помимо этого, возможно, вам придется приобрести новое оборудование, например, качественные микрофоны и программное обеспечение для редактирования звука. Конечно же, все это дополнительное содержимое внутри мультимедиа-файла может увеличить его размер, что в свою очередь сделает его менее удобным для загрузки и использования вашими посетителями. По этим причинам многие Web-сайты, сознательно работающие над доступностью, предпочитают вообще отказываться от мультимедиа.

Часто задаваемые вопросы

Где я могу узнать больше о рекомендациях по специальным возможностям?

Всестороннее обсуждение рекомендаций по специальным возможностям можно найти в Интернете по адресу ***www.w3.orf/WAI****.*

Компании по разработке программного обеспечения работают над проблемой. Компания Macromedia, в частности, отличается своим инструментальным средством Flash MX, которое используют почти все для разработки Flash-роликов. Программа Flash MX позволяет вставлять текстовые эквиваленты, клавишные сокращения и другие средства повышения доступности в содержимое Flash-ролика. Даже с большой натяжкой это нельзя назвать волшебными решениями, однако это конкретные шаги в правильном направлении. Принимая во внимание, что технология Flash является самым популярным мультимедиа-форматом на планете, любой тип прогресса обнадеживает. Однако важно следущее: ни один из мультимедиа форматов не является доступным на 100%, и не слушайте тех, кто будет утверждать обратное, особенно людей, использующих формат Flash. Если вы желаете использовать мультимедиа на вашем сайте, ознакомьтесь с рекомендациями консорциума W3C, и будьте готовы затратить дополнительные усилия для их правильной реализации.

Управление использованием цвета

Цвет является бесценным дизайнерским инструментом. Мы используем его для привлечения внимания к определенным элементам страницы. Мы используем его, чтобы проводить различия между одним типом содержимого и другим. Однако цвет является таким же визуальным элементом, как и изображение, поэтому, как разработчику Web-сайтов, думающему о доступности, вам необходимо быть осторожным при его использовании. В конце концов, многие ваши посетители имеют нарушение цветоощущения.

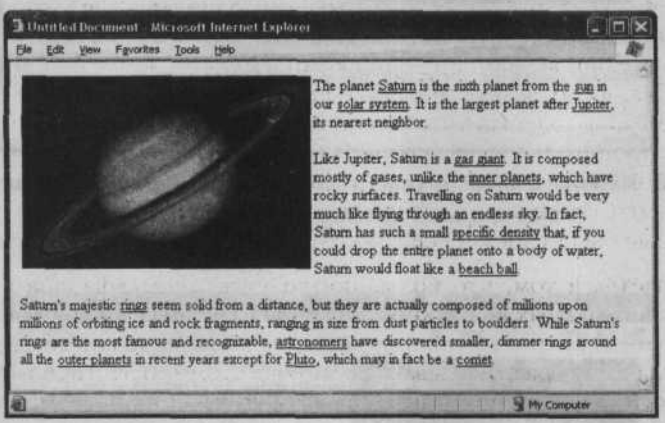


Рис. 6.1. Подчеркивание помогает привлечь внимание к гиперссылкам в текстовом блоке

Совет

Попробуйте рассмотреть дизайн сайта в черно-белых цветах. Если определенные элементы исчезают без использования цвета или если другие элементы теряют необходимое акцентирование, возможно, вам нужно заново пересмотреть вашу стратегию.

Рассмотрим пример на рисунке 6.1. Обратите внимание, как подчеркивание помогает определить ссылки в этом текстовом блоке. Однако если убрать подчеркивание, как показано на рисунке 6.2, вы не сможете сказать, где заканчивается текст и начинается гиперссылка.

Таким образом, хотя цвет и является эффективной визуальной подсказкой или индикатором для гиперссылки или другого элемента, он не должен быть единственной предоставляемой визуальной подсказкой. Конечно, вы можете использовать цвет для привлечения внимания к элементам страницы. Просто убедитесь, что вы дублируете его. Используйте подчеркивание, толстую рамку, символ стрелки, жирный шрифт, курсив – все, что переносится в черно-белую среду, как показано на рисунке 6.3.

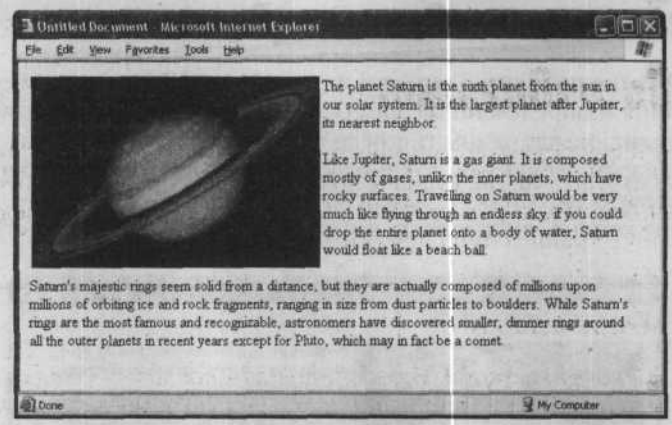


Рис. 6.2. Уберите подчеркивание, и тот, кто не может различать цвета, будет иметь проблемы с поиском ссылок

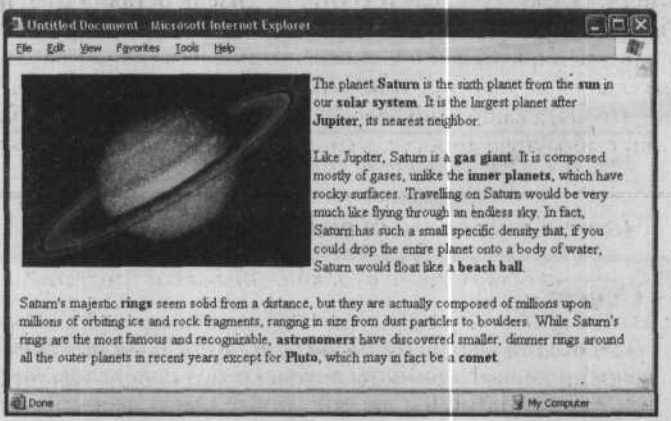


Рис. 6.3. Чтобы заставить работать ссылки без использования цвета, их не обязательно подчеркивать. Жирный шрифт подходит так же хорошо, как и любые другие визуальные подсказки, не зависящие от цвета

ГЛАВА 7.
Удобство навигации

У народных и детских сказок можно многому научиться. Безусловно, они могут показаться детскими игрушками, но, если вы познакомитесь с ними внимательно, то обнаружите, что они берутся за решение важных вопросов – зло, справедливость, храбрость, Web-дизайн – крайне символическими способами.

Определение

Строка навигации (Breadcrumb trail – дословно «след из хлебных крошек») – это элемент навигации, который помогает посетителям определить, где они находятся в иерархии вашего сайта, показывающий связь текущей страницы с остальной окружающей структурой.

Возьмем, к примеру, сказку о Гензель и Гретель. Гензель оставил след из хлебных крошек, чтобы он и его сестра Гретель смогли найти дорогу из леса, где злая мачеха попыталась их оставить. На первый взгляд, эта сказка кажется просто еще одой диатрибой (резкой обличительной речью) против злых мачех, однако попробуйте представить, что злая мачеха – это вы, а ваши многострадальные посетители – смелые Гензель и Гретель, которые ищут обратную дорогу из запутанного леса вашего Web-сайта.

Часто задаваемые вопросы

О чем на самом деле сказка о Гензеле и Гретели?

В сказке «Гензель и Гретель» Братьев Гримм, злая мачеха дважды пытается оставить своих детей в лесу. Первый раз Гензель оставляет след из белых камешков. Второй раз он проделывает то же самое с хлебными крошками, однако это не оказывается столь эффективным. Птицы, живущие в лесу, склевывают крошки до того, как он и Гретель получают шанс воспользоваться ими. Поэтому, хотя на языке Интернета мы называем это следом из хлебных крошек, на самом деле подразумевается след из белых камешков.

Никто не хочет быть злой мачехой в чьей-либо сказке. Все злые мачехи умирают в страшных муках. Гензель и Гретель умирали с голода. Но успокойтесь, что сказал и Гензель своей сестре. Чтобы ваш Web-сайт избежал доли злой мачехи, все, что необходимо сделать, – это оставить след из хлебных крошек, по которому могли бы следовать Гензели и Гретели.

Строка навигации – это элемент навигации, который помогает посетителям определить, где они находятся в иерархии вашего сайта, показывающий связь текущей страницы с остальной окружающей структурой. Это что-то вроде миниатюрной карты сайта для конкретного раздела.

Это одна из тех концепций, которые приобретают больший смысл, когда вы увидите ее. Предположим, что ветвь вашего сайта о Европейском фольклоре выглядит следующим образом:

Детские сказки

- Братья Гримм
  - Гензель и Гретель
  - Рапунцель
  - Белоснежка и Краснозорька
  - Эльфы и сапожник
- Ганс Христиан Андерсен
  - Гадкий утенок
  - Дюймовочка
  - Новое платье императора
  - Девочка со спичками
- Шарль Перро
  - Красная шапочка
  - Золушка
  - Спящая красавица
  - Синяя борода

Строка навигации на странице Братьев Гримм будет выглядеть так:

Главная страница > Детские сказки > Братьев Гримм.

Переместившись на страницу сказки Рапунцель, вы увидите:

Главная страница > Детские сказки > Братьев Гримм > Рапунцель

Однако на странице сказки «Золушка» Шарля Перро строка навигации будет выглядеть так:

Главная страница > Детские сказки > Шарль Перро > Золушка

Обратите внимание, что строка навигации не пытается отразить структуру сайта целиком. Она просто показывает посетителям логический путь к текущей странице – и, что более важно, обратный путь к начальной странице.

Совет

Подобно тому, как Гензель начал бросать хлебные крошки, выйдя из дверей дома, всегда начинайте строку навигации вашего сайта со ссылки на первую страницу.

Обычно все элементы строки навигации являются ссылками, кроме последнего элемента в списке, представляющего текущую страницу.

Строки навигации чрезвычайно полезны, особенно если сайт имеет сложную структуру с множеством уровней страниц. Эти строки помогают посетителям найти обходной путь и предоставляют быструю информацию вроде «Вы находитесь здесь» людям, которые пришли на сайт с сервера поиска.

Совет

Если глубина структуры вашего сайта не превышает, скажем, два уровня страниц (основная категория и подкатегория), нет необходимости использовать строки навигации.

Безусловно, строка навигации будет полезна настолько, насколько вы сделаете ее такой. Попробуйте расположить ее в верхней части страницы, там, где посетители будут хорошо видеть ее, как показано на рисунке 7.1. Вдобавок к этому, хорошей идеей является отделение строки навигации от других типов текстовой навигации. Если строка навигации будет иметь одинаковый визуальный стиль, как показано на рисунке 7.2, посетители могут не понять, что к чему. Соглашение для строк навигации заключается в использовании заглавных и строчных букв, вместо всех заглавных, а для разделения элементов использовать знак «больше», вместо знака окантовки, как показано на рисунке 7.3. Конечно же, вы можете экспериментировать с другими видами строки навигации, но имейте в виду, что ваши посетители, возможно, уже привыкли к такому.

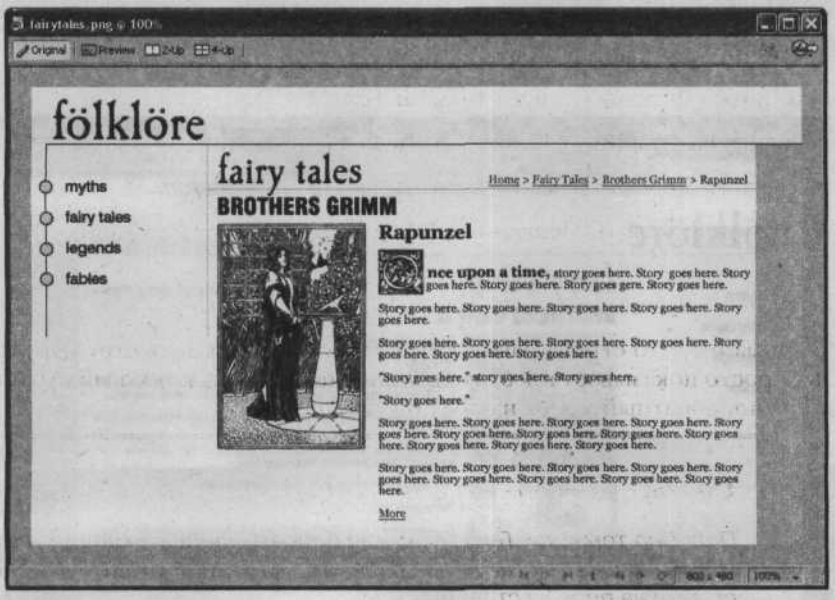
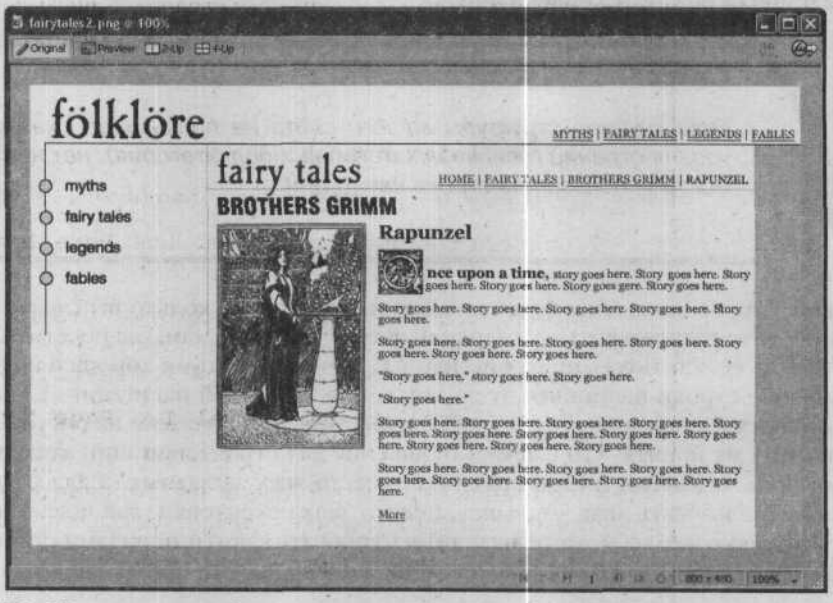


Рис. 7.1. Старайтесь размещать строку навигации вверху страницы

В конце концов, строки навигации отлично работают в качестве альтернативного метода навигации. Все хорошие сайты имеют, по крайней мере, два метода. Не пытайтесь строить основную схему навигации по вашему сайту с использованием строк навигаций, иначе вы получите множество разочарованных посетителей, которые

ищут ссылку на страницу Ганса Христиана Андерсена в области страницы Братьев Гримм. Однако если вы добавите типичную схему с панелью и строкой навигации, то существенно упростите передвижение для посетителей.


fölklöre
MYTHS | FAIRY TALES | LEGENDS | FABLES
myths
fairy tales
legends
fables
fairy tales
HOME | FAIRY TALES | BROTHERS GRIMM | RAPUNZEL
BROTHERS GRIMM
Rapunzel
nce upon a time, story goes here. Story goes here. Story goes here.

Инструментарий

Строка навигации

Добавьте этот фрагмент кода в вашу страницу, чтобы создать связанную строку навигации. Просто замените заполнители правильными путями и названиями страниц вашего сайта. Если вам нужно меньше, чем три уровня страниц, удалите соответствующие строки кода.

```
<p>
  <a href="index.htm">Home</a> &gt;
<!– &gt; - это код для знака "больше". >
  <a href="levelone.htm">Level One Page Name</a>
&gt;
<!– Скопируйте и вставьте следующую строку
необходимое число раз, если в строке навигации
вам необходимо больше, чем три уровня страниц. >
  <a href="leveltwo.htm">Level Two Page Name</a>
&gt;
<!– Последний элемент списка не получает ссылку,
поскольку он представляет текущую страницу. >
  Level Three Page Name
</p>
```

ГЛАВА 8.
Использование меню переходов

Меню переходов – это удобный для использования элемент навигации, если у вас есть большой сайт с множеством страниц. По существу, это быстрый указатель на наиболее популярные страницы вашего сайта, как показано на рисунке 8.1. Посетители выбирают элемент из списка, а простая функция на языке JavaScript загружает определенную страницу. Таким образом, посетителям не придется перемещаться по сайту шаг за шагом.

Определение

Меню переходов – это быстрый указатель на наиболее популярные страницы сайта.

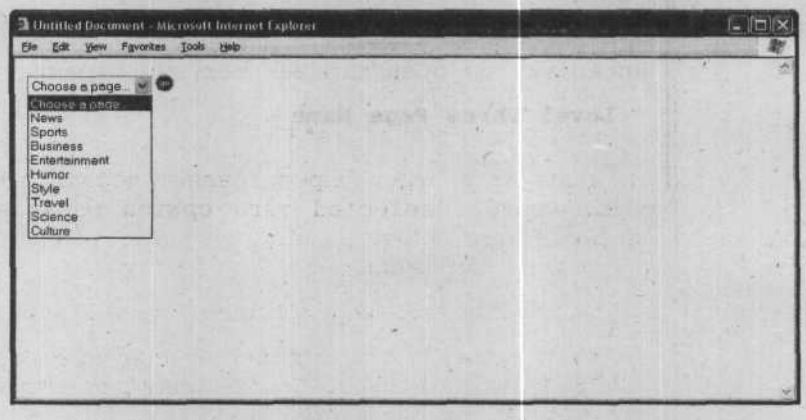


Рис. 8.1. Используйте подобное меню переходов в качестве указателя на наиболее популярные страницы вашего сайта. Посетитель выбирает страницу в меню, и выбранная страница загружается в окне браузера

Для построения меню переходов потребуется два компонента: HTML-форма (на ней размещаются различные элементы, на которых щелкает посетитель), иначе называемая **пользовательским интерфейсом** (Front end)), и функция на языке JavaScript (наспех написанная программа, которая заставляет работать меню переходов, также называемая **прикладной частью** (Back end)). В инструментарии, приведенном в данной главе, демонстрируется, как создавать оба компонента.

Совет

Меню переходов – это приятная дополнительная возможность для сайта, однако не доверяйте ей всю вашу навигацию. Не существует замены для доброй, старомодной панели навигации.

Что касается меню переходов, то существует два подхода. Первый подход заключается в использовании кнопки **Go** (Перейти) или чего-то еще, на чем посетитель может щелкнуть после выбора страницы в меню. Ничего не произойдет, пока посетитель не щелкнет на кнопке. Альтернативным подходом является использование самоактивирующегося меню переходов. Это означает, что как только посетитель выбрал страницу, меню переходов автоматически выполняет соответствующий переход.

Какой подход лучше? Трудно сказать. Если использовать кнопку **Go** (Перейти), вероятность совершения ошибки посетилелями уменьшается. Если они решат, что не хотят использовать меню переходов после того, как открыли его (на радость отдела рекламы), или если они захотят открыть другую страницу после осуществленного выбора, кнопка **Go** (Перейти) предоставит им возможность выбрать другую страницу перед переходом. С другой стороны, для кнопки **Go** (Перейти) требуется дополнительный щелчок, что замедляет посетителей. Для самоактивирующегося меню переходов необходим всего один щелчок, поэтому оно быстрее. Вы можете испытать оба типа меню переходов на вашем сайте, чтобы увидеть, какой подходит лучше, но, если сомневаетесь, сделайте некую защиту от дурака и используйте кнопку **Go** (Перейти).

Инструментарий

Меню переходов с кнопкой Go – пользовательский интерфейс

Этот фрагмент кода на языке HTML реализует пользовательский интерфейс вашего меню переходов.

```
<form name="jumpmenu">
   <select name="pages">
<!– Следующая строка представляет первый элемент в меню. Атрибут selected тега option означает, что этот элемент появится по умолчанию в меню, когда загрузится страница. –>
      <option selected>Choose a page…</option>
<!– Следующая строка вставляет название раздела в список элементов меню. При желании эту строку можно удалить. –>
      <option>First category</option>
<!– Следующими идут элементы меню переходов. В качестве значения атрибута value укажите путь к странице, на которую вы желаете осуществить переход, например, aboutus/index.htm или ../products/brochure.htm. Путь, как и гиперссылка, может быть абсолютным, относительно документа или относительно корня. –>
      <option value="firstpath">First page name
</option>
      <option value="secondpath">Second page name
</option>
      <option value="thirdpath">Third page name
</option>
```

```
<!- Повторите строки, наподобие представленных выше, столько раз, сколько элементов меню вы хотите видеть в данной части списка. ->
      <option>Second category</option>
<!- Здесь идут дополнительные элементы меню переходов. Повторите эту строку столько раз, сколько элементов меню вам необходимо, после чего закройте тег select. ->
      <option value="fourthpath">Fourth page name</option>
   </select>S
<!- Следующая строка добавляет кнопку на форму. ->
   <input type="button" name="go" value="Go" onClick="doJumpMenu();">
<!- Теперь закройте тег form и все готово. ->
</form>
```

Инструментарий

Самоактивирующееся меню переходов – пользовательский интерфейс

Используйте этот фрагмент кода на языке HTML для размещения самоактивирующегося меню переходов на вашей странице.

```
<form name="jumpmenu">
<!- Следующая строка добавляет объект выбора, известный также как открывающийся список, на форму, и дает указание следить за событием onChange языка JavaScript. Когда посетитель изменит значение формы, запустится функция перехода. ->
   <select name="pages" onChange="doJumpMenu();">
<!- Остальная часть формы выглядит и работает так же, как и предыдущая форма. ->
      <option selected>Choose a page…</option>
      <option>First category</option>
      <option value="firstpath">First page name</option>
      <option value="secondpath">Second page name</option>
      <option value="thirdpath">Third page name</option>
      <option>Second category</option>
```

```
    <option value="fourthpath">Fourth page
name</option>
<!- Добавьте необходимое количество разделов и
элементов перехода. ->
  </select>
</form>
```

Инструментарий

Меню переходов – прикладная часть

Добавьте эту функцию на языке JavaScript на ваш сайт, чтобы заработало меню переходов. Данная функция одинакова как для меню переходов с кнопкой **Go** (Перейти), так и для самоактивирующихся меню переходов.

```
<script language="JavaScript">
/* Используйте теги script только в том случае, если
вставляете эту функцию внутрь HTML-страницы.
Опустите теги script, если вы добавляете эту функцию
во внешний JavaScript-файл. */
  function doJumpMenu() {
    var menu = document.jumpmenu.pages;
/* Следующая строка получает значение атрибута
value для выбранного элемента меню. */
    var menuValue = menu.options[menu.
  selectedIndex].value;
/* Следующая конструкция if/then выполняет переход
на выбранную страницу, если значение атрибута value
не было пустым. */
    if (menuValue != "") {
      location.href = menuValue;
    }
  }
</script>
```

Инструментарий

Меню переходов с графической кнопкой Go (Перейти) – Пользовательский интерфейс

Если вы желаете использовать графическую кнопку **Go** (Перейти) вместо стандартной серой кнопки языка HTML, все, что необходимо, – это разработать кнопку в любимой программе для работы с графикой. Затем добавьте следующий фрагмент кода на языке HTML на свою страницу, чтобы создать меню:

```
<form name="jumpmenu" onSubmit="return doJumpMenu
GraphicButton();">
<!– Тег form, расположенный выше, ожидает события
onSubmit, чтобы выполнить скрипт меню переходов. –>
<!– Ниже представлено меню. Оно работает также, как
и все остальные. Добавьте желаемое количество раз-
делов и элементов меню. –>
   <select name="pages">
      <option selected>Choose a page…</option>
      <option>First category</option>
      <option value="firstpath">First page name
   </option>
      <option value="secondpath">Second page name
   </option>
      <option value="thirdpath">Third page name
   </option>
      <option>Second category</option>
      <option value="fourthpath">Fourth page name
   </option>
   </select>
<!– Здесь добавляется графическая кнопка Go (Перей-
ти). –>
   <input type="image" name="go" src="imagepath"
width="imagewidth" height="imageheight" bor-
der="0">
</form>
```

Инструментарий

Меню переходов с графической кнопкой Go (Перейти) – прикладная часть

Для меню переходов с графической кнопкой **Go** (Перейти) необходима немного отличающаяся функция на языке JavaScript. Web-браузеры, например, Internet Explorer и Netscape, рассматривают кнопки-изображения в качестве кнопок отправки данных – когда посетитель щелкает на такой кнопке, браузер пытается отправить форму, что нежелательно и в чем нет необходимости для простого меню переходов. Эти небольшие дополнения в функцию предотвратят отправку формы браузером.

```
<script language="JavaScript">

/* Если вы вставляете этот скрипт во внешний
JavaScript-файл, теги script, находящиеся в начале
и в конце данного листинга, не нужны. Если вы
вставляете этот фрагмент кода в HTML-документ, оста-
вьте теги script, и добавьте код в раздел head
вашей страницы. */

   function doJumpMenuGraphicButton() {

      var menu = document.jumpmenu.pages;

/* Следующая строка получает значение атрибута
value для выбранного элемента меню. */

      var menuValue = menu.options[menu.
   selectedIndex].value;

      if (menuValue == "") {

/* Следующая строка предотвращает отправку формы. */

         return false;

      } else {

/* Следующая строка выполняет переход на выбранную
страницу, если значение атрибута value не было пус-
тым. */

         location.href = menuValue;

/* Следующая строка предотвращает отправку формы. */

         return false;

      }

   }

</script>
```

ГЛАВА 9.
Управление Всплывающими окнами

В фэн-шуй Web-дизайна всплывающие окна подобны кабинету с углами странной формы, который не всем подходит. Некоторые люди никогда не любили всплывающие окна, называя их враждебными по отношению к пользователю, считая их лично оскорбительными и полагая что их могут делать только люди, у которых нет других проблем. Однако всплывающие окна стали конкурентоспособными. Они заняли свое место в Web-дизайне. Всплывающие окна отлично подходили для вызова экранов помощи, быстрых определений, расширенных описаний и других кусочков полезной, но второстепенной информации, которая не заслуживала помещения на собственную страницу. При правильном использовании всплывающие окна помогали сделать схему навигации более изящной и сфокусированной.

Определение

Всплывающее окно – это отдельное, обычно меньшее по размерам окно браузера, в котором загружается предопределенная страница.

Позднее корыстные рекламодатели, вроде людей из отдела рекламы, взяли на вооружение всплывающие окна и превратили их в раздражающую сферу спама. Скоро вы не сможете посетить какой-либо ресурс в Интернете без многократно повторяющихся, сверкающих, медленно загружающихся окон с содержимым, едва имеющим отношение к сайту, которое вам не нужно, которое вы не запрашивали или не хотели получить. Поэтому всплывающие окна везде потеряли доверие, и дизайнеры не используют их по назначению. Необычно видеть всплывающие окна, содержащие существующий сервис Web-сайта, в то время как продолжают открываться всплывающие окна с рекламой.

Если вы желаете перевоспитать своих посетителей и иметь дело со случайными электронными письмами, наполненными ненавистью, всплывающие окна все еще могут оставаться ценным свойством вашего сайта. Они не стали менее полезными для отображения второстепенного содержимого со времени, когда всплывающие окна были на пике популярности. Они могут превратить во всех отношениях неудобный дизайн, как представленный на рисунке 9.1, в более понятный, как показано на рисунке 9.2.

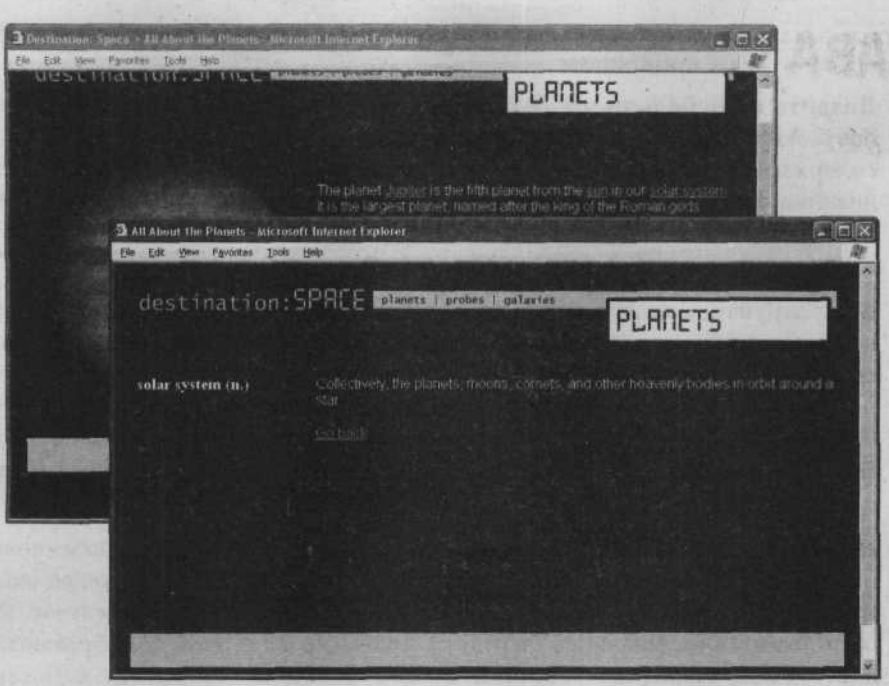


Рис. 9.1. В этом дизайне щелчок на термине приводит к загрузке страницы с определением. Хорошая идея, но неуклюжая реализация. Посетитель вынужден переходить по страницам взад и вперед, что прерывает поток основного текста

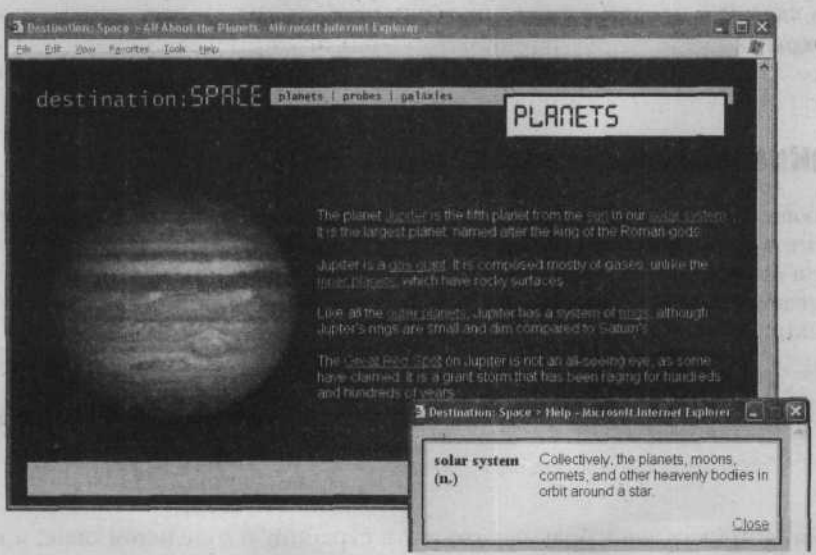


Рис. 9.2. Если же загрузить определения во всплывающее окно, посетителям не придется терять место, где они остановились в основном тексте, чтобы получить вспомогательную информацию

Хитрость заключается в эффективном использовании всплывающих окон. Вот несколько руководящих принципов:

- **Делайте их небольшими**. Это относится к физическим размерам всплывающего окна (которые можно контролировать) и к помещаемому в него объему содержимого. Не пытайтесь его переполнять слишком большим количеством дизайна. Большие всплывающие окна с множеством изображений и сложными макетами могут слишком долго загружаться, что даст вашим посетителям все причины полагать, что загружается моргающее приставание.
- **Фокусируйтесь на его содержимом**. Содержимое всплывающих окон должно быть коротким и приятным. Если вы развернулись так, что появляется необходимость создания в окне полос прокрутки, возможно, вам следует еще раз подумать о своей стратегии. Помните, что всплывающие окна наилучшим образом подходят для быстрого отображения второстепенной информации. Большие порции информации находятся на законченных страницах вашего сайта.
- **Не полагайтесь на них**. Поскольку некоторые люди никогда не побеспокоятся посмотреть на самое невинное, разработанное с умом всплывающее окно, не используйте их для критически важного содержимого. В любом случае, эта работа ложится на основные страницы. Всплывающие окна приобретают больший смысл, выполняя вспомогательную функцию; например, коротенькие страницы помощи.
- **Предоставьте управление посетителям**. Не заставляйте всплывающие окна открываться автоматически! Всплывающее окно должно открываться только тогда, когда посетитель определенно щелкнул на ссылке или кнопке. Кроме того, сделайте простым закрытие всплывающего окна, после того, как оно было открыто.

Создание Всплывающих окон

Всплывающее окно на самом деле является такой же простой страницей, как и любые другие. Содержимое всплывающего окна помещается в HTML-файл, как это делается для обычной Web-страницы, и в любом месте на сайте можно создать ссылку на всплывающее окно. Однако вместо указания пути к странице, как в случае обычной ссылки:

```
<a href="popup.htm">Open the popup window</a>
```

мы направляем браузер на простую функцию, написанную на языке JavaScript:

```
<a href="javascript:doPopup('popup.htm');">Open the popup window</a>
```

Эта функция приказывает браузеру открыть страницу в отдельном окне, а не загружать ее в основном окне. Вы можете написать собственный код на языке JavaScript, выполняющий эту задачу, или можете использовать следующий код:

```
<script language="JavaScript">
   function doPopup(popupPath) {
      window.open(popupPath,'name',
         'width=400,height=200,scrollbars=YES');
   }
</script>
```

Вот и все. Инструкция `window.open` позаботится обо всем, если вы передали в функцию правильный путь. Поэтому, если HTML-файл вашего всплывающего окна находится не в той папке, где расположена вызывающая страница, следует отформатировать ссылку подобным образом:

```
<a href="javascript:doPopup('../help/info.htm');">Get help</a>
```

или любой возможный правильный путь. Вставьте путь между одинарными кавычками. Если передать в функцию неправильный путь, во всплывающем окне загрузится неожиданное содержимое или знаменитая страница File Not Found (Файл не найден).

Язык JavaScript предоставляет достаточное количество возможностей для управления размерами и поведением нового окна. Всплывающему окну можно присвоить уникальное имя, которое передается в инструкцию `window.open` в качестве второго параметра, следующего за путем. Замените слово name в предыдущем скрипте на слово *popup*, *help*, *window*, *Charlie* или любое другое слово, которое вам нравится.

За именем окна следует список свойств, в котором определяется внешний вид всплывающего окна. В одной строке можно объединить столько свойств, сколько пожелаете, – просто отделяйте свойства друг от друга запятой и не вставляйте пробелы после запятых. В таблице 9.1 приведены некоторые распространенные свойства.

Итак, если вы хотите для всплывающего окна отобразить поле адреса и строку состояния в функции `doPopup()`, просто расширьте набор свойств, как в следующем примере:

```
<script language="JavaScript">
   function doPopup(popupPath) {
      window.open(popupPath,'name',
         'width=300,height=150,scrollbars=YES,location=YES,
      status=YES');
   }
</script>
```

На рисунке 9.3 показаны различные части окна браузера, если вам необходимо освежить память.

Табл. 9.1. Распространенные свойства всплывающего окна

Свойство	Описание	Пример
height	Задает высоту всплывающего окна в пикселах	height=300
location	Отвечает за отображение поля адреса во всплывающем окне	location=YES
menubar	Отвечает за отображение строки меню во всплывающем окне	menubar=YES
resizable	Позволяет посетителю изменять размер и форму всплывающего окна	resizable=YES
scrollbars	Отвечает за отображение горизонтальных и вертикальных полос прокрутки во всплывающем окне, если необходимо	scrollbars=YES
status	Отвечает за отображение строки состояния во всплывающем окне	status=YES
toolbar	Отвечает за отображение панели инструментов во всплывающем окне	toolbar=YES
width	Задает ширину всплывающего окна в пикселах	width=150

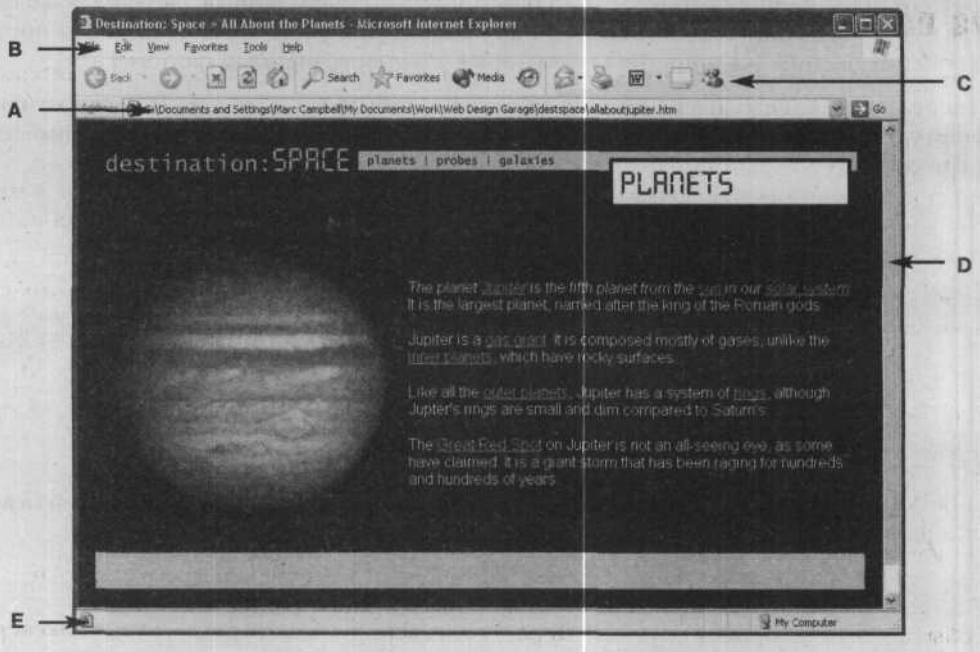


Рис. 9.3. Вот части окна браузера: (A) Адресная строка, (B) Строка меню, (C) Панель инструментов, (D) Полоса прокрутки, (E) Строка состояния

Совет

Если на вашем сайте используются различные типы всплывающих окон, возможно, вы захотите присвоить каждому типу различные имена и различные наборы свойств. Чтобы сделать это, измените ссылку на всплывающее окно подобным образом (подставив реальные значения вместо заполнителей):

```
<a  href="javascript:doPopup('path','name','proper-
ties');">
```

Как видите, этот код передает три значения в функцию doPopup, а не одно. Теперь, чтобы использовать эти значения, просто внесем небольшие изменения в код JavaScript-функции:

```
function doPopup(popupPath, popupName, popupProper-
ties) {
      window.open(popupPath, popupName, popupProper-
ties);
}
```

Перемещение всплывающего окна на передний план

Одна из проблем со всплывающими окнами заключается в том, что они стремятся исчезнуть из вида. Всплывающие окна сворачиваются на панели задач вместе с небольшой коллекцией рекламы, спрятанной под главным окном браузера. Если ваш сайт загрузит другую страницу в открытое всплывающее окно – это происходит часто, когда всплывающее окно используется для отображения подсказок помощи, – опущенное всплывающее окно не перепрыгнет обратно наверх, если вы определенным образом не заставите его сделать это.

Чтобы сделать это, добавьте событие `onLoad` к тегу `body` HTML-файла всплывающего окна:

```
<body onLoad="window.focus();">
```

Инструкция `window.focus` гарантирует, что всплывающее окно займет верхнюю позицию на рабочем столе, но только когда загрузится страница. Всплывающее окно снова исчезнет из вида, когда посетитель щелкнет на главном окне браузера.

Если вы действительно хотите раздражать ваших посетителей и предотвратить исчезание всплывающего окна из вида, попробуйте следующее:

```
<body onBlur="window.focus();">
```

Событие `onBlur` происходит всякий раз, когда всплывающее окно теряет свою верхнюю позицию, после чего инструкция `window.focus` перемещает всплывающее окно обратно на передний план. Единственный способ избавиться от подобного всплывающего окна – закрыть его.

Совет

Никогда, никогда, никогда не используйте в дизайне нескрывающиеся всплывающие окна. Ваше всплывающее окно может быть крепче, чем Титаник, но ваш сайт утонет, как камень.

Добавление ссылки Close

Простой трюк с использованием языка JavaScript позволяет добавить ссылку `Close` (Закрыть) в тело вашего всплывающего окна, что было продемонстрировано на рисунке 9.2. Когда посетители щелкнут на этой ссылке, всплывающее окно закроется, избавив их от необходимости использовать кнопку X вверху окна.

Код для этой ссылки выглядит следующим образом:

```
<a href="javascript:window.close();">Close</a>
```

Для этого даже не нужно создавать отдельную функцию. Одна инструкция `window.close` делает все, что вам необходимо.

ГЛАВА 10.
Создание открывающихся меню

Открывающиеся меню очень популярны в наши дни. Несомненно вы видели их в Интернете. Вы наводите указатель мыши на элемент главного меню навигации, и появляется меню со связанными элементами. Открывающееся меню обладает стильным эффектом, как показано на рисунке 10.1.

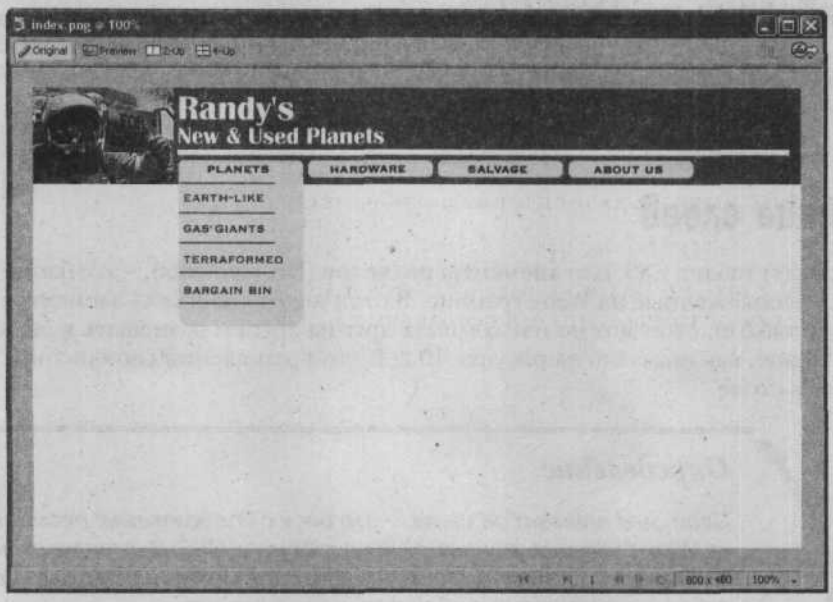


Рис. 10.1. Ничто не говорит о стиле так, как всплывающее меню

Определение

Открывающееся меню – это список элементов навигации, который появляется при наведении указателя на элемент в главном меню навигации.

Реализация открывающихся меню была кошмаром. К счастью, программа Internet Explorer с версии 5 и программа Netscape с версии 6 поддерживают метод `getElementById` языка JavaScript, упрощающий написание скриптов для всех видов интерактивных возможностей, включая и открывающиеся меню. Браузеры, понимающие функцию `getElementId`, доступны уже пару лет, и теперь они являются доминирующими браузерами в Интернете; поэтому для большинства типов сайтов можно с чистой совестью писать скрипты, используя этот метод. Тем не менее, у более старых браузеров есть проблемы с данным методом, поэтому, имейте это в виду.

Определение

Динамический HTML, или DHTML, – это комбинация языков HTML и JavaScript, и каскадных таблиц стилей (CSS).

Динамический HTML, или **DHTML**, – это комбинация языков HTML и JavaScript, и каскадных таблиц стилей (CSS), которая применяется для создания интерактивных Web-сайтов. Типичное открывающееся меню является отличным примером языка DHTML в действии. Макет навигационной панели создается на языке HTML. Элементы таблиц CSS, называемые **слоями**, предоставляют меню, а несколько простых функций на языке JavaScript оживляют меню.

Понятие слоев

Слои (Layers) таблиц CSS, или **элементы разделов** (Div elements), – это блоки с содержимым, расположенные на Web-странице. В отличие от «плоских» элементов, например ячеек таблиц, слои можно накладывать друг на друга и размещать в любой позиции на экране, как показано на рисунке 10.2. В этом отношении слои похожи на окна на рабочем столе.

Определение

Слой, или элемент раздела, – это блок с содержимым, размещенный на Web-странице, основанный на таблицах CSS. В отличие от плоских элементов, например, ячеек таблиц, слои можно накладывать друг на друга и размещать в любой позиции на экране.

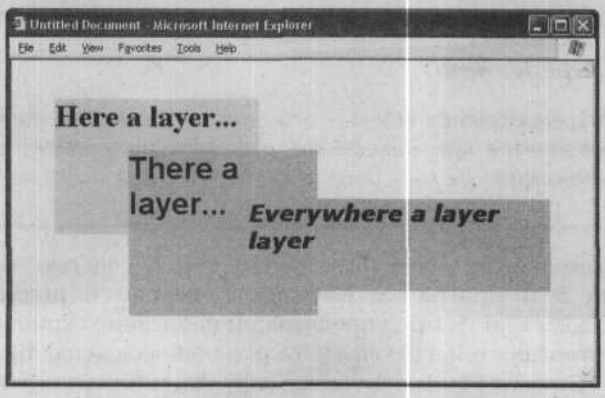


Рис. 10.2. Слои таблиц CSS – это блоки с содержимым. В отличие от ячеек таблиц, слои можно накладывать друг на друга и свободно размещать в любой позиции на странице

Над слоями можно производить любые действия, используя язык JavaScript. Их можно анимировать. Можно изменять размер слоев. Их можно сделать перетаскиваемыми. И можно сделать так, чтобы они исчезали и появлялись снова по желанию, что и использует большинство разработчиков для создания эффекта всплывания в открывающихся меню.

Каждое открывающееся меню размещается на отдельном слое, как показано на рисунке 10.3. Однако вместо того, чтобы сделать эти слои видимыми, вы указываете, что по умолчанию они должны быть спрятанными или невидимыми. Меню все еще находятся там же – они запрограммированы в коде, но посетитель не может увидеть их или взаимодействовать с ними. Они напоминают привидений: невидимые, неосязаемые и на них нельзя щелкнуть. Меню остаются в данном состоянии до тех пор, пока посетитель не наведет указатель мыши на определенную «горячую» точку, например, на ссылку в навигационной панели. Затем начинает действовать JavaScript-функция и изменяет значение свойства `visibility` для слоя, который вы хотите показать. В результате отобразится скрытый слой, и у вас будет открывающееся меню, как показано на рисунке 10.4. Другой скрипт, который прячет меню, когда посетитель завершает работу с ним, снова заставляет слой стать невидимым.

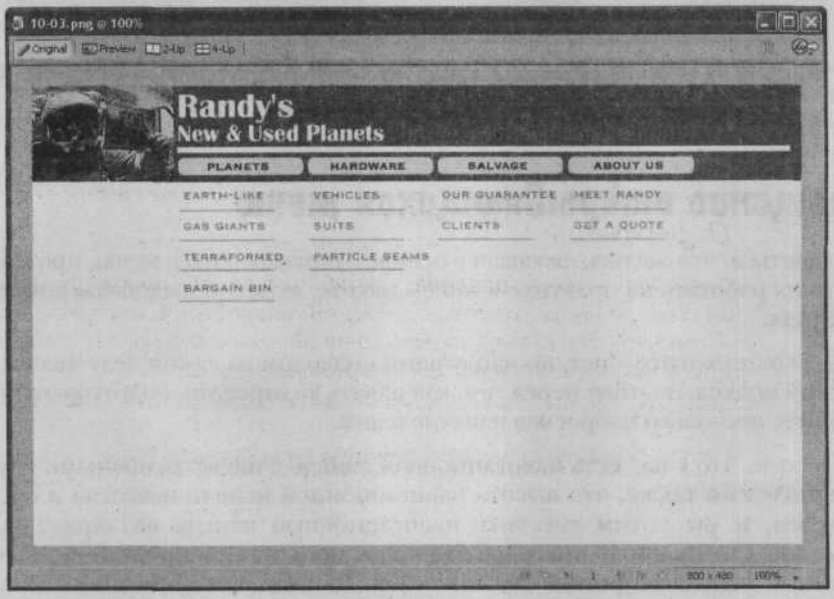


Рис. 10.3. Каждое открывающееся меню находится на отдельном слое. Для создания эффекта всплывания по умолчанию слои спрятаны или невидимы

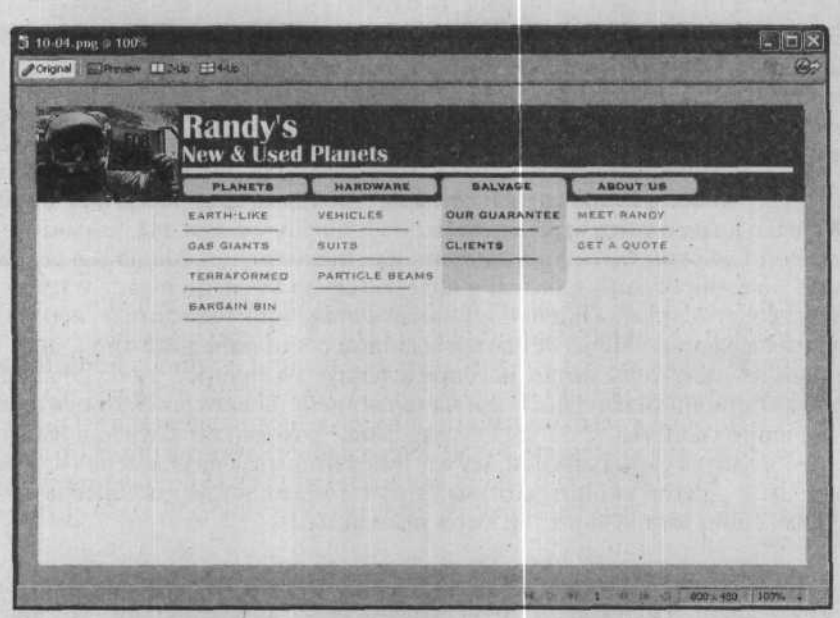


Рис. 10.4. Наведение указателя мыши на категорию навигационной панели приводит к отображению соответствующего слоя

Размещение открывающихся меню

Будем надеяться, что логика, лежащая в основе открывающихся меню, проста. Чтобы заставить их работать на практике в вашем макете, может понадобиться немного тяжелого труда.

Хотите – верьте, хотите – нет, но наилучшим подходом на самом деле является математический подход, поэтому перед тем, как начать кодировать, подготовьте черновик и выполните несколько набросков и вычислений.

Предположим, что у нас есть навигационная панель с пятью основными категориями. Предположим также, что высота навигационной панели известна и составляет 19 пикселов, и мы хотим уместить навигационную панель на экране шириной 800 пикселов. Отнимаем 40 пикселов для полос прокрутки и прочего и получаем навигационную панель шириной 760 пикселов. Разделим получившуюся ширину на пять одинаковых частей, ширина каждой из которых составляет 152 пиксела, как показано на рисунке 10.5.

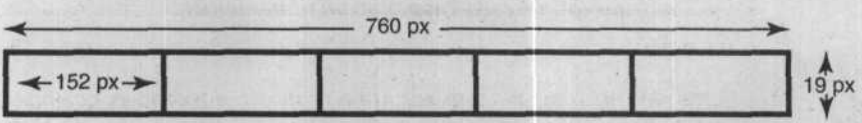


Рис. 10.5. Начните с приблизительного наброска дизайна навигационной панели и вычисления размера каждой ячейки таблицы

Совет

Чтобы данные расчеты соответствовали как в программе Netscape, так и в программе IE, обязательно установите значения следующих атрибутов промежутков в 0: `leftmargin`, `topmargin`, `marginwidth` *и* `marginheight` *в теге* `body`; *и* `border`, `cellpadding` *и* `cellspacing` *в каждом теге* `table`.

Теперь перейдем к слоям. Мы знаем ширину каждого слоя: 152 пиксела, как и ширина каждой ячейки таблицы навигационной панели. Необходимо рассчитать высоту каждого слоя, которая зависит от того, сколько навигационных элементов вы хотите расположить на конкретном слое. Очевидно, что это число может изменяться от меню к меню. Поскольку высота навигационной панели равняется 19 пикселам, упростим себе жизнь и возьмем высоту в 19 пикселов в качестве стандартной высоты элемента меню. Высота меню с тремя элементами должна составлять 57 пикселов (19 умножить на 3), а высота меню с семью элементами – 19 умножить на 7, или 133 пикселов.

Чтобы выровнять слои с ячейками таблицы, положение каждого слоя выражается в смещении от верхней и левой стороны окна браузера. Таким образом, если высота навигационной панели составляет 19 пикселов, мы желаем разместить слои по вертикали с отступом, равным 19 пикселам от верхней стороны экрана. По горизонтали размещение слоев начинается с 0 (без смещения от левой стороны) и позиция каждого нового слоя увеличивается на 152, как показано на рисунке 10.6.

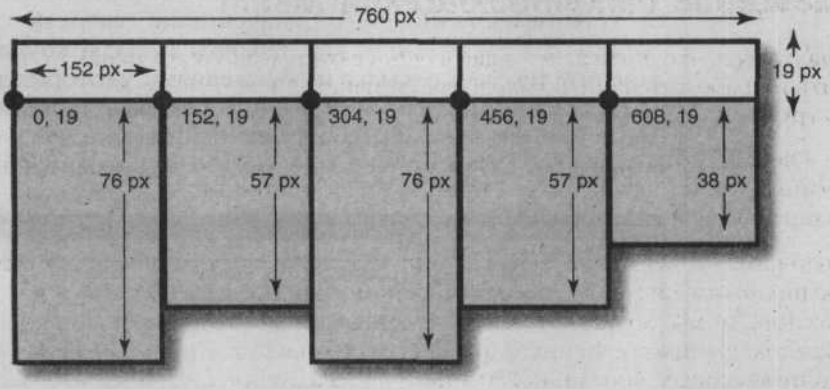


Рис. 10.6. Размещение слоев по смещениям от верхней и левой стороны окна браузера

Совет

Если внутри открывающихся меню вместо текстовых ссылок вы используете изображения, не забудьте добавить дополнительное свободное место в дизайн изображений. Это поможет сделать меню менее сдавленными.

И, наконец, создайте таблицу для меню в каждом слое, используя одну строку для каждого элемента меню, как показано на рисунке 10.7.

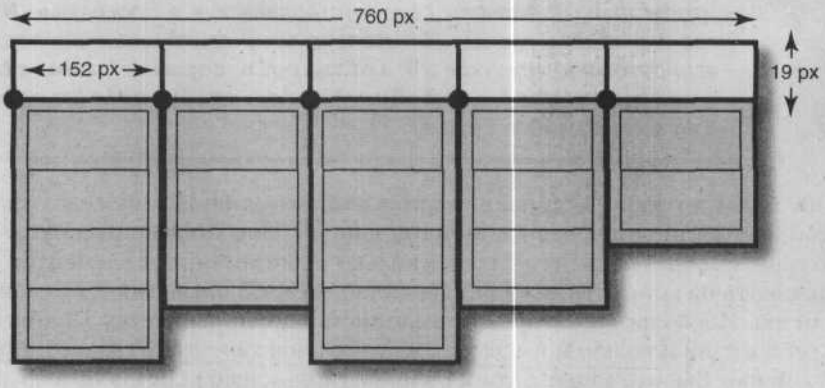


Рис. 10.7. Поместите таблицу внутрь каждого слоя для меню. Каждый элемент меню получает отдельную строку в таблице

Инструментарий

Создание открывающихся меню

Этот DHTML-документ демонстрирует создание системы открывающихся меню, как показано на рисунке 10.8. Для простоты, в навигационной панели и в меню используются текстовые ссылки. С установленными в 0 значениями свойств `border`, `cellpadding` и `cellspacing`, таблицы выглядят компактными. Замените текстовые ссылки изображениями, у которых достаточно свободного места по вертикали, и ваш макет будет выглядеть еще лучше, как показано на рисунке 10.9. Однако придется изменить высоту слоев, чтобы разместить более высокие таблицы.

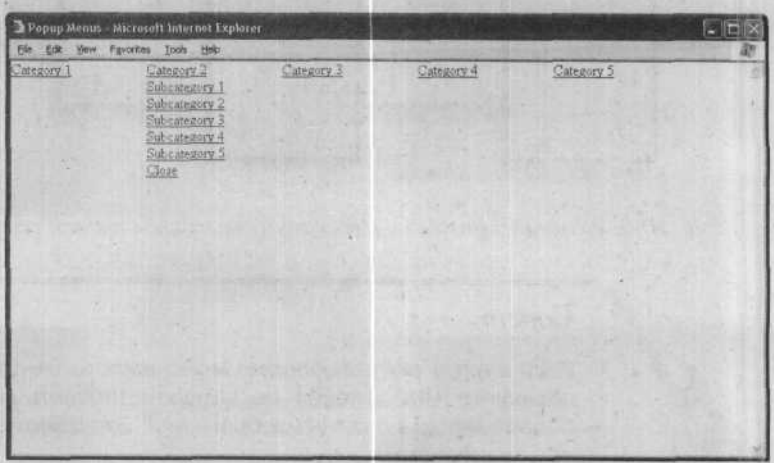


Рис. 10.8. Данный инструментарий создает систему открывающихся меню

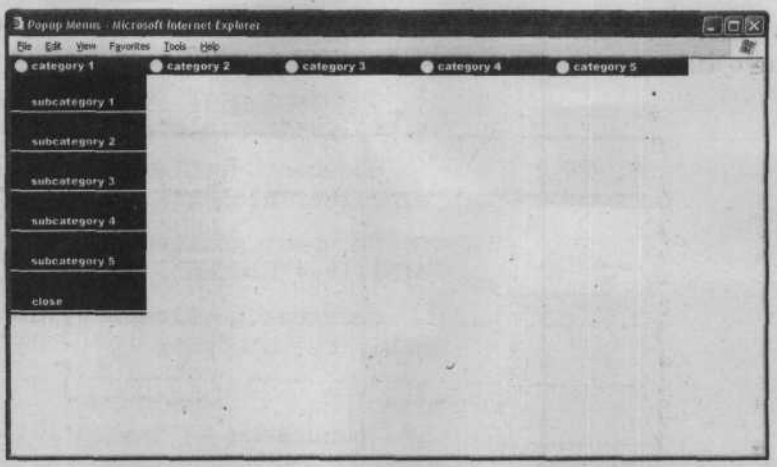


Рис. 10.9. Замените текстовые ссылки собственными изображениями (добавив достаточно свободного места по вертикали), и меню не будут выглядеть сдавленными

```
<html>
<head>
<title>Popup Menus</title>
<script language="JavaScript">
   function doShowHide(popupMenu) {
/* Данная функция выполняется, когда посетитель наводит указатель мыши на элементы навигационной панели. Она получает ID соответствующего меню и делает это меню видимым, при этом скрывая четыре остальные меню. */
      if (popupMenu == "menu1") {
         document.getElementById("menu1").style.visibility="visible";
         document.getElementById("menu2").style.visibility="hidden";
         document.getElementById("menu3").style.visibility="hidden";
         document.getElementById("menu4").style.visibility="hidden";
         document.getElementById("menu5").style.visibility="hidden";
      }
      if (popupMenu == "menu2") {
```

```
    document.getElementById("menu1").style.vi
sibility="hidden";
    document.getElementById("menu2").style.vi
sibility="visible";
    document.getElementById("menu3").style.vi
sibility="hidden";
    document.getElementById("menu4").style.vi
sibility="hidden";
    document.getElementById("menu5").style.vi
sibility="hidden";
}
if (popupMenu == "menu3") {
    document.getElementById("menu1").style.vi
sibility="hidden";
    document.getElementById("menu2").style.vi
sibility="hidden";
    document.getElementById("menu3").style.vi
sibility="visible";
    document.getElementById("menu4").style.vi
sibility="hidden";
    document.getElementById("menu5").style.vi
sibility="hidden";
}
if (popupMenu == "menu4") {
    document.getElementById("menu1").style.vi
sibility="hidden";
    document.getElementById("menu2").style.vi
sibility="hidden";
    document.getElementById("menu3").style.vi
sibility="hidden";
    document.getElementById("menu4").style.vi
sibility="visible";
    document.getElementById("menu5").style.vi
sibility="hidden";
}
if (popupMenu == "menu5") {
    document.getElementById("menu1").style.vi
sibility="hidden";
    document.getElementById("menu2").style.vi
sibility="hidden";
```

```
        document.getElementById("menu3").style.vi
    sibility="hidden";

        document.getElementById("menu4").style.vi
    sibility="hidden";

        document.getElementById("menu5").style.vi
    sibility="visible";

    }

  }

  function doHide(popupMenu) {

/* Эта функция запускается, когда посетитель щел-
кает на кнопке Close (Закрыть) внизу меню. Она по-
лучает ID текущего меню и затем прячет это меню.
Просто, как пирог. */

    document.getElementById(popupMenu).style.vis
  ibility="hidden";

  }

</script>

</head>

<body leftmargin="0" topmargin="0" marginwidth="0"
marginheight="0">

<!— Теперь по порядку. Создаем код для открывающих-
ся меню. Само меню - это таблица, состоящая из шес-
ти строк: по одной строке для каждого из пяти эле-
ментов меню и еще одна строка для ссылки Close
(Закрыть). Если вам нужно меньше элементов, удали-
те строки из таблицы. Если необходимо больше эле-
ментов, добавьте строки в таблицу. Просто не за-
будьте разместить таблицу между тегами div, кото-
рые вставляют элемент в слой.

Обратите внимание на тег div, у которого свойство
id задает ID слоя, чрезвычайно важный для
JavaScript-функций, приведенных выше. В значении
атрибута style обратите внимание, как размещение
слоя выражается в виде смещений от левой и верхней
границ.

Значения атрибутов width и height задают физичес-
кие размеры слоя. Ширина, равная 152, получилась,
исходя из ширины навигационной панели (760 пикс-
лов), деленной на число основных категорий, пять.
Высота, равная 114, - это высота элемента меню (19
пикселов), умноженная на число элементов в меню,
шесть (включая ссылку Close (Закрыть)). Высота слоя
```

изменяется в зависимости от высоты каждого элемента меню и общего числа элементов в меню. Число 19, используемое в качестве высоты каждого элемента меню, не высечено в камне. После того, как вы поработаете с размером и форматом каждого меню для размещения на вашем сайте, не забудьте соответствующим образом изменить значение высоты.

Обратите внимание, что значение атрибута visibility установлено в hidden, что делает слой невидимым по умолчанию. -->

```
<div id="menu1" style="position: absolute; left:
0px; top: 19px; width: 152px; height:114px; visi-
bility: hidden;">
   <table width="152" border="0" cellpadding="0"
cellspacing="0">
      <tr>
        <td><a href="path01">Subcategory 1</a></td>
      </tr>
      <tr>
        <td><a href="path02">Subcategory 2</a></td>
      </tr>
      <tr>
        <td><a href="path03">Subcategory 3</a></td>
      </tr>
      <tr>
        <td><a href="path04">Subcategory 4</a></td>
      </tr>
      <tr>
        <td><a href="path05">Subcategory 5</a></td>
      </tr>
<!- Эта строка содержит элемент Close (Закрыть)
внизу меню. Щелчок на этой ссылке запускает функцию
doHide. Ссылка передает ID меню для закрытия. -->
      <tr>
        <td><a href="javascript:doHide('menu1');"
      >Close</a></td>
      </tr>
   </table>
</div>
<!- Этот элемент div содержит второе открывающееся
меню. Оно работает так же, как и первое. -->
```

```
<div id="menu2" style="position: absolute; left:
152px; top: 19px; width: 152px; height: 114px; vis-
ibility: hidden;">
   <table width="152" border="0" cellpadding="0"
cellspacing="0">
      <tr>
        <td><a href="path06">Subcategory 1</a></td>
      </tr>
      <tr>
        <td><a href="path07">Subcategory 2</a></td>
      </tr>
      <tr>
        <td><a href="path08">Subcategory 3</a></td>
      </tr>
      <tr>
        <td><a href="path09">Subcategory 4</a></td>
      </tr>
      <tr>
        <td><a href="path10">Subcategory 5</a></td>
      </tr>
      <tr>
         <td><a href="javascript:doHide('menu2');"
         >Close</a></td>
      </tr>
   </table>
</div>
<!- Этот элемент div содержит третье открывающееся
меню. ->
<div id="menu3" style="position: absolute; left:
304px; top: 19px; width: 152px; height: 114px; vis-
ibility: hidden;">
   <table width="152" border="0" cellpadding="0"
cellspacing="0">
      <tr>
        <td><a href="path11">Subcategory 1</a></td>
      </tr>
      <tr>
        <td><a href="path12">Subcategory 2</a></td>
      </tr>
      <tr>
```

```
        <td><a href="path13">Subcategory 3</a></td>
      </tr>
      <tr>
        <td><a href="path14">Subcategory 4</a></td>
      </tr>
      <tr>
        <td><a href="path15">Subcategory 5</a></td>
      </tr>
      <tr>
        <td><a href="javascript:doHide('menu3');"
      >Close</a></td>
      </tr>
   </table>
</div>
<!- Этот элемент div содержит четвертое открываю-
щееся меню. ->
<div id="menu4" style="position: absolute; left:
456px; top: 19px; width: 152px; height: 114px; vis-
ibility: hidden;">
   <table width="152" border="0" cellpadding="0"
cellspacing="0">
      <tr>
        <td><a href="path16">Subcategory 1</a></td>
      </tr>
      <tr>
        <td><a href="path17">Subcategory 2</a></td>
      </tr>
      <tr>
        <td><a href="path18">Subcategory 3</a></td>
      </tr>
      <tr>
        <td><a href="path19">Subcategory 4</a></td>
      </tr>
      <tr>
        <td><a href="path20">Subcategory 5</a></td>
      </tr>
      <tr>
        <td><a href="javascript:doHide('menu4');"
      >Close</a></td>
```

```
      </tr>
   </table>
</div>
<!- Этот элемент div содержит пятое открывающееся
меню. ->
<div id="menu5" style="position: absolute; left:
608px; top: 19px; width: 152px; height: 114px; vis-
ibility: hidden;">
   <table width="152" border="0" cellpadding="0"
cellspacing="0">
      <tr>
         <td><a href="path21">Subcategory 1</a></td>
      </tr>
      <tr>
         <td><a href="path22">Subcategory 2</a></td>
      </tr>
      <tr>
         <td><a href="path23">Subcategory 3</a></td>
      </tr>
      <tr>
         <td><a href="path24">Subcategory 4</a></td>
      </tr>
      <tr>
         <td><a href="path25">Subcategory 5</a></td>
      </tr>
      <tr>
         <td><a href="javascript:doHide('menu5');"
      >Close</a></td>
      </tr>
   </table>
</div>
<!- И, наконец, таблица, создающая навигационную
панель. Это первый (и единственный) видимый элемент
на странице. Остальное содержимое спрятано в неви-
димых слоях. ->
<table width="760" border="0" cellpadding="0"
cellspacing="0">
   <tr>
      <td width="152">
```

```
<!– Событие onMouseOver возникает, когда посетитель
помещает указатель мыши над «горячей» точкой.
В данном случае горячей точкой является ссылка для
Category 1. Посетитель наводит указатель мыши на
эту ссылку, и браузер запускает функцию doShowHide,
передавая в функцию ID соответствующего меню. –>
          <a href="path26" onMouseOver="doShowHide
      ('menu1');">
             Category 1
          </a>
      </td>
      <td width="152">
          <a href="path27" onMouseOver="doShowHide
      ('menu2');">
             Category 2
          </a>
      </td>
      <td width="152">
          <a href="path28" onMouseOver="doShowHide
      ('menu3');">
             Category 3
          </a>
      </td>
      <td width="152">
          <a href="path29" onMouseOver="doShowHide
      ('menu4');">
             Category 4
          </a>
      </td>
      <td width="152">
          <a href="path30" onMouseOver="doShowHide
      ('menu5');">
             Category 5
          </a>
      </td>
   </tr>
</table>
</body>
</html>
```

ГЛАВА 11.
Обеспечение путей отхода

Представьте, что вы вошли в комнату через дверь, а когда повернулись, чтобы выйти, дверного проема уже не было. Вы бы не стали разрабатывать проект такого дома, кроме как для карнавала. Однако в Интернете существует достаточное количество сайтов, похожих на павильон смеха, которые по существу делают то же самое – выбрасывают посетителей на страницу, не предоставляя при этом пути назад.

Разработка хорошей навигации помогает предотвратить эффект павильона смеха. Рассмотрим диаграмму на рисунке 11.1. В этой схеме навигации посетители переходят от страницы верхнего уровня к страницам нижних уровней шаг за шагом, удачно разработанным, понятным, прямолинейным способом. Предоставление обратного пути сводится к простому размещению на каждой странице нижнего уровня ссылки, ссылающейся на следующую верхнюю страницу в структуре сайта,точно так же, как на каждой странице верхнего уровня размещается ссылка на следующую страницу нижнего уровня.

Множество Web-сайтов не строго придерживается подобной прямолинейности. На самом деле, хороший Web-сайт обычно является нелинейным, позволяя посетителю выполнять большие скачки по структуре сайта в интересах скорости. Схемы, наподобие представленной на рисунке 11.2, совсем не редкость. В данном случае посетитель может попасть на страницу продукта несколькими различными путями: выполняя переход по иерархии шаг за шагом, перейдя через раздел **Specials** (Специальные предложения) или перепрыгнуть прямо на страницу продукта, воспользовавшись определенными элементами на начальной странице.

Совет

Когда посетитель опускается вглубь структуры сайта на несколько уровней, принято связывать каждую из страниц нижнего уровня непосредственно со страницей верхнего уровня. Это очень удобно, поскольку посетитель имеет возможность сделать несколько шагов за один раз. Но не забывайте также включать пошаговые переходы. Всегда предоставляйте посетителям возможность вернуться обратно на страницу, которую они посетили непосредственно перед текущей страницей.

Для такого сценария предоставление обратного пути является не таким уж простым действием, как связывание страниц в обоих направлениях. Мы бы создали беспорядок на экране, разместив ссылки на все возможные ссылаемые страницы. В то же время, для нелинейного сайта предоставление обратного пути является еще более важным из-за абсолютного количества возможностей. Вы же не хотите, чтобы посетители чувствовали себя затерянными в океане.

Рис. 11.1. Когда сайт построен с использованием линейного способа, обеспечивайте связи в обоих направлениях: со страницы верхнего уровня на страницу нижнего и со страницы нижнего уровня на страницу верхнего

Рис. 11.2. Нелинейный сайт предоставляет множество способов получения одной и той же страницы. Предоставление обратного пути не является таким же простым, как связывание страниц в обоих направлениях

Если посетители потеряются, они воспользуются кнопкой **Back** (Назад) своего браузера, который выполнит эту работу. Это также отвлечет их внимание от сайта и его содержимого, что не очень хорошо. Это плохо и для посетителя, поскольку замедляется его работа. Простая ссылка с использованием языка JavaScript полностью решает эту проблему:

```
<a href="javascript:history.back();">Go back</a>
```

Метод `history.back` работает точно так же, как и кнопка **Back** (Назад). Метод загружает предыдущую страницу. Не имеет значения, какую дверь используют ваши посетители, ссылка `Go Back` (Перейти назад) оставит открытой для них ту же самую дверь.

Совет

Некоторые дизайнеры, слишком тесно сотрудничающие с отделом рекламы, ощущают необходимость заблокировать кнопку браузера ***Back*** *(Назад), чтобы заставить пользователя остаться на сайте, якобы посетитель каким-то образом забудет закрыть окно браузера и никогда больше не вернется. Пожалуйста, воздержитесь от побуждения управлять навыками работы посетителя с браузером. Ваш сайт никогда не должен блокировать кнопку* ***Back*** *(Назад) (или любые другие кнопки браузера, коли на то пошло), даже если ваша навигация является безупречной. Некоторые посетители предпочитают использовать интерфейс браузера. Кто посмеет осудить их? Обычно посетители больше полагаются на интерфейс браузера, чем на интерфейс сайта.*

Совет

Сделайте так, чтобы ссылку ***Go Back*** *(Перейти назад) было просто найти, иначе ваши посетители будут просто использовать кнопку* ***Back*** *(Назад) своего браузера, что делает бессмысленным размещение ссылки* ***Go Back*** *(Перейти назад) на странице.*

Метод `history.back` – это не единственный метод объекта `history` языка JavaScript. В таблице 11.1 перечислены некоторые другие методы для справки, однако применяйте их умеренно. Ссылка **Go Back** (Перейти назад) на самом деле очень полезна, однако ссылка **Go Forward** (Перейти вперед) или ссылка **Go Two Steps Back** (Перейти на два шага назад) создают неразбериху.

Табл. 11.1. Методы объекта **history** языка JavaScript

Метод	Описание	Пример
`back()`	Переходит на один шаг назад	`history.back()`
`forward()`	Переходит на один шаг вперед	`history.forward()`
`go(x)`	Переходит на x шагов вперед (положительное значение) или на x шагов назад (отрицательное значение)	`history.go(3)`, `history.go(-2)`

ГЛАВА 12.

Работа со скриптами и таблицами стилей

Интернет больше не описывается только языком HTML. Такая ситуация наблюдается уже несколько лет. Даже если не принимать во внимание технологии, например, для соединения с базой данных, и языки программирования сервера, например, PHP и Cold Fusion, разработчики Web-сайтов полагаются на интерактивную мощь скриптов и управление стилями каскадных таблиц стилей (Cascading Style Sheets – CSS), чтобы сделать свои сайты более интерактивными и эстетичными. Если вы уже попробовали некоторые инструментарии в данной книге, то заметили, что в них широко используются язык JavaScript и таблицы CSS.

Определение

Внедрение такого материала, как JavaScript- или CSS-кода, означает его включение в HTML-документ. Присоединение означает сохранение материала в отдельном файле и его последующее связывание с HTML-документом.

Используя на Web-сайте так много дополнительного материала, хорошо было бы посмотреть на то, как этот материал можно разместить. Для кода на языке JavaScript и таблиц CSS есть два варианта: внедрение и присоединение. Внедрение материала означает его размещение в том же документе, где находится HTML-код, а присоединение заключается в создании отдельного файла, содержащего скрипты или CSS-код, и последующем связывании HTML-кода с этим файлом.

Внедрение скриптов

Программа-браузер является глупой в том смысле, что считает все кодом на языке HTML. Подайте ему Web-страницу, и все будет в порядке. Однако если подать браузеру текстовый документ или JavaScript-код, он попытается интерпретировать этот документ, как код на языке HTML, и мы получим нечто похожее на то, что изображено на рисунке 12.1. Вместо выполнения внедренного скрипта, браузер посчитал его HTML-кодом и отобразил код на странице.

Чтобы избежать такой проблемы, необходимо сказать браузеру, где точно начинается и заканчивается скрипт. Для этого и необходим тег `script`. Поместите скрипт между парой тегов `script`, как показано в следующем примере:

```
<script language="JavaScript">
   function doAlert() {
      alert("I am a script.");
   }
</script>
```

В этом случае, когда браузер транслирует HTML-код, он знает, что нужно пропустить все, что находится между тегами `script`.

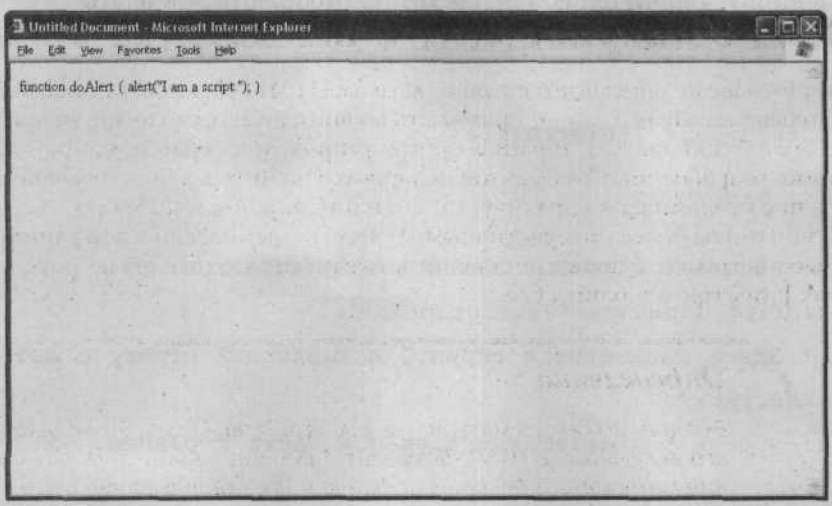


Рис. 12.1. Если внедрить скрипт, не заключив его в теги `script`*, браузер предположит, что это HTML-код, и отобразит его на странице*

Это первый шаг для внедрения скрипта. Второй шаг заключается в определении, где разместить внедренный скрипт в HTML-коде. Ответ зависит от того, к какому типу принадлежит скрипт. Скрипты с повторяющимися или интерактивными функциями (функции, которые могут вызываться на странице более чем один раз, или функции, изменяющие внешний вид страницы после того, как браузер сгенерировал ее) находятся вверху HTML-кода, между тегами `head` и после тега `title`:

```
<html>
   <head>
      <title>Page Title</title>
      <script language="JavaScript">
            /* JavaScript-код размещается здесь. */
      </script>
   </head>
   <body>
            /* Содержимое страницы размещается здесь. */
   </body>
</html>
```

Скрипты для ролловерных изображений и открывающихся меню попадают в эту категорию, как и скрипты, выполняющие проверку введенных данных.

Однако если у вас есть скрипт, который выполняется только раз, когда браузер первоначально генерирует страницу, как в случае со специальным текстом или временной меткой, внедрите скрипт прямо там, где хотите отобразить результаты:

```
<html>
   <head>
      <title>Page Title</title>
   </head>
   <body>
      <!- Вставка строки с датой вверху страницы. ->
      <script language="JavaScript">
      /* Здесь размещается скрипт, вставляющий строку с датой. */
      </script>
      <!- Здесь размещается оставшаяся часть страницы. ->
   </body>
</html>
```

Внедрение таблиц стилей

Как и в случае с кодом на языке JavaScript, браузер интерпретирует таблицы CSS, как HTML-код, если ему не указать иной способ. Теги `style` делают для таблиц CSS то же, что и теги `script` для JavaScript-кода. Они говорят браузеру рассматривать все, что находится между ними, как CSS-код, а не HTML:

```
<html>
   <head>
      <title>Page Title</title>
      <style type="text/css">
            /* CSS-код размещается здесь. */
      </style>
   </head>
   <body>
            /* Содержимое страницы размещается здесь. */
   </body>
</html>
```

Как и в случае с JavaScript-кодом, размещайте внедренные таблицы стилей в разделе HTML-кода `head`.

Совет

Некоторые старые браузеры не понимают теги `script` и `style`. Такие браузеры рассматривают все, что находится между данными тегами, как текст. Хотя трудно представить посетителя, получающего хоть какое-то удовольствие при посещении современных Web-сайтов с использованием такого антиквариата, возможно, вы захотите поместить теги комментариев языка HTML до и после тегов `script` или `style`, как показано ниже:

```
<script language="JavaScript">
      <!-


      ->
</script>
```

Добавление тегов комментариев предотвращает отображение старыми браузерами скриптов или кода в виде HTML-кода.

Присоединение скриптов

Альтернативой внедрению является присоединение, когда JavaScript-код размещается в отдельном файле. Однако чтобы на Web-странице можно было использовать внешний скрипт, необходимо сказать браузеру, где его искать.

Для этого используется атрибут `src` тега `script`, подобным образом:

```
<html>
   <head>
      <title>Page Title</title>
      <script language="JavaScript" src="../script/myscript.js">
      </script>
   </head>
   <body>
      <!- Содержимое страницы размещается здесь. ->
   </body>
</html>
```

Атрибут `src` задает путь от текущей страницы к файлу со скриптами, который вы желаете присоединить, наподобие гиперссылки.

Присоединение таблиц стилей

Здесь есть маленькая хитрость. У тега `style` нет атрибута `src`, поэтому нельзя использовать тег `style` для присоединения внешней таблицы стилей. Вместо этого используется тег `link`:

```
<html>
   <head>
      <title>Page Title</title>
      <link   href="css/mystyles.css"   rel="stylesheet"   type=
   "text/css">
   </head>
   <body>
      <!- Содержимое страницы размещается здесь. ->
   </body>
</html>
```

Совет

Другой альтернативой является импортирование внешней таблицы стилей. За дополнительной информацией обратитесь к главе 42.

Обратите внимание на пару вопросов, касающихся тега `link`. У этого тега также нет атрибута `src`, однако есть атрибут `href`, который работает точно так же. В нем указывается путь от текущей страницы к внешней таблице стилей. Значение атрибута `rel` описывает отношение связанного документа, а значение атрибута `type` сообщает тип таблицы стилей, находящейся в присоединенном документе. Последние два атрибута не являются критически важными, т.к. для присоединения таблицы стилей на самом деле необходим только атрибут `href`, однако эти атрибуты могут быть полезны для программного или аппаратного обеспечения, читающего Web-страницу, поэтому следует попытаться включить и их.

Совет

*Тег **link** всегда должен размещаться в разделе **head** документа HTML. Его невозможно разместить в каком-либо другом месте.*

Важно отметить, что для тега `link` не требуется закрывающая версия, `</link>`. Тег `link` является самодостаточным, а тег `script` – нет. Это означает, что после открытия тега `script` его всегда необходимо стараться закрывать, даже если между тегами не размещен никакой JavaScript-код. А с тегом `link` такой проблемы не возникает.

Дневник: Внедрение или присоединение?

В нашем распоряжении есть два метода: внедрение и присоединение. Какой из них лучше?

Ответ – присоединение. Несомненно.

Мне нравится присоединять скрипты и таблицы стилей. Этот подход позволяет использовать один и тот же файл со скриптами или таблицей стилей в любом желаемом количестве Web-страниц, что бесценно для часто используемых функций, например, ролловеров и всплывающих меню, и стилей, используемых по всему сайту. Если бы я попытался внедрить их, мне бы пришлось копировать и вставлять одни и те же блоки кода снова и снова в каждую страницу, где требуются данные скрипты или таблица стилей. Мне это кажется пустой тратой времени, если не сказать больше. Браузер посетителя вынужден загружать внедренный код снова и снова в то время, когда присоединенные файлы браузер кэширует.

И что даже более важно, если в процессе разработки я захочу изменить что-либо, мне придется поправить это только в одном месте – в присоединяемом скрипте или таблице стилей. Если бы код был внедрен, необходимо было бы проходить по всем экземплярам кода и вносить одинаковые исправления снова и снова. Еще одна пустая трата времени.

Более того, для тех, кто стремится к стандартизации, консорциум W3C рекомендует присоединять код, а не внедрять.

Вывод: внедрение отлично подходит для простых одноразовых функций и стилей. Для всего остального, что, возможно, будет использовано повторно, присоединение является единственным разумным способом.

ГЛАВА 13.
Запуск скриптов при помощи ссылок

В старые добрые времена простая гиперссылка выполняла строго определенную задачу: навигацию. Посетитель щелкнул на ссылке и перешел куда-то. Это было так просто.

Даже сегодня навигация остается основным назначением гиперссылок. Однако в Интернете посетители желают делать больше, чем просто переходить от одной страницы к другой. Они хотят видеть функциональность. Они хотят скрипты и другие интерактивные штучки. Они хотят программные мини-решения.

Когда вы приметесь разрабатывать такие типы интерфейсов, то вскоре поймете, что что-то на странице должно запускать все эти заказные функции. Логическим выбором является что-либо, на чем можно щелкнуть. Вариантом являются кнопки HTML-формы. Также подходят и изображения, похожие на кнопки. Подходят также и гиперссылки. Когда вы бродите по Интернету, щелчок на том, что похоже на гиперссылку, может не иметь ничего общего с навигацией. Вместо этого может быть запущен скрипт.

Совет

Поскольку гиперссылки для навигации и гиперссылки для запуска скриптов на странице выглядят одинаково, их нельзя отличить, полагаясь только на внешний вид. Вместо этого используйте другие ухищрения, чтобы передать назначение ссылки вашим посетителям. Такие факторы, как текст ссылки и ее размещение относительно других элементов страницы, помогают посетителям понять, что произойдет, когда они щелкнут на ссылке.

С точки зрения удобства использования, запуск скриптов при помощи гиперссылок является рискованным делом из-за потенциальных проблем, возникающих на практике. Если вы не решите, как отличать гиперссылки для запуска скриптов от гиперссылок для навигации, это может очень просто сбить с толку ваших посетителей и расстроить их надежды относительно того, что произойдет после щелчка на ссылке. С другой стороны, текстовые гиперссылки являются более привлекательными и скромными, чем большинство графических кнопок, и они ни капли не похожи на грубые и отталкивающие кнопки HTML-формы. Если использовать гиперссылки, запускающие скрипты, в правильном контексте, они улучшат общий вид и впечатление от вашего интерфейса, который никогда не будет плохим. Одним из способов запуска скрипта является создание ссылки, ведущей в никуда, и присвоение результата событию ссылки onClick, наподобие следующего кода:

```
<a href="#" onClick="doScript();">Launch the script</a>
```

Ну хорошо, с технической точки зрения эта ссылка ведет куда-то. Знак нумерации, формирующий значение атрибута `href`, ссылается на текущую страницу, поэтому щелчок на этой ссылке аналогичен щелчку на кнопке **Refresh** (Обновить) браузера.

Зачем нужна самоотносящаяся ссылка? Обычный HTML-текст не реагирует на событие `onClick`, а гиперссылка – реагирует. Посетителю необходимо предоставить

что-нибудь, на чем можно щелкнуть, поэтому указывается ссылка, однако мы не хотим, чтобы гиперссылка вела куда-либо. Ссылка, которая, по существу, обновляет страницу, выглядит, как безобидный компромисс.

Такая стратегия работает достаточно хорошо, однако она может доставить неудобства, особенно если посетитель вынужден пролистывать страницу, чтобы найти ссылку. Щелчок на самоотносящейся гиперссылке заставляет браузер перепрыгнуть обратно наверх страницы, так что посетители теряют текущее место в тексте.

Лучшей альтернативой является указание скрипта в значении самого атрибута `href`, наподобие следующего кода:

```
<a href="javascript:doScript();">Launch the script</a>
```

Выполнив это, вы избавитесь от самоотносящейся ссылки, поэтому браузер не будет перепрыгивать по странице, когда посетитель щелкнет на этой ссылке. Этот способ лучше для всех. Просто не забывайте указывать фразу `javascript:` перед JavaScript-кодом в значении атрибута `href`. В противном случае браузер предположит, что значение атрибута `href` является путем к странице.

Дневник: JavaScript-ссылки или onClick-ссылки?

JavaScript-ссылки, или ссылки со словом `javascript` в значении атрибута `href`, вызывают проблемы у посетителей, которые либо отключили поддержку языка JavaScript в своем браузере, либо совсем ее не имеют. Боюсь показаться бесчувственным, но почти во всех случаях количество таких посетителей мало.

В качестве компромисса всегда можно использовать самоотносящиеся ссылки, реагирующие на события `onClick`. Такие ссылки проще обрабатывать браузерам, не имеющим поддержки языка JavaScript, но у вас появится проблема с перескакиванием браузера обратно наверх страницы. Это будет мешать большей части ваших посетителей, а не помогать.

Если вы действительно хотите оказать внимание пуристам языка HTML, не используйте язык JavaScript на своем сайте. Однако если вы хотите иметь или вам необходима дополнительная функциональность, JavaScript-ссылки для большинства посетителей работают лучше, чем onClick-ссылки.

ЧАСТЬ 2.
Вопросы, касающиеся макетов

- Создание макетов с фиксированной шириной при помощи таблиц
- Дневник: для создания макета использовать HTML- или CSS-код?
- Создание макетов с фиксированной шириной при помощи таблиц CSS
- Центрирование CSS-макетов с фиксированной шириной
- Создание текучих макетов при помощи таблиц
- Создание текучих макетов при помощи таблиц CSS
- Разделение изображений на срезы для макетных таблиц
- Создание многоколонных макетов при помощи таблиц
- Создание многоколонных макетов при помощи таблиц CSS
- Предотвращение разрыва навигационной панели
- Раскрашивание ячеек таблицы
- Раскрашивание и форматирование CSS-разделов
- Управление пространством дизайна в языке HTML
- Управление пространством дизайна при помощи таблиц CSS

Глава 14.
Создание макетов с фиксированной шириной при помощи таблиц

В макетах с **фиксированной шириной** используются абсолютные размеры в пикселах для определения ширины страницы. Противоположностью макетам с фиксированной шириной являются **текучие** макеты и **автоматически изменяющиеся** макеты, которые автоматически изменяют свой размер, чтобы умещаться по ширине окна браузера посетителя.

Определение

Макеты с фиксированной шириной используют абсолютные размеры в пикселах для определения ширины страницы.

Макеты с фиксированной шириной являются единственным выбором для страниц, требующих точного контроля над графическим дизайном. Нельзя ожидать от размещения визуальных элементов какой-либо точности, когда текучий макет изменяет ширину страницы, чтобы соответствовать ширине окна браузера. В то же время фиксированная ширина означает фиксированную ширину независимо от того, совпадает размер окна браузера с реальным размером страницы или значительно превышает его. На рисунке 14.1 продемонстрировано, как макеты с фиксированной шириной могут одновременно выглядеть сдавленными и потерянными в межзвездном пространстве, когда размеры окна браузера слишком велики по сравнению с размерами страницы. Выравнивание макета с фиксированной шириной по центру сокращает этот эффект, как показано на рисунке 14.2.

Часто задаваемые вопросы

Где я могу найти сайты, использующие макеты с фиксированной шириной?

К известным сайтам, использующим макеты с фиксированной шириной, относятся ***nytimes.com*** *и* ***yahoo.com****. Оба сайта укладывают огромное количество информации в тщательно разработанные макеты, требующие точности фиксированной ширины.*

Разработка всех хороших макетов для Интернета начинается с эскиза, похожего на представленный на рисунке 14.3, на котором размечаются основные области страницы. Не создавайте слишком ухищренный дизайн. Атрибуты `colspan` и `rowspan` языка HTML позволяют строить чрезвычайно сложные таблицы, однако делайте все возможное, чтобы избегать их. Объединения столбцов и строк отлично подходят для таблиц данных или таблиц, представляющих строки и столбцы данных в соответствии со спецификацией языка HTML. Таблицы макета, которые вы создаете здесь, не соответствуют спецификации, и браузеры стремятся выполнить враждебные действия по отношению к дизайну, использующему объединение столбцов и строк.

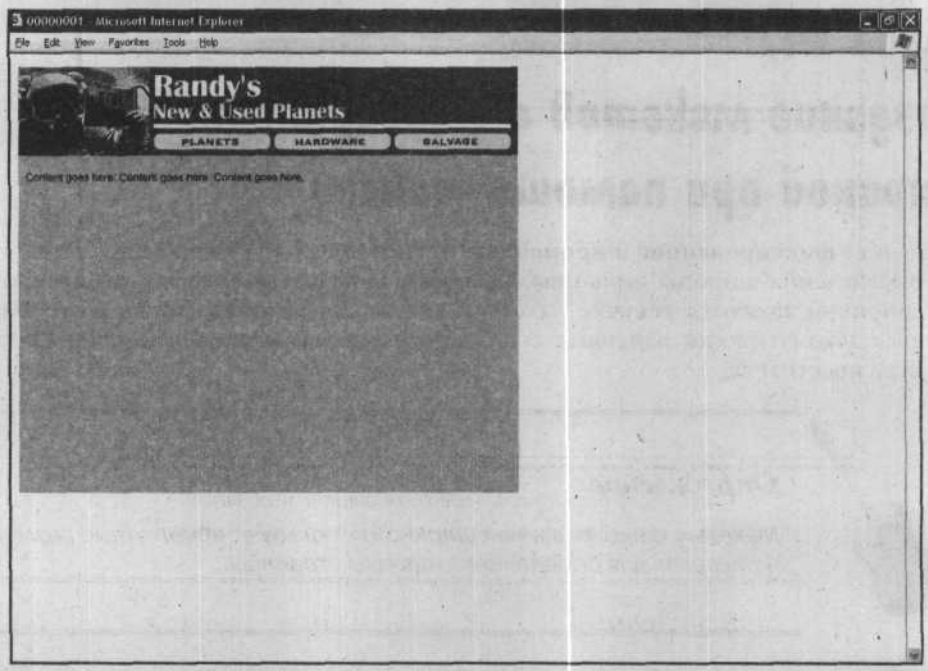


Рис. 14.1. Страница с фиксированной шириной может выглядеть ужатой, если ее просматривать в окне браузера, ширина которого значительно превышает ширину страницы

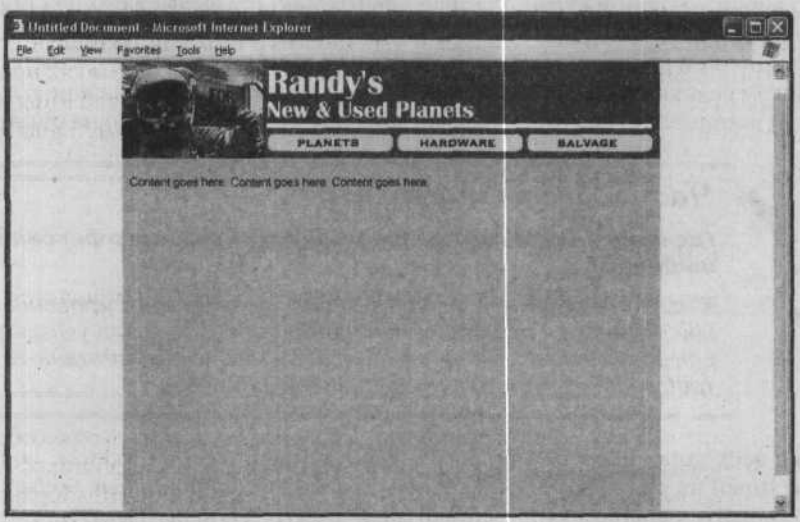


Рис. 14.2. Выравнивание макета с фиксированной шириной по центру улучшает положение вещей

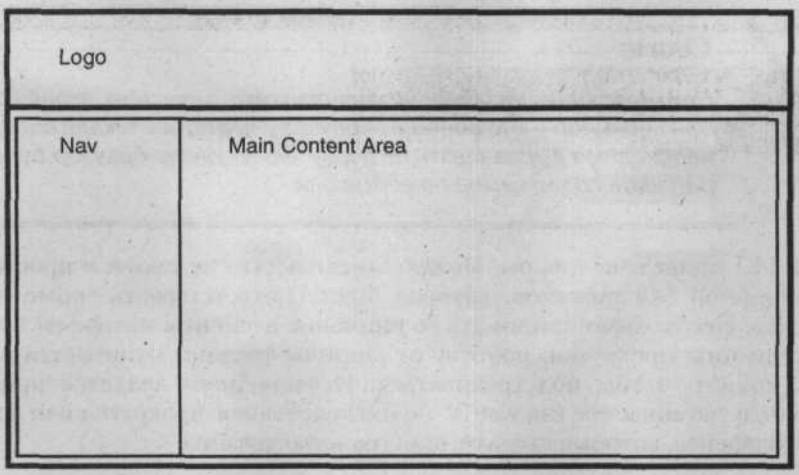


Рис. 14.3. Начните с создания эскиза макета, который вы желаете построить, и пометьте области

Определение

Таблица данных – это таблица, представляющая строки и столбцы данных. Это единственный тип таблиц, который официально поддерживается языком HTML. Использование таблиц для создания макетов, хотя и чрезвычайно распространено, противоречит правилам.

Если вы желаете получить более авантюрный дизайн, попытайтесь использовать **вложенные таблицы**. Вложенная таблица – это таблица, которая размещается внутри ячейки другой таблицы. Дизайн, представленный на рисунке 14.3, никак не назовешь авантюрным даже с натяжкой, однако в нижней ячейке основной таблицы вкладывается другая таблица, чтобы избежать объединения столбцов в верхней ячейке.

Определение

Вложенная таблица – это таблица, которая размещается внутри ячейки другой таблицы.

Как только набросок дизайна будет готов, вы должны начать думать о размерах в пикселах. Пиксел является основной единицей измерений в HTML-таблицах. Довольно удобно, что операционные системы компьютеров измеряют ширину экрана также в пикселах, и это делает процесс задания размеров макета для конкретных настроек экрана безболезненной операцией. Вы просто выбираете ширину экрана, которая имеет смысл для ваших посетителей, а затем создаете макет, который вмещается в таблицу аналогичных размеров.

Совет

При создании дизайна устанавливайте значение атрибута ячеек `valign` *в «top». Это сообщит браузеру, что по вертикали содержимое необходимо выравнивать по верху ячеек. Иначе браузер будет центрировать содержимое по вертикали.*

В таблице 14.1 приведены цифры. Можно заметить, что не так уж и просто создать таблицу шириной 640 пикселов, которая будет соответствовать экрану шириной 640 пикселов. Необходимо принимать во внимание элементы интерфейса браузера, например, полосы прокрутки, поэтому от ширины таблицы отнимается несколько пикселов, просто чтобы подстраховаться. Исключением является приложение MSNTV, иногда упоминаемое как WebTV, не имеющее полос прокрутки или других элементов интерфейса, которые съедают пространство дизайна.

Табл. 14.1. Распространенные ширины экрана и соответствующие им ширины таблицы

Ширина экрана	Ширина таблицы	Комментарии
544 пикселов	544 пикселов	Приложение WebTV/MSNTV
640 пикселов	600 пикселов	Абсолютный минимум для большинства приложений
800 пикселов	760 пикселов	Стандартный безопасный размер
1024 пикселов	955 пикселов	Выходящий за рамки стандартного размер
1280 пикселов	1210 пикселов	Не рекомендуется для универсальных сайтов
1600 пикселов	1530 пикселов	Не рекомендуется для универсальных сайтов

Совет

И хотя в действительности многие бродят по Интернету с использованием настольных мониторов и экранов ноутбуков, разрешение которых установлено в 1024 пикселов по ширине, не забывайте, что также существуют и другие типы устройств просмотра для Интернета, например карманные компьютеры (PDA), телевизоры, телефоны, консоли видеоигр, кухонные приспособления, наручные часы, и кто знает, что еще. Если ваш сайт обращается к беспроводной публике, откажитесь от больших размеров экрана.

Для конечной цели данного примера не будем рисковать и выберем экран шириной 800 пикселов, для которого ширина таблицы составляет 760 пикселов. Теперь, когда мы знаем общую ширину таблицы, рассчитать ширины областей, расположенных внутри таблицы, не составит труда. Смотрите. Сделайте свой наилучший выбор. Позднее, при необходимости, можно будет изменить значения. Просто удостоверьтесь, что сумма отдельных ширин не больше (или не меньше) общей ширины таблицы.

Нанесите значения на эскиз, как показано на рисунке 14.4, и код таблицы практически напишет себя сам. Замените метки в HTML-коде содержимым сайта – используя вложенные таблицы, если необходим особый дизайн, – и у нас готов макет с фиксированной шириной.

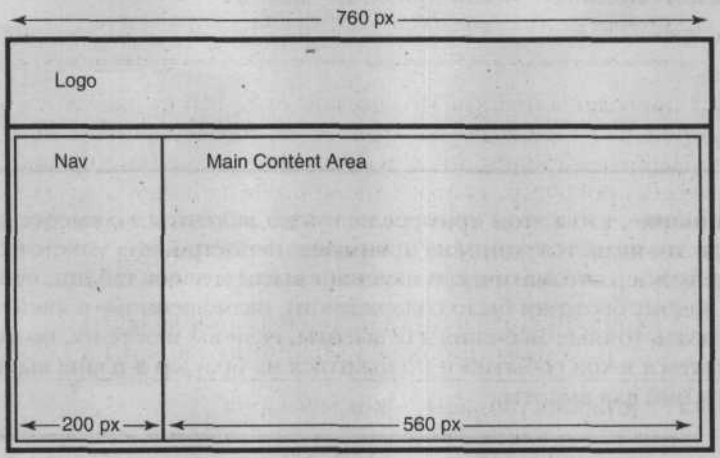


Рис. 14.4. После определения ширины таблицы остальные значения становятся на свои места

Совет

Не забудьте установить значения атрибутов таблицы макета `border`, `cellpadding` *и* `cellspacing` *в 0. В данном случае, браузер не будет автоматически добавлять дополнительные промежутки, как он делает это при отображении таблиц.*

Листинг 14.1. Просмотр исходного кода для рисунка 14.4

```
<div align="center">
<!- Если вы не желаете выравнивать таблицу макета по центру, удалите тег div здесь и в конце листинга. ->
   <table width="760" border="0" cellpadding="0" cellspacing="0">
      <tr>
         <td width="760" valign="top">Logo</td>
      </tr>
      <tr>
         <td width="760" valign="top">
<!- Вложенная таблица начинается здесь. ->
            <table width="760" border="0" cellpadding="0" cellspacing="0">
               <tr>
                  <td width="200" valign="top">Nav</td>
```

```
                <td width="560" valign="top">Content</td>
            </tr>
        </table>
<!- Вложенная таблица заканчивается здесь. ->
        </td>
    </tr>
  </table>
</div>
```

Обратите внимание, что в этом примере не нужно заботиться о высоте ячеек таблиц. На самом деле это является одним из преимуществ построения макетов с использованием таблиц. Браузер автоматически изменяет высоты ячеек таблиц, чтобы они соответствовали какому бы то ни было содержимому, размещенному в ячейках. Безусловно, можно указать точные значения для высоты, если вы знаете их, но можно и просто не вмешиваться в ход событий и положиться на браузер в плане вычисления правильных значений для высоты.

Совет

Если ширина ячейки таблицы равняется ширине самой таблицы, атрибут `width` *ячейки таблицы можно опустить. В данном случае, если вы измените свое мнение относительно ширины таблицы, вам придется обновить значение всего лишь одного атрибута* `width`.

Вот как мог бы выглядеть листинг после добавления нескольких значений для атрибутов `height`:

```
<div align="center">
<table width="760" border="0" cellpadding="0" cellspacing="0">
    <tr>
        <td width="760" height="100" valign="top">Logo</td>
    </tr>
    <tr>
        <td width="760" height="400" valign="top">
<!- Вложенная таблица начинается здесь. ->
            <table width="760" border="0" cellpadding="0" cellspac-
        ing="0">
                <tr>
                    <td   width="200"   height="400"   valign="top"
                >Nav</td>
                    <td   width="560"   height="400"   valign="top"
                >Content</td>
                </tr>
            </table>
```

```
<!- Вложенная таблица заканчивается здесь. ->
          </td>
        </tr>
    </table>
</div>
```

Обратной стороной отказа от использования атрибутов `height` является то, что в окне браузера таблица будет выглядеть полностью свернутой до тех пор, пока вы не начнете добавлять содержимое в ячейки. Если вы считаете это ошибкой, при построении таблицы просто задайте атрибутам ячеек `height` произвольные значения, однако потом, когда содержимое будет находиться в ячейках, не забудьте удалить атрибуты `height`.

Дневник: для создания макета использовать HTML- или CSS-код?

Существует два основных подхода по созданию макетов на Web-странице: HTML-таблицы и каскадные таблицы стилей (Cascading Style Sheets – CSS). Подход, использующий таблицы, является классическим. Его легко реализовать, и он надежно работает на множестве различных типов браузеров. Однако использование HTML-таблиц для создания макета приводит все стандарты в беспорядок. Группы, наподобие консорциума World Wide Web (W3C), ненавидят, когда таблицы используются подобным образом, поскольку использование тегов `table` языка HTML предполагается для строк и столбцов данных. Использование таблиц для графического дизайна напоминает использование молота для вбивания шурупа – задача будет выполнена, однако это не та задача, на которую рассчитывали молот или шуруп. Этот тип аргумента не является всего лишь придиркой к мелочам. Средства специальных возможностей, например, преобразователи текст-речь, и читатели экрана полагаются на точные интерпретации стандартов языка HTML, поэтому ваш макет, в основе которого лежат таблицы, может с большой легкостью ввести в заблуждение инвалидов по зрению. Однако несмотря на эти ограничения, макеты, основанные на использовании таблиц, доминируют в Интернете.

Альтернативный подход с использованием CSS-кода следует стандартам буквально, удовлетворяя даже самых привередливых членов консорциума W3C. Таблицы CSS уже некоторое время используются повсеместно, но до сих пор они являются «созревающей» технологией. Таблицы стилей не были бы такими неудачными, если бы браузеры выполняли совместные действия в отношении их. Самые последние версии браузеров Internet Explorer, Netscape и Opera достаточно хорошо поддерживают таблицы CSS, однако ни один из них не поддерживает таблицы CSS полностью и без существенных несогласованностей. Для построения макета, основанного на таблицах CSS, требуется больше усилий и больше времени для тестирования, и, кажется, что данный подход наилучшим образом работает для простых дизайнов. Пока же макеты, основанные на таблицах CSS, или что-то очень похожее на них, являются будущим графического дизайна в Интернете.

Если вы пытаетесь охватить максимально возможное количество посетителей, выбор очевиден: используйте таблицы. Однако если вы заботитесь о стандартах и проблемах доступности людям с ограниченными возможностями и смотрите в будущее, таблицы CSS – это ваш выбор.

ГЛАВА 15.
Создание макетов с фиксированной шириной при помощи таблиц CSS

Для тех, кто только что присоединился к нам, напомним, что макет с фиксированной шириной использует абсолютные размеры для определения ширины страницы, и это дает более высокий уровень управления графическим дизайном. В главе 14 описывается, как создавать макеты с фиксированной шириной при помощи таблиц. В этой главе показано, как добиться того же результата при помощи каскадных таблиц стилей (Cascading Style Sheets – CSS).

Совет

В результате выполнения данной процедуры вы получите макет с фиксированной шириной, выровненный по левой стороне окна браузера. Если же вы хотите выровнять макет, основанный на таблицах CSS, по центру, сразу перейдите к главе 16.

Если в этой книге взять любую главу, посвященную макетам, основанным на таблицах, вы снова и снова будете читать о том, как следует всегда создавать дизайн с использованием вложенных таблиц. В этой главе вы не найдете таких советов, поскольку при использовании таблиц CSS необходимость во вложенных таблицах отпадает. Вместо этого макет делится на ряд прямоугольных областей, называемых **разделителями** (divs), или разделами. Каждая область дизайна разрабатывается по отдельности с использованием тегов `div` языка HTML, а затем эти элементы выкладываются на Web-странице, как части мозаики. В этом смысле таблицы CSS более ориентированы на графику, чем таблицы языка HTML. Мыслить в понятиях прямоугольников является абсолютно естественным для людей со зрительными способностями, которые в любом случае стремятся видеть мир в фигурах.

Определение

Тег `div` языка HTML обозначает раздел (division). Он размечает область или раздел Web-страницы. Каждый раздел может иметь свои собственные свойства CSS-стиля.

Как и в случае с макетами, основанными на таблицах, разработка макетов, основанных на таблицах CSS, начинается с эскизов, как показано на рисунке 15.1. Снова обращаем внимание, что при работе с таблицами CSS не нужно думать о вложенных таблицах. Необходимо просто определить различные прямоугольные области, составляющие страницу.

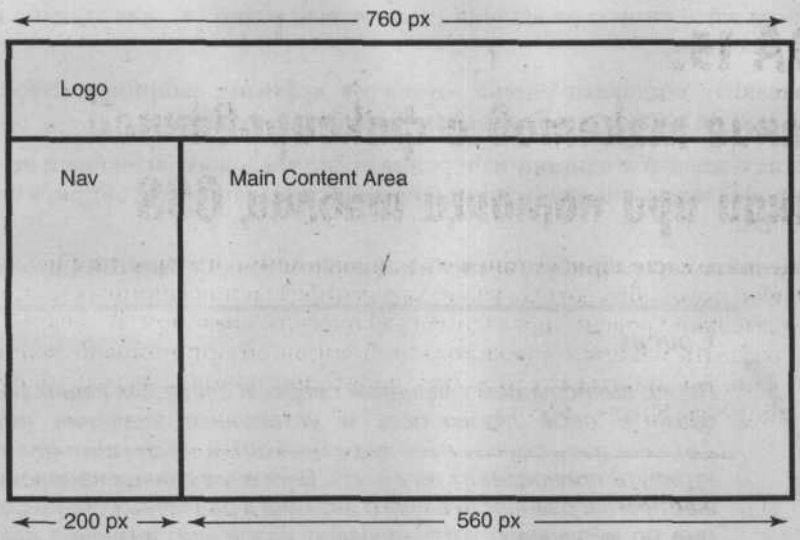


Рис. 15.1. Не называйте данные прямоугольники ячейками таблицы. Они являются разделами

Для создания этого макета потребуется три прямоугольника:

1. Один для области **Logo** (Логотип) шириной 760 пикселов;
2. Один для области **Nav** (Навигация) шириной 200 пикселов;
3. Один для области **Content** (Содержимое) шириной 560 пикселов.

Три прямоугольника подразумевают три элемента раздела. Вот они:

```
<div id="logo" style="position: absolute; width: 760px;">
   Logo
</div>

<div id="nav" style="position: absolute; width: 200px;">
   Nav
</div>

<div id="content" style="position: absolute; width: 560px;">
   Content
</div>
```

Значения атрибута `style` данных тегов задают ширину в CSS-формате, а не в виде `width="760"`, который можно было бы использовать при построении таблицы. Важно помнить о необходимости использования CSS-формата, вместо HTML-формата, для всего, что находится внутри атрибута `style`. Это означает следующее:

- Используйте символ двоеточия (:), а не знак равно (=), для задания пар атрибут/значение;
- Разделяйте составные имена атрибутов дефисом, например, `background-color`, а не `backgroundcolor` или `background_color`;
- Всегда указывайте единицу измерения после числового значения и не вставляйте пробел между числом и единицей измерения, например, `560px`, а не `560` или `560 px`;
- Разделяйте пары атрибут/значение символом точки с запятой (;).

Совет

Перед вычислением смещений сверху и слева для ваших разделов, окажите себе любезность и установите значения атрибутов `leftmargin`, `topmargin`, `marginwidth` и `marginheight` в 0. Эти атрибуты принадлежат тегу `body`. Внесение данных изменений улучшает согласованность вашего дизайна в различных браузерах, которые по умолчанию устанавливают различные значения для пространств полей страницы.

Если помните, теги `div` представляют разделы или области страницы. Следующим шагом является их размещение на странице. Для этого расположение элементов выражается через смещение от левой и верхней сторон окна браузера. Разместить первый элемент раздела не составит труда: он начинается в левом верхнем углу, поэтому у него нет смещения сверху или слева:

```
<div id="logo" style="position: absolute; width: 760px; top: 0px;
left: 0px;">
   Logo
</div>
```

Второй элемент раздела начинается слева, поэтому у него нет смещения слева, но как насчет смещения сверху? Здесь необходимо что-то указать. Иначе второй раздел частично перекроет первый раздел, что не предусмотрено вашим дизайном. Предположим, что вы планируете вставить графический логотип в первый раздел, и вы знаете, что высота изображения составляет 100 пикселов. Если второй раздел начинается прямо под логотипом, смещение будет составлять 100 пикселов, как показано на рисунке 15.2. Следовательно, код выглядит так:

```
<div id="nav" style="position: absolute; width: 200px; top: 100px;
left: 0px;">
   Nav
</div>
```

Для обеспечения согласованности и очень четкого разъяснения всего для браузеров, критично относящихся к таблицам CSS, следует вернуться обратно к разделу **Logo** (Логотип) и жестко закодировать значение атрибута `height`:

```
<div id="logo" style="position: absolute; width: 760px; height:
100px; top: 0px; left: 0px;">
```

```
   Logo
</div>
```

Для третьего элемента раздела необходимо смещение сверху, равное 100 пикселам, чтобы не закрывать область **Logo** (Логотип), и смещение слева, равное 200 пикселам, чтобы не закрывать область **Nav** (Навигация):

```
<div id="content" style="position: absolute; width: 560px; top:
100px; left: 200px;">
   Content
</div>
```

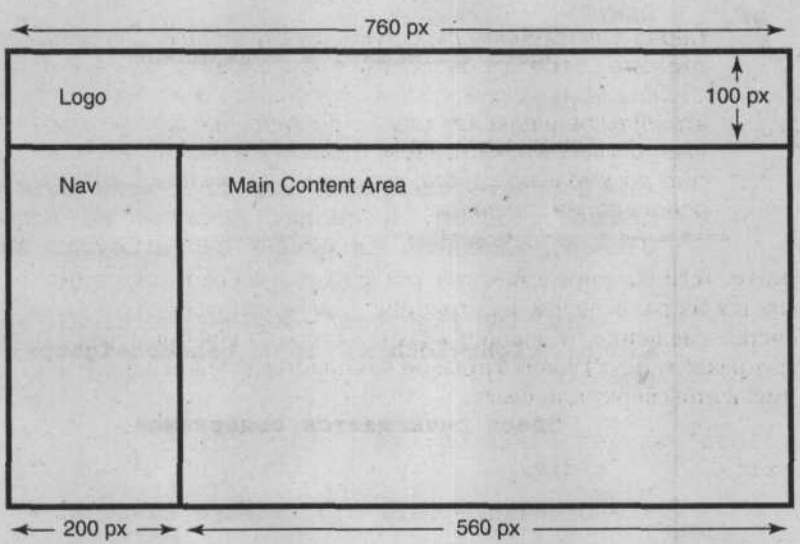


Рис. 15.2. Поскольку размещение каждого элемента раздела осуществляется в соответствии со смещениями сверху и слева, необходимо вычислить высоту элемента ***Logo*** *(Логотип). Всегда можно выбрать произвольное значение и уточнить его позднее*

Инструментарий

Макет с фиксированной шириной, построенный при помощи CSS-кода

Данный инструментарий предоставляет макет с фиксированной шириной, построенный при помощи CSS-кода, с любой комбинацией следующих элементов: баннер, левый столбец, средний столбец и правый столбец.

```
<body leftmargin="0" topmargin="0" marginwidth="0"
marginheight="0">

<!- Добавьте следующий тег div, если хотите, чтобы
по всему верху макета располагался баннер. ->
```

```
   <div id="banner" style="position: absolute;
width: designwidthpx; height: bannerheightpx; top:
0px; left: 0px;">Здесь размещается содержимое
   </div>
<!- Окончание тега div баннера ->
<!- Добавьте следующий тег div, если хотите иметь
левый столбец. Если вы не используете баннер, за-
мените ниже выражение bannerheight значением 0. ->
   <div id="leftcol" style="position: absolute;
width: leftwidthpx; top: bannerheightpx; left:
0px;">
      Здесь размещается содержимое
   </div>
<!- Окончание тега div левого столбца. ->
<!- Добавьте следующий тег div, если хотите иметь
правый столбец. И снова, если вы не используете
баннер, замените выражение bannerheight значением
0. ->
   <div id="rightcol" style="position: absolute;
width: rightwidthpx; top: bannerheightpx; right:
0px;">
      Здесь размещается содержимое
   </div>
<!- Окончание тега div правого столбца. ->
<!- Следующий тег div определяет основную область
для содержимого страницы. Замените нулями значе-
ния, которые вам не нужны. ->
   <div id="middlecol" style="position: absolute;
width: middlewidthpx; top: bannerheightpx; left:
leftwidthpx; right: rightwidthpx;">
      Здесь размещается содержимое
   </div>
<!- Окончание тега div среднего столбца. ->
</body>
```

ГЛАВА 16.
Центрирование CSS-макетов с фиксированной шириной

Хороший метод для центрирования CSS-макетов с фиксированной шириной заключается в использовании вложенных слоев, также известных как вложенные разделы. **Вложенный слой** – это элемент раздела внутри другого раздела, точно так же, как вложенные таблицы – это таблицы внутри таблиц. В данном сценарии основной раздел является контейнером. Вложенные внутри него слои представляют содержимое сайта. На рисунке 16.1 показан типичный CSS-макет, использующий вложенные слои.

Определение

***Вложенный слой**, или вложенный раздел, – это элемент раздела, который расположен внутри другого раздела.*

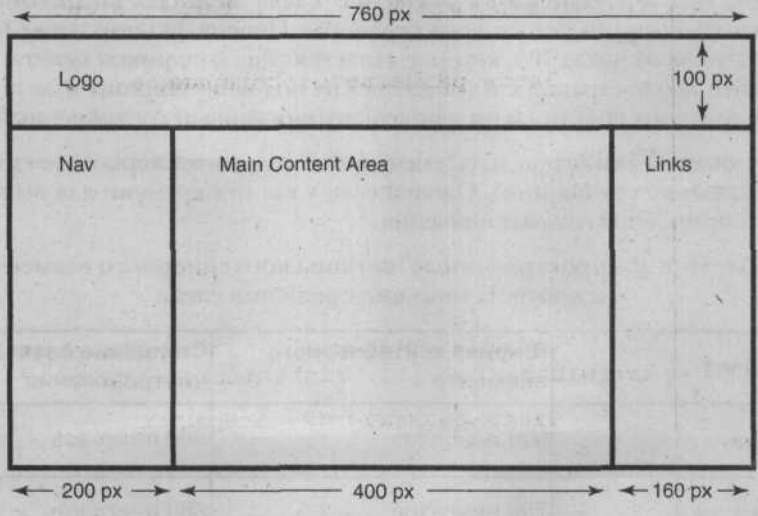


Рис. 16.1. Для построения CSS-макета, выровненного по центру, используйте контейнерный элемент, в котором будут размещаться вложенные элементы с содержимым, а затем отцентрируйте контейнер

Для этого макета требуется четыре вложенных раздела:

1. Один для области **Logo** (Логотип) шириной 760 пикселов;
2. Один для области **Nav** (Навигация) шириной 200 пикселов;

3. Один для области **Content** (Содержимое) шириной 400 пикселов;
4. Один для области **Links** (Ссылки) шириной 160 пикселов.

Также необходим контейнерный элемент, который будет содержать четыре перечисленных раздела.

Хитрость данного метода заключается в установке левого края страницы на центральную точку окна браузера, что достигается добавлением атрибута `style` к тегу `body` страницы:

```
<body style="margin-left: 50%;">
```

Обратите внимание, что используется значение, выраженное в процентах, а не абсолютный размер, выраженный в пикселах. Использование здесь процентного отношения является разумным решением, поскольку мы не знаем наверняка ширину окна браузера посетителя.

Теперь, когда левый край страницы начинается в центре окна браузера, необходимо выразить положение контейнерного элемента относительно этого края. Используйте этот код:

```
<div id="container" style="position: relative; width: 760px; left:
-380px;">
```

Смещение слева выражается отрицательным значением, и на то есть причина. Поскольку положение контейнерного элемента является относительным, а не абсолютным, браузер использует смещение слева немного по-другому, вычитая 380 пикселов от позиции левого края страницы. Почему 380 пикселов? Потому что 380 – это половина от числа 760, которое является общей шириной макета. По существу нужно найти центр страницы и сместиться на половину ширины макета в одну сторону. В результате мы получим идеально отцентрированный контейнерный элемент.

При использовании данного метода, смещение слева контейнерного элемента всегда равняется половине его ширины. Однако если у вас нет времени для вычислений, в таблице 16.1 приведены готовые значения.

Табл. 16.1. Распространенные ширины контейнерного элемента и соответствующие смещения слева

Ширина экрана	Ширина контейнерного элемента	Смещение слева для центрирования
640 пикселов	600 пикселов	300 пикселов
800 пикселов	760 пикселов	380 пикселов
1024 пикселов	954 пикселов	477 пикселов
1280 пикселов	1210 пикселов	605 пикселов
1600 пикселов	1530 пикселов	765 пикселов

Вложенные элементы выглядят так же, как элементы в главе 15. Их положение является абсолютным, однако поскольку они являются вложенными, браузер использует их родительский элемент – в данном случае, контейнерный элемент, – в качестве базисной точки для вычисления смещений слева и сверху, а не левый верхний угол окна браузера.

Вот исходный код для готового макета:

```
<body style="margin-left: 50%;">
<!- Атрибут style перемещает левый край страницы на центр окна браузера. ->
<!- Это контейнерный элемент. Отрицательное смещение слева задано относительно левого края страницы, определенного в теге body, и его значение равняется половине ширины. ->
   <div id="container" style="position: relative; width: 760px; left: -380px;">
<!- Здесь идут вложенные элементы. ->
      <div id="logo" style="position: absolute; width: 760px; top: 0px; left: 0px;">
         Logo
      </div>
      <div id="nav" style="position: absolute; width: 200px; top: 100px; left: 0px;">
         Nav
      </div>
      <div id="content" style="position: absolute; width: 400px; top: 100px; left: 200px;">
         Content
      </div>
      <div id="links" style="position: absolute; width: 160px; top: 100px; left: 600px;">
         Links
      </div>
<!- Перечисление вложенных элементов закончено. ->
   </div>
<!- Предыдущая строка закрывает контейнерный элемент. ->
</body>
```

ГЛАВА 17.
Создание текучих макетов при помощи таблиц

Текучие, или **автоматически изменяющиеся,** макеты – это макеты, которые автоматически изменяют свои размеры для соответствия размерам окна браузера посетителя. Сравните текучие макеты с их противоположностью – макетами с фиксированной шириной, использующими абсолютные размеры, например, 600 пикселов или 760 пикселов, независимо от размеров окна браузера, которые предпочитает конкретный посетитель.

Определение

***Текучие**, или **автоматически изменяющиеся**, макеты – это макеты, которые автоматически изменяют свои размеры для соответствия размерам окна браузера посетителя.*

Очевидное преимущество текучих макетов заключается в использовании всей доступной реальной площади экрана, благодаря чему они выглядят более изящно. Текучие макеты не выглядят слишком сдавленными при просмотре в очень узких окнах браузера, как иногда выглядят макеты с фиксированной шириной. Более того, посетители, имеющие большие мониторы и профессиональные видеокарты, могут максимизировать окна браузера и увидеть больше информации на сайте без необходимости прокрутки, а посетители с особыми требованиями вряд ли нарушат ваш макет, установив размер шрифта браузера в значение **Largest** (Самый крупный). Управление графическим дизайном в текучих макетах является гораздо более трудоемкой операцией. Что толку от того, что вы знаете, в каком именно пикселе появится конкретный элемент, если ширина области дизайна изменяется при изменении ширины окна браузера?

Часто задаваемые вопросы

Где я могу найти сайты, использующие текучие макеты?

Известные сайты, использующие текучие макеты, включают Amazon и большую часть eBay. Оба сайта содержат большое количество информации и имеют простой (можно даже сказать, скучный) графический дизайн.

Жизнь текучих макетов начинается с того же, что и жизнь макетов с фиксированной шириной. Это означает, что при создании текучего макета вы разрабатываете таблицу, благодаря которой макет умещается в окне браузера определенной минимальной ширины, скажем, 600 пикселов. Посетители, просматривающие страницы в окнах браузера, ширина которых меньше минимальной ширины макета, увидят крайне непривлекательные горизонтальные полосы прокрутки, даже в случае с текучим макетом.

Если вам не нужно ничего, кроме макета с фиксированной шириной, остановитесь здесь. Однако для текучего макета еще не все готово. Для создания эффекта текучести нужно очень аккуратно присвоить параметру `width` определенных ячеек таблиц символ звездочки (*), которое в языке HTML служит сокращением слова **любой**. Использование этой общей ширины, вместо фиксированного значения в пикселах, заставляет браузер вычислять ширину области дизайна, учитывая ширину окна браузера. Отсюда и текучесть.

Итак, ширина (параметр `width`) каких ячеек таблиц устанавливается **любой**? Это как раз тот случай, когда пригодится план и один или два эскиза.

Начните с исходного дизайна с фиксированной шириной, наподобие представленного на рисунке 17.1. Обратите внимание на простоту макета, что очень важно. В данном макете используются две вложенные таблицы внутри контейнерной, состоящей из двух ячеек. Можно добиться такого же результата при помощи одной таблицы, используя атрибут `colspan`, однако мы не хотим делать это в данном случае. Никогда, ни при каких обстоятельствах, не используйте атрибуты `rowspan` или `colspan` в текучем дизайне. Если вы используете эти атрибуты, при увеличении таблицы браузер посетителя осмелится вставить промежутки везде, особенно в тех местах, где вы не хотите, чтобы они появились.

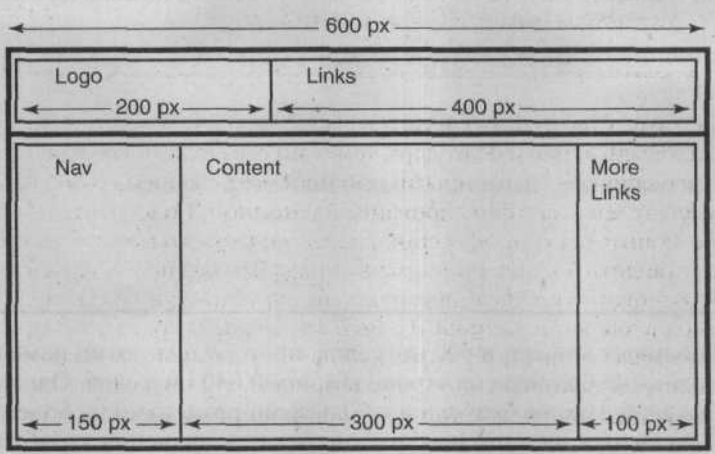


Рис. 17.1. Начните с исходного дизайна с фиксированной шириной – и поддерживайте его таким. Никогда не используйте объединение столбцов или строк для получения сложного дизайна. Вместо этого используйте вложенные таблицы

Листинг 17.1. Просмотр исходного кода для рисунка 17.1

```
<table width="600" border="0" cellpadding="0" cellspacing="0">
  <tr>
    <td>
<!- Первая вложенная таблица начинается здесь. ->
      <table width="600" border="0" cellpadding="0" cellspac-
    ing="0">
        <tr>
```

```
                <td width="200">Logo</td>
                <td width="400">Links</td>
            </tr>
        </table>
<!- Первая вложенная таблица заканчивается здесь. ->
    </td>
  </tr>
  <tr>
    <td>
<!- Вторая вложенная таблица начинается здесь. ->
        <table width="600" border="0" cellpadding="0" cellspac-
    ing="0">
            <tr>
                <td width="150">Nav</td>
                <td width="350">Content</td>
                <td width="100">More links</td>
            </tr>
        </table>
<!- Вторая вложенная таблица заканчивается здесь. ->
    </td>
  </tr>
</table>
```

Данная таблица имеет ширину в 600 пикселов, поэтому она уютно помещается в максимизированном окне браузера на экране шириной 640 пикселов. Однако она выглядит слишком сжатой в окнах браузеров с большими размерами на более широких экранах. Чтобы увеличить таблицу, необходимо решить, значения атрибутов `width` каких ячеек таблиц сменить с абсолютных, основанных на пикселах, на относительные, основанные на процентах.

Совет

При использовании данного метода создания текучести, выбирайте одну и только одну ячейку в строке таблицы, которая будет иметь переменную ширину. Не пытайтесь сделать две текучих ячейки в одной и той же строке. Более того, если таблица состоит из более, чем одной строки, соответствующая ячейка в каждой строке также должна быть текучей.

В первой вложенной таблице хорошим выбором является ячейка **Links** (Ссылки), поэтому ее атрибуту `width` присвоим символ звездочки, как показано на рисунке 17.2. Также установим значение атрибута `width` для всей таблицы в 100%. Если этого не сделать, ширина таблицы останется фиксированной, равной 600 пикселам, даже

и с текучей ячейкой.) В этом случае ширина ячейки **Logo** (Логотип) не станет больше, чем 200 пикселов, однако ячейка **Links** (Ссылки) увеличится и будет соответствовать ширине окна браузера. Если вместо этого увеличить ячейку **Logo** (Логотип), браузер сохранит ширину ячейки **Links** (Ссылки), равную 400 пикселам, и добавит свободное место внутри ячейки **Logo** (Логотип), что может разбалансировать макет.

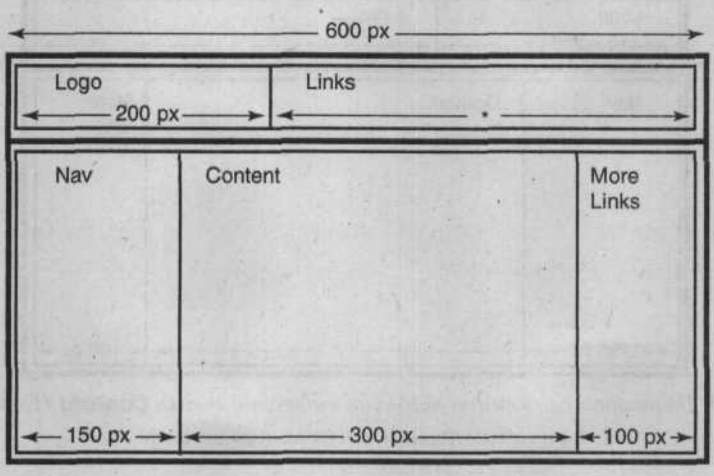


Рис. 17.2. В первой вложенной таблице ячейка ***Links*** *(Ссылки) является хорошим выбором для использования переменной ширины*

Листинг 17.2. Просмотр исходного кода для рисунка 17.2

```
<table width="100%" border="0" cellpadding="0" cellspacing="0">
   <tr>
      <td width="200">Logo</td>
      <td width="*">Links</td>
   </tr>
</table>
```

Во второй вложенной таблице ячейка **Content** (Содержимое) является наилучшим выбором для использования относительной ширины, как показано на рисунке 17.3. Ячейки, находящиеся по обе стороны от ячейки **Content** (Содержимое), будут иметь управляемые ширины, при этом позволяя содержимому страницы заполнять окно браузера естественным образом. Поэтому установите символ звездочки для атрибута `width` ячейки **Content** (Содержимое), а значение атрибута `width` вложенной таблицы установите равным 100%.

Совет

Хорошее правило выбора текучей ячейки заключается в выборе ячейки с наибольшим количеством содержимого.

Теперь просто установите значение атрибута `width` контейнерной таблицы в 100%, чтобы увеличить ее ширину до ширины окна браузера, и все готово. На рисунке 17.4 показан завершенный эскиз макета.

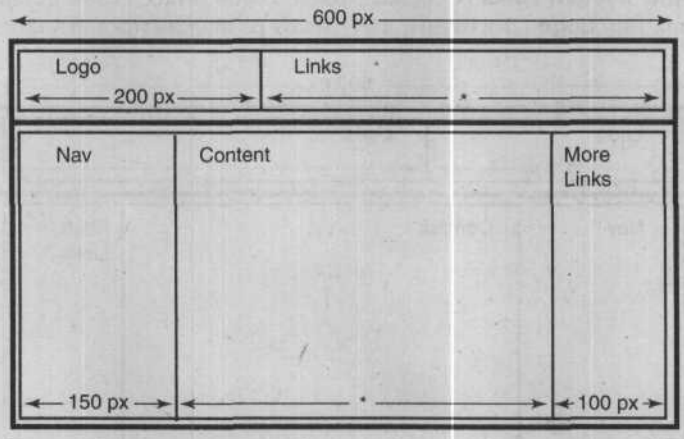


Рис. 17.3. Во второй вложенной таблице выберите ячейку **Content** *(Содержимое) в качестве ячейки с переменной шириной*

Листинг 17.3. Просмотр исходного кода для рисунка 17.3

```
<table width="100%" border="0" cellpadding="0" cellspacing="0">
   <tr>
      <td width="150">Nav</td>
      <td width="*">Content</td>
      <td width="100">More links</td>
   </tr>
</table>
```

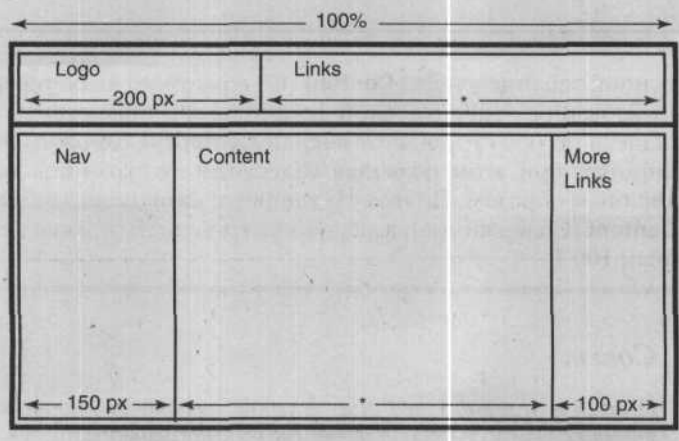


Рис. 17.4. Установите параметр `width` *контейнерной таблицы в значение 100%, и ваш макет станет текучим*

Листинг 17.4. Просмотр исходного кода для рисунка 17.4

```
<table width="100%" border="0" cellpadding="0" cellspacing="0">
  <tr>
    <td>
<!- Первая вложенная таблица начинается здесь. ->
      <table width="100%" border="0" cellpadding="0" cellspac-
    ing="0">
        <tr>
          <td width="200">Logo</td>
          <td width="*">Links</td>
        </tr>
      </table>
<!- Первая вложенная таблица заканчивается здесь. ->
    </td>
  </tr>
  <tr>
    <td>
<!- Вторая вложенная таблица начинается здесь. ->
      <table width="100%" border="0" cellpadding="0" cellspac-
    ing="0">
        <tr>
          <td width="150">Nav</td>
          <td width="*">Content</td>
          <td width="100">More links</td>
        </tr>
      </table>
<!- Вторая вложенная таблица заканчивается здесь. ->
    </td>
  </tr>
</table>
```

ГЛАВА 18.
Создание текучих макетов при помощи таблиц CSS

Как следует из главы 17, **текучие,** или **автоматически изменяющиеся,** макеты – это макеты, которые автоматически изменяют свои размеры для соответствия размерам окна браузера посетителя. Однако если в предыдущей главе рассматривались текучие макеты, созданные при помощи таблиц, то в этой главе демонстрируется, как создавать текучие макеты с использованием каскадных таблиц стилей (Cascading Style Sheets – CSS).

Совет

Как и в случае с CSS-макетами с фиксированной шириной, вам нужно установить атрибуты `leftmargin`, `topmargin`, `marginheight` *и* `marginwidth` *тега* `body` *в значение 0, чтобы математические расчеты корректно согласовывались между разными браузерами.*

Начните работу с создания эскиза макета, наподобие представленного на рисунке 18.1. Для CSS-макетов не требуется использование вложенных таблиц, поэтому не беспокойтесь об этом. Просто разделите дизайн на прямоугольные области. Каждая из этих областей станет элементом раздела в HTML-коде.

На следующем шаге необходимо решить, какие разделы будут текучими. Предположим, что мы хотим, чтобы раздел **Content** (Содержимое) увеличивался и уменьшался, но при этом левая и правая боковые панели имели фиксированную ширину, скажем, 200 пикселов для раздела **Nav** (Навигация) и 150 пикселов для раздела **Links** (Ссылки). Очевидно, что раздел **Content** (Содержимое) должен быть текучим, но как насчет раздела **Logo** (Логотип)? Если бы этот макет строился с использованием таблиц, нам не нужно было бы обязательно устанавливать значение атрибута `width` верхней ячейки, равным 100%, поскольку сама таблица имела бы ширину, равную 100%, и браузер изменил бы размеры ячейки **Logo** (Логотип) для полного соответствия ширине таблицы.

В таблицах CSS такого сокращения нет. Если вы желаете, чтобы раздел **Logo** (Логотип) растянулся вдоль всей ширины окна браузера, это необходимо явно указать в CSS-коде. Поэтому добавьте раздел **Logo** (Логотип) к все еще небольшому списку текучих элементов и разметьте эскиз, как показано на рисунке 18.2.

Одной из наименее желанных особенностей CSS-макетов является то, что не существует одного единственного решения для создания эффекта текучести. Необходимо использовать различные методы в зависимости от положения раздела в макете:

- Если раздел растягивается вдоль всей ширины окна браузера, эффект текучести создается посредством присвоения атрибуту раздела `width` значения 100%.
- Если справа или слева от раздела расположены другие элементы – раздел не растягивается на все окно браузера – не присваивайте значение атрибуту раздела `width`.

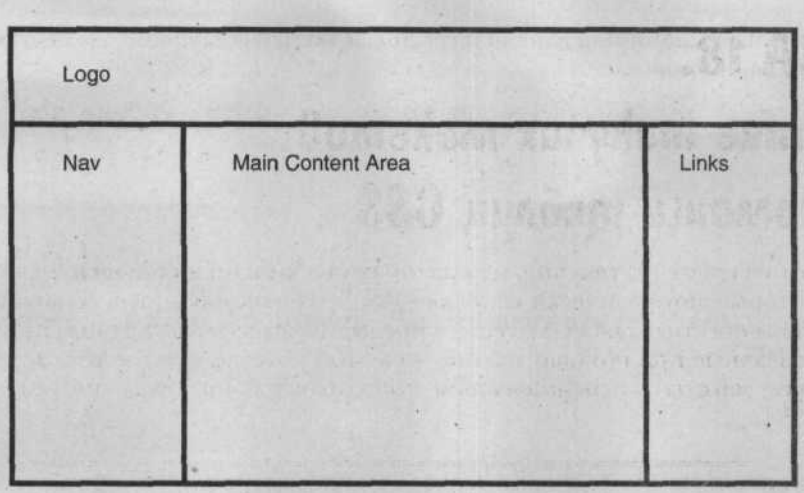


Рис. 18.1. Этот макет несложно создать при помощи таблиц CSS. Но как его сделать текучим?

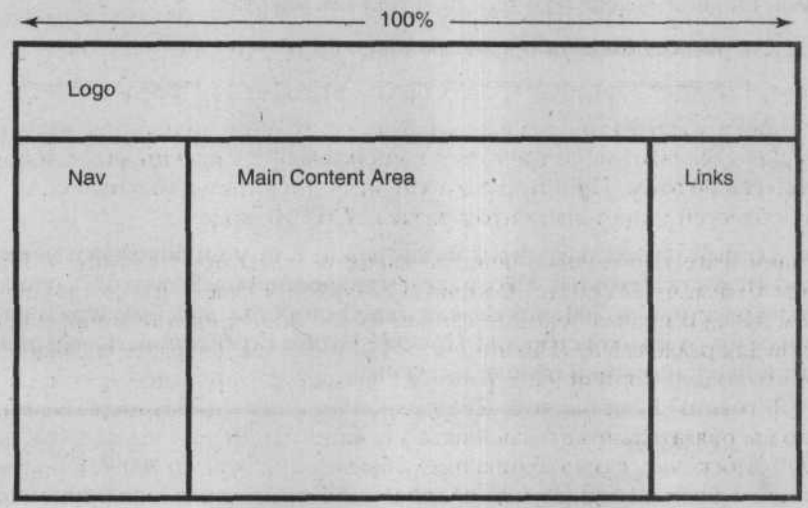


Рис. 18.2. Поскольку мы имеем дело с таблицами CSS, в этом макете необходимо сделать текучим раздел **Logo** *(Логотип). Если бы мы создавали таблицу, нам бы не пришлось этого делать*

Раздел **Logo** (Логотип) растягивается вдоль всей ширины браузера, от левой стороны до правой стороны, поэтому этот элемент попадает в первую категорию. Его код выглядит следующим образом:

```
<div id="logo" style="position: absolute; width: 100%; height:
100px; top: 0px; left: 0px;">
   Logo
</div>
```

Разделы для навигационной панели и ссылок являются обычными элементами с фиксированной шириной:

```
<div id="nav" style="position: absolute; width: 200px; top: 100px;
left: 0px;">
   Nav
</div>

<div id="links" style="position: absolute; width: 150px; top:
100px; right: 0px;">
   Links
</div>
```

В значение атрибута `style` раздела **Links** (Ссылки), помимо всего прочего, включена пара `right: 0px`. Этот особый кусочек кода размещает элемент рядом с правой стороной экрана. Атрибут `style` необходимо определять именно таким способом, поскольку мы не знаем, какую ширину будет иметь область **Content** (Содержимое), являющаяся самой главной, в первую очередь, в текучих макетах.

Теперь код для раздела **Content** (Содержимое):

```
<div id="content" style="position: absolute; top: 100px; left:
200px; right: 150px;">
   Content
</div>
```

Поскольку данный раздел находится между левой и правой боковыми панелями, он попадает во вторую категорию. Мы не задаем значение атрибута `width`, однако располагаем его на расстоянии 200 пикселов от левой стороны окна браузера и на расстоянии 150 пикселов от правой стороны. Почему? Чтобы гарантировать, что он не перекроет области навигационной панели и ссылок.

Совет

Если вы назначаете разделам фоновые цвета, необходимо сделать так, чтобы основная область с содержимым имела такой же цвет фона, как и страница. Это предотвратит отображение браузером, так сказать, швов дизайна. Раздел ***Content*** *(Содержимое) может не прилегать точно к правому столбцу в зависимости от того, что находится в основной области содержимого, и мы не хотим, чтобы цвет фона страницы проступил через пространство между разделами.*

Вот и он: текучий макет, созданный с использованием таблиц CSS. На рисунке 18.3 показан завершенный эскиз макета.

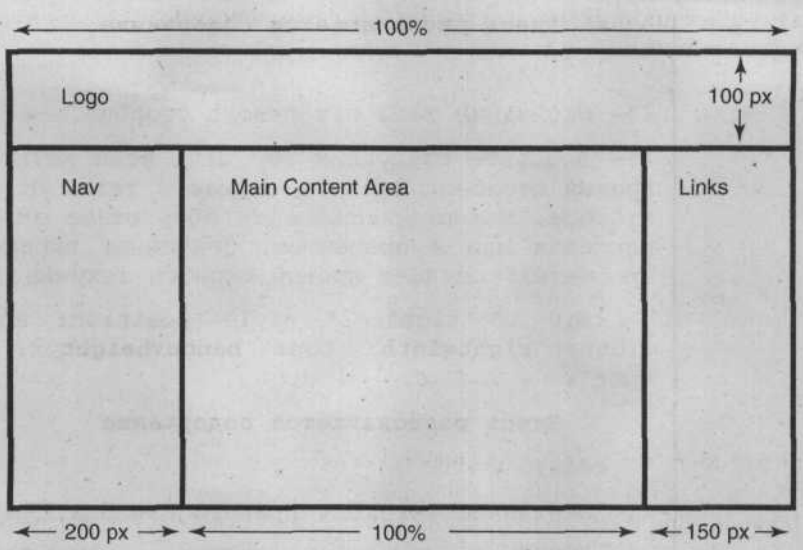


Рис. 18.3. Завершенный эскиз для текучего макета, создаваемого при помощи таблиц CSS

Инструментарий

Создание текущего макета при помощи таблиц CSS

Данный инструментарий демонстрирует, как создавать текучий CSS-макет с любой комбинацией следующих элементов: баннер, левый столбец, средний столбец и правый столбец.

```
<body leftmargin="0" topmargin="0" marginheight="0" marginwidth="0">

<!- Добавьте следующий тег div, если хотите, чтобы по всему верху макета располагался баннер. ->

   <div id="banner" style="position: absolute; width: 100%; height: bannerheightpx; top: 0px; left: 0px;">

      Здесь располагается содержание

   </div>

<!- Окончание тега div баннера. ->

<!- Добавьте следующий тег div, если хотите иметь левый столбец. Используйте значение в пикселах для атрибута width, чтобы получить столбец с фиксированной шириной, или задайте значение в процентах для текучего столбца. Если вы не используете баннер, замените ниже выражение bannerheight значением 0. ->

   <div id="leftcol" style="position: absolute; width: leftwidth; top: bannerheightpx; left: 0px;">
```

```
      Здесь располагается содержание
   </div>
<!- Окончание тега div левого столбца. ->
<!- Добавьте следующий тег div, если хотите иметь
правый столбец. Как и в случае с тегом div левого
столбца, можно использовать абсолютное значение в
пикселах или в процентах. Значение, выраженное в
процентах, сделает правый столбец текучим. ->
   <div id="rightcol" style="position: absolute;
width: rightwidth; top: bannerheightpx; right:
0px;">
      Здесь располагается содержание
   </div>
<!- Окончание тега div правого столбца. ->
<!- Следующий тег div определяет текучий столбец
для основного содержимого страницы. Замените нуля-
ми значения, которые вам не нужны. ->
   <div id="middlecol" style="position: absolute;
top: bannerheightpx; left: leftwidth; right:
rightwidth;">
      Здесь располагается содержание
   </div>
<!- Окончание тега div среднего столбца. ->
</body>
```

ГЛАВА 19.
Разделение изображений на срезы для макетных таблиц

Популярным методом создания Web-макетов является метод **разделения на срезы**. При использовании данного метода макет разрабатывается в виде графического файла в таких приложениях, как, например, Macromedia Fireworks, Adobe Photoshop или Adobe ImageReady. Затем файл буквально разделяется на кусочки для различных частей страницы, как показано на рисунке 19.1, и приложение для работы с графикой генерирует HTML-таблицу, которая снова собирает полученные кусочки в Web-браузере. Это является отличным решением для людей, обученных графическому дизайну: разработка ведется в полностью визуальной среде, и нет необходимости самому выполнять какое-либо кодирование.

Определение

Разделение на срезы – это метод создания макетных таблиц в графических приложениях, например, Fireworks и Photoshop. Сначала макет разрабатывается в графическом редакторе, после чего полученный макет разрезается на кусочки или срезы, основанные на различных частях страницы. При экспортировании срезов в виде отдельных графических файлов вы получаете HTML-таблицу, которая снова собирает полученные кусочки в Web-браузере.

По крайней мере, это – теория. Чтобы выполнить работу по разделению на срезы в практическом смысле, необходимо знать кое-что о создании макетных таблиц трудным способом, начиная с языка HTML, а не с графического файла. В этой книге заведено предостерегать вас от дурной привычки использовать объединение столбцов и строк в макетных таблицах, и макетные таблицы, полученные из разрезанных графических файлов, не являются исключением. Если в результате разделения на срезы получена макетная таблица, использующая объединения столбцов или строк (что, кстати, получилось при разделении на срезы рисунка 19.1), почти наверняка у вас появятся проблемы при добавлении содержимого к макету в HTML-редакторе. Содержимое, выходящее за границы таблицы, заставляет макет некрасиво разрываться, как показано на рисунке 19.2.

Избежать использования объединения столбцов и рядов достаточно легко, если каждую область страницы разделять на срезы поодиночке. Разделите исходные срезы, показанные на рисунке 19.1, на отдельные графические файлы, как на рисунке 19.3, а затем разделите на срезы и экспортируйте эти макеты по отдельности. В данном случае вы получите четыре отдельных макетных таблицы для четырех областей страницы.

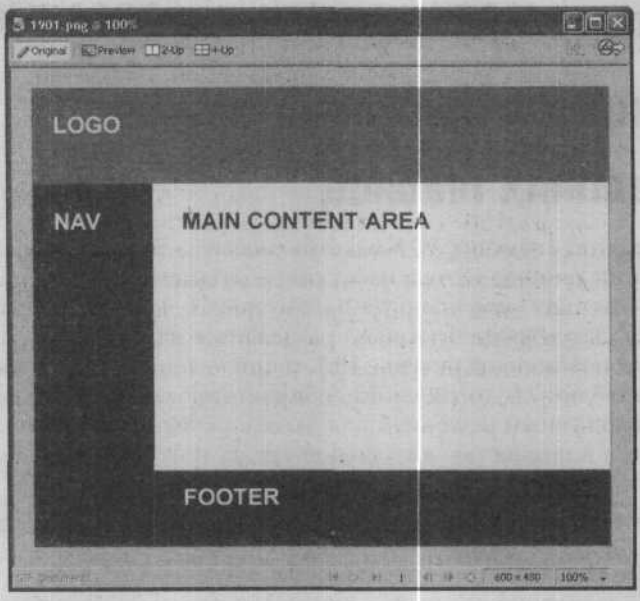


Рис. 19.1. В результате разделения макета на срезы в графическом приложении, например, Macromedia Fireworks, получаются отдельные графические файлы и HTML-таблица, которая снова собирает полученные срезы в Web-браузере

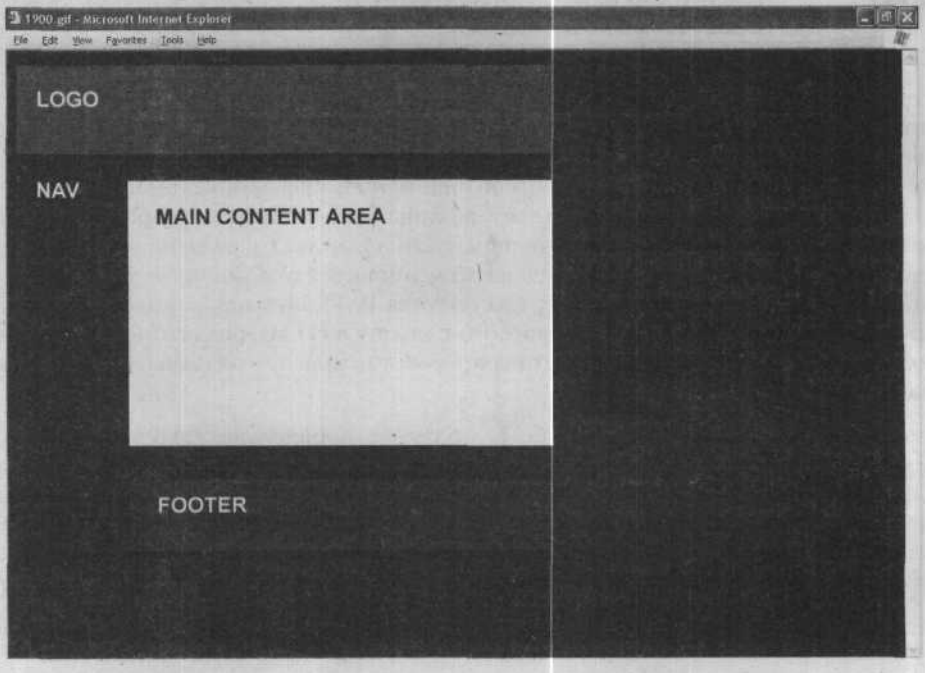


Рис. 19.2. Есть проблема. Макетная таблица использует объединения столбцов и строк, поэтому макет может быть легко разорван при работе с ним в редакторе

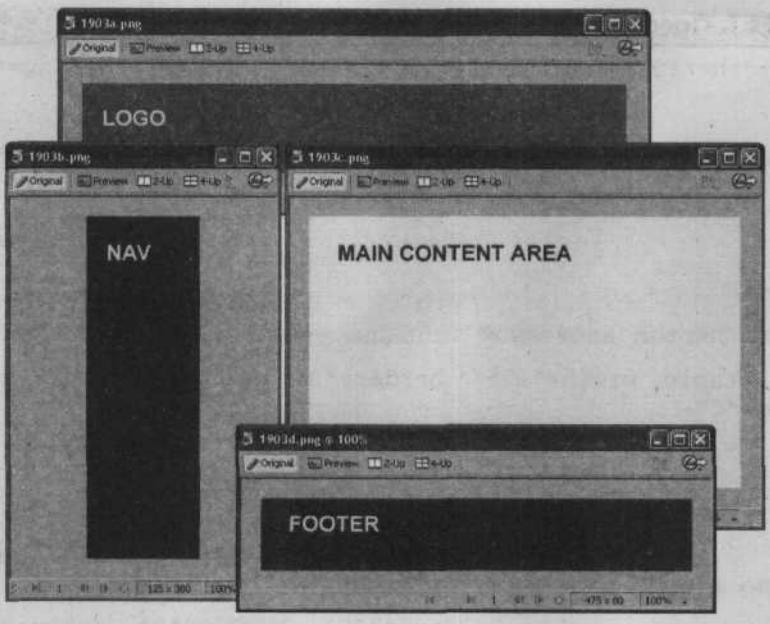


Рис. 19.3. Чтобы избежать использования объединений столбцов и строк, разделите области макета на отдельные графические файлы, а затем по отдельности разделите на срезы каждый графический файл

Как теперь обратно собрать области вместе? Наилучший способ – закатать рукава, запустить HTML-редактор и создать контейнерную таблицу вручную. Создайте полный эскиз макета, как показано на рисунке 19.4. Обратите внимание, что в нижней ячейке контейнерной таблицы необходимо использовать вложенную таблицу, чтобы избежать объединения столбцов, а в правой ячейке вложенной таблицы необходимо использовать еще одну вложенную таблицу, чтобы избежать объединения строк.

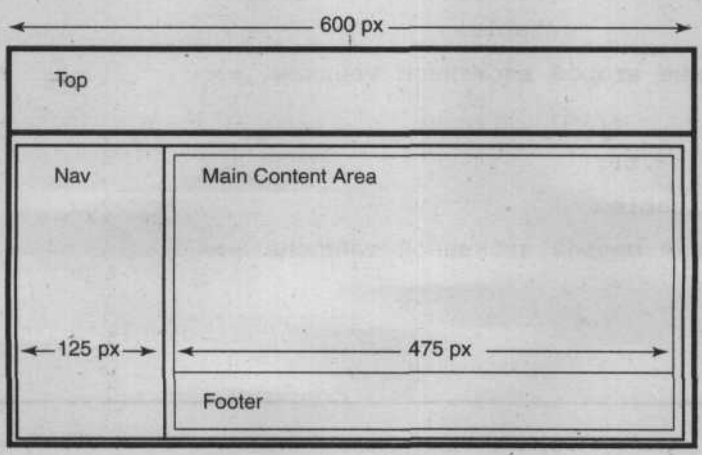


Рис. 19.4. Создайте эскиз контейнерной таблицы, используя несколько вложенных таблиц, чтобы избежать объединения столбцов и строк, и постройте эту таблицу в HTML-редакторе

Листинг 19.1. Просмотр исходного кода для рисунка 19.4

```
<table width="600" border="0" cellpadding="0" cellspacing="0">
   <tr>
      <td>Top</td>
   </tr>
   <tr>
      <td>
<!- Начало первой вложенной таблицы. ->
         <table width="125" border="0" cellpadding="0" cellspac-
      ing="0">
            <tr>
               <td>Nav</td>
               <td>
<!- Начало второй вложенной таблицы. ->
                  <table width="475" border="0" cellpadding="0"
               cellspacing="0">
                     <tr>
                        <td>Main Content Area</td>
                     </tr>
                     <tr>
                        <td>Footer</td>
                     </tr>
                  </table>
<!- Окончание второй вложенной таблицы. ->
               </td>
            </tr>
         </table>
<!- Окончание первой вложенной таблицы. ->
      </td>
   </tr>
</table>
```

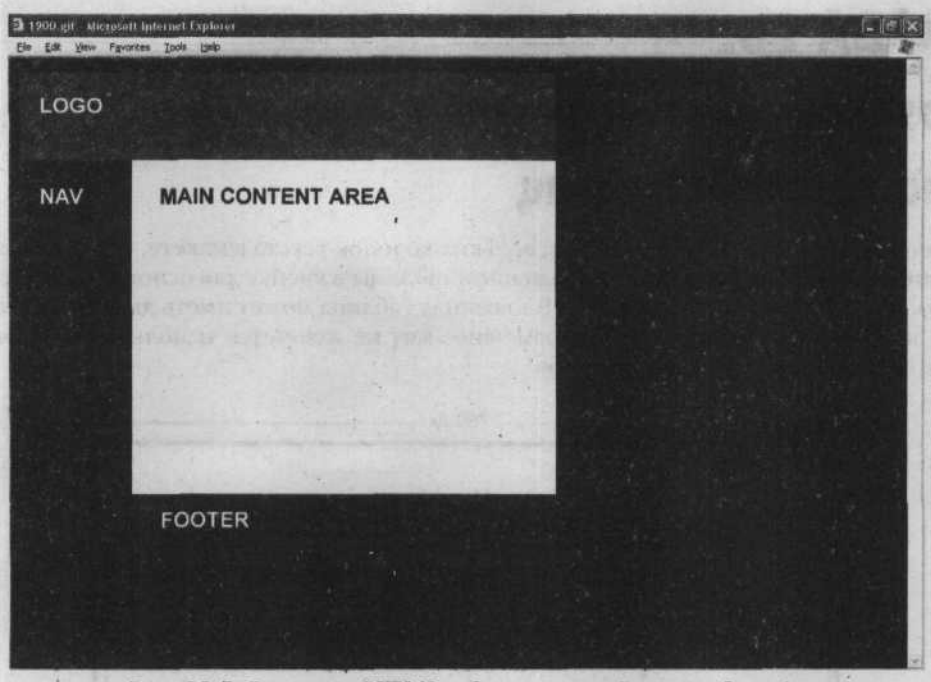


Рис. 19.5. Вставьте HTML-код, полученный при разделении графических файлов на срезы, в контейнерную таблицу. Когда размеры таблицы изменятся, макет не будет разорван

Теперь просто вставьте HTML-код четырех разделенных на срезы изображений в соответствующие ячейки контейнерной таблицы, и вы, наконец, получите макет, основанный на использовании таблиц, показанный на рисунке 19.5.

Совет

Ваше графическое приложение может иметь возможность экспортирования разделенного макета во вложенные таблицы, вместо одной таблицы с использованием объединения столбцов и строк. Вы можете попробовать данный метод, хотя зачастую лучше построить контейнерную таблицу с нуля. Кустарные таблицы в основном более прочные, чем те, которые выплевывают графические приложения.

ГЛАВА 20.
Создание многоколонных макетов при помощи таблиц

Наилучшим способом получения нескольких колонок текста в макете, основанном на таблицах, является размещение вложенной таблицы в ячейке для основного содержимого, как показано на рисунке 20.1. Вложенная таблица может иметь любое необходимое количество столбцов, хотя, возможно, вам не захочется использовать больше двух или, может быть, трех столбцов.

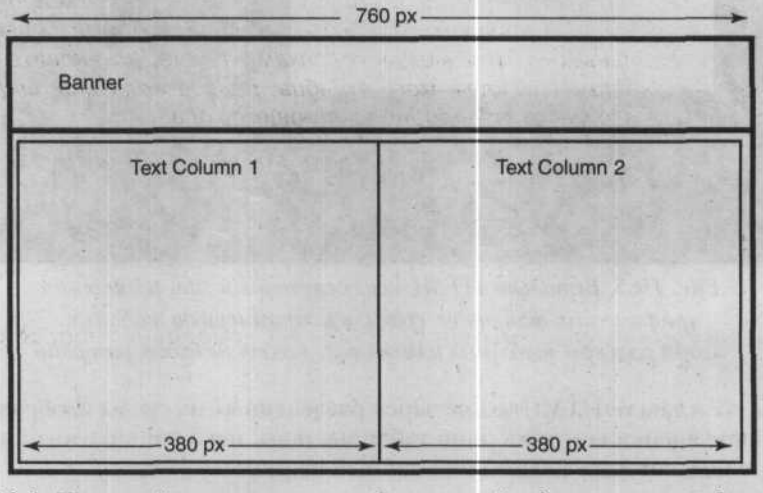


Рис. 20.1. Используйте вложенную таблицу в ячейке для основного содержимого вашего макета, чтобы создать несколько колонок для текста

Обычно, когда одна макетная таблица вкладывается внутрь другой макетной таблицы, мы устанавливаем значения атрибутов `border`, `cellpadding` и `cellspacing` в 0. Это гарантирует, что вложенная таблица будет точно подогнана к своей контейнерной ячейке. Однако когда в макетной таблице размещается несколько колонок текста, мы не хотим иметь такой эффект. Попытайтесь прочитать две колонки текста, находящихся друг возле друга и не имеющих достаточного расстояния между собой, как показано на рисунке 20.2, и вы полностью поймете почему.

Совет

Вам не следует использовать больше двух или трех колонок в многоколонном макете, и три колонки уже теснятся. Подумайте о тех посетителях вашего сайта, которые не имеют огромных мониторов с чрезвычайно высоким разрешением экрана. Если вы попытаетесь втиснуть слишком много колонок в ячейку с содержимым, то получите колонки, которые будут слишком узкими для чтения, или ужасные горизонтальные полосы прокрутки.

Здесь может помочь атрибут `cellspacing` тега `table`. Атрибут `cellspacing` управляет расстоянием между соседними ячейками. Просто задайте атрибуту `cellspacing` любое значение, отличное от 0, и вы автоматически получите свободное пространство. Точное значение зависит от того вида, который вы собираетесь получить, но значения, лежащего в пределах от 12 до 16 пикселов, обычно достаточно. Макет, представленный на рисунке 20.3, использует значение атрибута `cellspacing`, равное 12.

Совет

Значение атрибута `cellspacing` также влияет и на пространство вокруг ячеек таблиц, а не только на расстояние между ними. Поэтому колонки текста не будут точно подогнаны к сторонам контейнерной ячейки. Если это не согласуется с видом, которого вы пытаетесь достичь, оставьте значение атрибута `cellspacing` вложенной таблицы равным 0, и добавьте отдельный столбец, который будет служить свободным местом между колонками текста. Установите значение атрибута `width` для этого столбца, равное желаемой ширине свободного места, наподобие следующего кода:

```
<table width="412" border="0" cellpadding="0"
cellspacing="0">
    <tr>
    <td width="200" valign="top">First column of
text</td>
    <td width="12"><!- Свободное пространство -
></td>
    <td width="200" valign="top">Second column of
text</td>
    </tr>
</table>
```

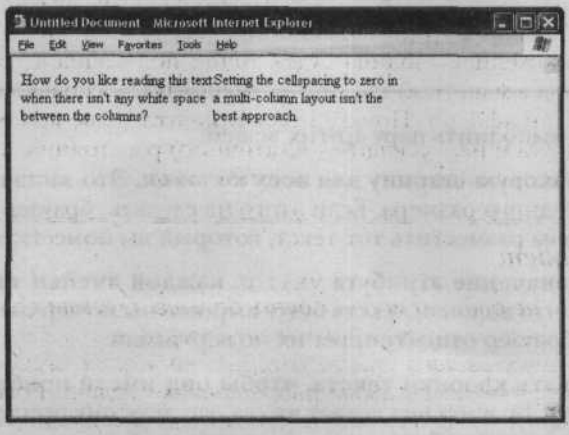


Рис. 20.2. Необходимо подумать о свободном пространстве между колонками, или вы получите что-то наподобие этого

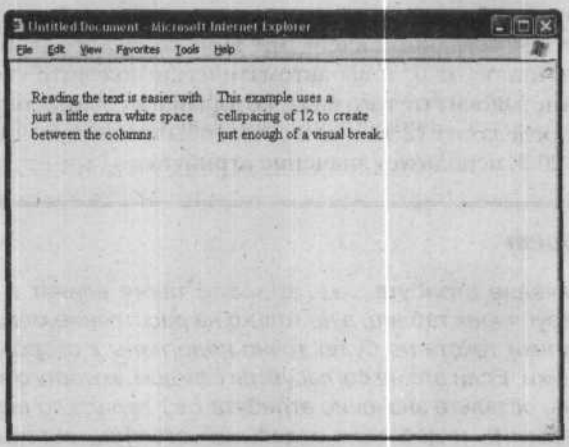


Рис. 20.3. Установка атрибута таблицы `cellspacing` в значение, например, 12, автоматически создает свободное место

Листинг 20.1. Просмотр исходного кода для рисунка 20.3

```
<table width="400" border="0" cellpadding="0" cellspacing="12">
   <tr>
      <td width="50%" valign="top">
         <p>Reading the text is easier with just a little extra
      white space between the columns.</p>
      </td>
      <td width="50%" valign="top">
         <p>This example uses a cellspacing of 12 to create just
      enough of a visual break.</p>
      </td>
   </tr>
</table>
```

Также необходимо выполнить пару других вещей:

- **Задать одинаковую ширину для всех колонок.** Это заставит браузер сделать все колонки одного размера. Если этого не сделать, браузер подгонит ширины колонок, чтобы разместить тот текст, который вы поместите в них.
- **Установить значение атрибута `valign` каждой ячейки таблицы в `top`.** Это гарантирует, что колонки текста будут выровнены по верхней линии. В противном случае, браузер отцентрирует их по вертикали.
- **Сбалансировать колонки текста, чтобы они имели приблизительно одинаковую высоту.** Браузер не сделает это за вас, поэтому придется делать руками. Убедитесь, что в каждой колонке находится приблизительно одинаковое количество текста, иначе вы испортите макет.

Инструментарий

Создание многоколонного макета при помощи таблиц

Данный инструментарий демонстрирует, как использовать вложенную таблицу для нескольких колонок текста. Разместите эту таблицу внутри основной области содержимого вашего макета.

```
<table width="contentcellwidth" border="0" cellpadding="0" cellspacing="whitespacewidth">
<!- В теге table, расположенном выше, вставьте ширину ячейки содержимого вашей макетной таблицы. Также задайте ширину свободного пространства, необходимого между колонками, обычно это значение находится между 12 и 16 пикселами. ->
   <tr>
<!- Этот блок кода создает одну колонку. Скопируйте и вставьте эту часть столько раз, сколько столбцов вам необходимо. Не забывайте указывать значение атрибута width для каждого столбца. В макете с двумя колонками значение атрибута width составляет 50%. В макете с тремя колонками значение атрибута width составляет 33%. В макете с четырьмя колонками значение атрибута width составляет 25% и так далее. ->
      <td width="columnwidth" valign="top">
         Здесь располагается содержание
      </td>
<!- Окончание кода для колонки. ->
   </tr>
</table>
```

ГЛАВА 21.
Создание многоколонных макетов при помощи таблиц CSS

Точно так же, как для создания многоколонного макета для основанного на таблицах дизайна, который был рассмотрен в главе 20, используются вложенные таблицы, для создания многоколонного макета при помощи таблиц CSS используются вложенные элементы разделов. Поместите вложенные разделы в основной раздел с содержимым макета, как показано на рисунке 21.1, и результат не заставит себя ждать. Единственное препятствие – основной раздел с содержимым должен иметь фиксированную ширину. Раздел не может быть текучим, иначе некоторые браузеры подавятся, если вы примените методику, представленную здесь.

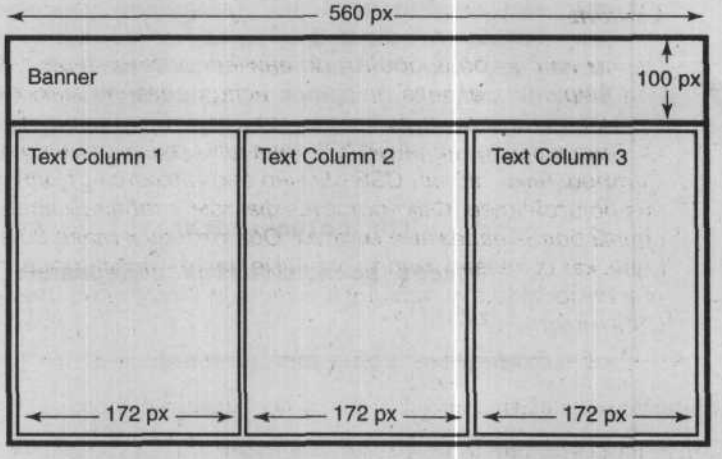


Рис. 21.1. В основном разделе содержимого данного макета находятся вложенные разделы для каждой колонки текста

Листинг 21.1. Просмотр исходного кода для рисунка 21.1

```
<div id="logo" style="position: relative; width: 560px; height:
100px;">
   Banner
</div>
<div id="container" style="position: relative; width: 560px;">
<!– Вложенные разделы начинаются здесь –>
   <div id="leftcolumn" style="position: absolute; width: 172px;
left: 0px;">
      Text Column 1
   </div>
```

```
   <div id=middlecolumn" style="position: absolute; width: 172px;
left: 186px;">
      Text Column 2
   </div>

   <div id="rightcolumn" style="position: absolute; width: 172px;
left: 372px;">
      Text Column 3
   </div>
<!- Окончание вложенных разделов ->
</div>
```

Совет

Одним из способов обойти ограничение, связанное с фиксированной шириной раздела, является использование макетной таблицы для многоколонной структуры внутри основного раздела содержимого. Смешивание таблиц и CSS-кода для макета отлично работает, хотя сторонники таблиц CSS обычно съеживаются от одного упоминания об этой идее. Факт остается фактом – таблицы предоставляют в целом более надежные макеты. Обратитесь к главе 20 за информацией, как создавать многоколонные макеты при помощи таблиц, а затем поместите эту таблицу в основной раздел содержимого вашего CSS-макета.

Выполните приведенные шаги для создания вложенных разделов:

1. Возьмите ширину основного раздела содержимого и разделите это значение на число колонок, которое вы хотите создать. На рисунке 21.1 ширина основного раздела содержимого составляет 560 пикселов, следовательно, разделив 560 на три колонки, мы получим приблизительно 186 пикселов. Это нерегулируемая ширина каждой из колонок.
2. Между колонками необходимо свободное место, поэтому удержите несколько пикселов от значения ширины, полученного на шаге 1. Четырнадцати пикселов для ширины свободного места вполне достаточно, поэтому, из 186 вычитаем 14 и получаем значение ширины, равное 172 пикселам на колонку.
3. Нам необходима горизонтальная позиция каждой колонки, выраженная в виде смещения от левой стороны родительского раздела – в данном случае, основной держатель содержимого. Смещение слева первой колонки всегда равняется 0 пикселам. Следующая колонка смещается на исходную ширину, рассчитанную на шаге 1, или на 186 пикселов слева. Следующая колонка смещается на двойную ширину колонки, или на 372 пиксела слева, и так далее, в зависимости от того, сколько колонок вы имеете.

Напишите код для вложенных разделов, используя значения из предыдущих шагов. Инструментарий, приведенный ниже, предоставляет вам шаблон.

Совет

Как и в случае с таблицами, не пытайтесь уместить на макете больше, чем две или три CSS-колонки.

Определение

Почему позиции колонок задаются относительно позиции основного раздела содержимого, а не окна браузера?

Поскольку разделы колонок являются вложенными, они наследуют свое базовое положение от родительского раздела. Этот же код для невложенных разделов создаст колонки, начинающиеся прямо от левой стороны окна браузера.

Инструментарий

Создание многоколонного макета при помощи таблиц CSS

Данный инструментарий предоставляет код для многоколонных разделов. Вложите их в основной раздел содержимого вашего макета. Помните, что основной раздел содержимого должен иметь фиксированную ширину – его ширина должна иметь абсолютный размер в пикселах, а не в процентах.

```
<!– Каждая колонка получает свой собственный раздел. Нумерация колонок слева направо начинается с 0, а не с 1, иначе формула в коде будет работать неправильно.

Замените выражение columnwidth значением, полученным на шаге 2 вышеописанной процедуры.

Замените выражение columnposition значением, полученным на шаге 1 вышеописанной процедуры. –>

<div id="column columnnumber" style="position: absolute; width: columnwidthpx; left: (columnposition * columnnumber)px;">

   Содержимое располагается здесь

</div>

<!– Повторите этот блок кода для каждой колонки макета. Для большинства сайтов не пытайтесь втиснуть в основной раздел содержимого больше, чем две или три колонки. –>
```

ГЛАВА 22.
Предотвращение разрыва навигационной панели

При разработке макетной таблицы все мысли должны быть направлены на использование наименьшего количества ячеек. Так будет проще для дизайнерского программного обеспечения, так будет проще для кодировщиков, и, что самое важное, так будет проще для вас.

В зависимости от того, где какой элемент будет расположен на странице, возможно, вы сможете сформировать макет, похожий на изображенный на рисунке 22.1. Просто заполните ячейки соответствующим содержимым, позвольте таблице увеличиваться до бесконечности по вертикали, загрузите страницу в Интернете и можете считать дело завершенным.

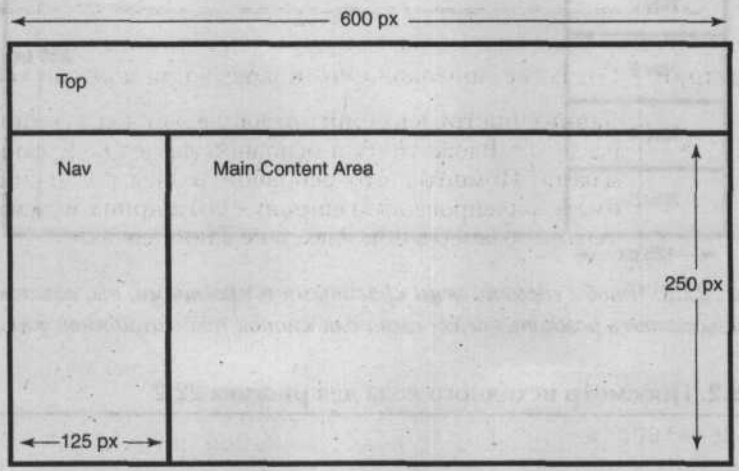


Рис. 22.1. Простой макет, наподобие представленного на этом рисунке, может сберечь ваше время и усилия

Листинг 22.1. Просмотр исходного кода для рисунка 22.1

```
<table width="600">
   <tr>
      <td colspan="2">Top</td>
   </tr>
   <tr>
      <td width="125">Nav</td>
      <td height="250">Main Content Area</td>
   </tr>
</table>
```

Воодушевленные простотой, вы можете испытать искушение разбить ячейку слева на отдельные строки для кнопок навигационной панели, используя атрибут rowspan, как показано на рисунке 22.2.

Макет, представленный на рисунке 22.2, работает в том случае, когда заранее известна высота основной области содержимого. (В данном случае высота равняется 250 пикселам.) Данное значение высоты не всегда просто высчитать. Возможно, ваш дизайн будет использоваться на разных страницах, каждая из которых имеет различный объем содержимого. Возможно, ваш сайт будет изменяться в будущем, и дизайн должен быть достаточно гибким для работы с различными объемами содержимого.

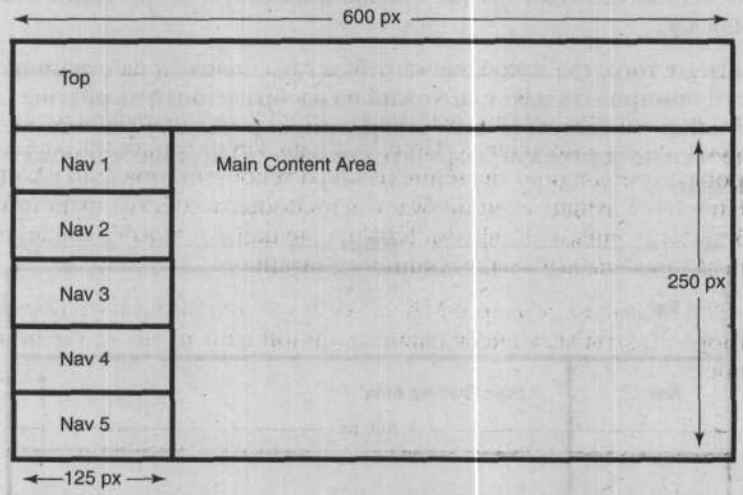


Рис. 22.2. Чтобы сделать вещи красивыми и простыми, вы, возможно, попытаетесь разбить ячейку слева для кнопок навигационной панели

Листинг 22.2. Просмотр исходного кода для рисунка 22.2

```
<table width="600">
   <tr>
      <td colspan="2">Top</td>
   </tr>
   <tr>
      <td width="125">Nav 1</td>
<!- Далее идет ячейка с объединением строк ->
      <td height="250" rowspan="5">Main Content Area</td>
   </tr>
   <tr>
      <td>Nav 2</td>
   </tr>
   <tr>
```

```
        <td>Nav 3</td>
    </tr>
    <tr>
        <td>Nav 4</td>
    </tr>
    <tr>
        <td>Nav 5</td>
    </tr>
</table>
```

Даже если вы все уже посчитали, вспомните старую поговорку: хочешь рассмешить бога – расскажи ему о своих планах. Пользователь, вероятно, будет работать с браузером, у которого по умолчанию значение размера текста установлено в **Largest** (Самый крупный). Ничуть не лучше, если он будет использовать собственную таблицу стилей с каким-нибудь загадочным шрифтом. Кто-то еще найдет способ, как вытолкнуть границы макета за пределы высоты, указанной в дизайне.

На рисунке 22.3 показано, что произойдет. При увеличении высоты основной области содержимого высоты всех ячеек навигационной панели также увеличиваются для согласованности.

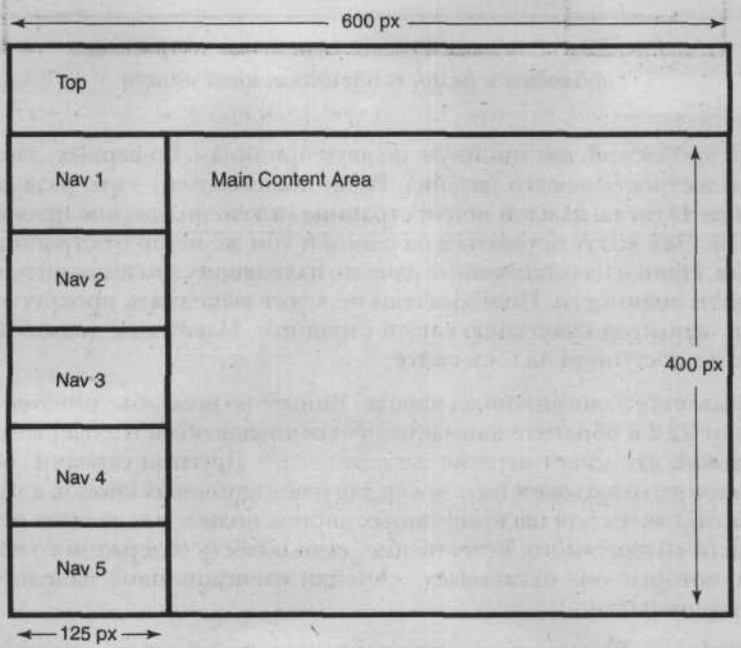


Рис. 22.3. При увеличении высоты основной области содержимого высоты ячеек навигационной панели не отстают от общей высоты

Теоретически это может показаться не такой уж и большой проблемой, однако посмотрите на рисунок 22.4. Обратите внимание, как растянулись кнопки навигационной панели, когда основная область содержимого вышла за пределы максимальной высоты. Белый фон страницы просматривается через дыры в дизайне, создавая уродливый беспорядок.

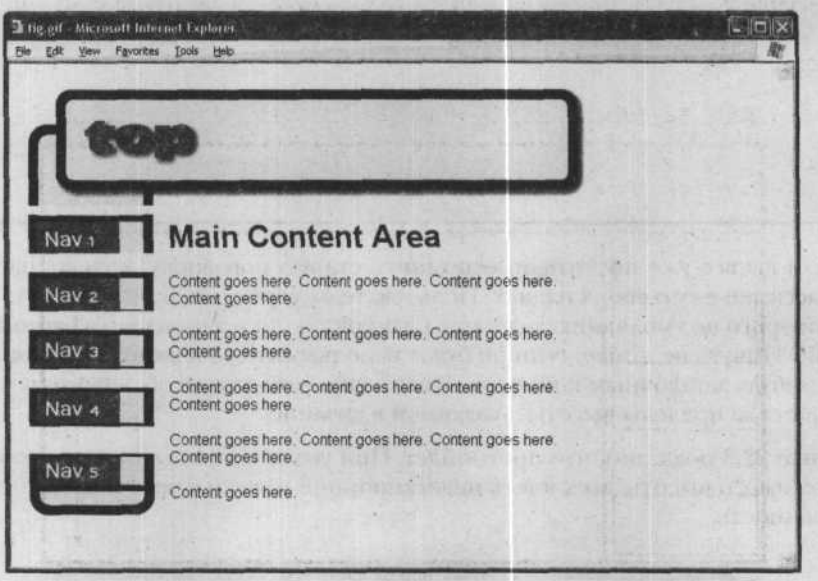


Рис. 22.4. Слишком большое количество содержимого приводит к разрыву навигационной панели

Это является проблемой, как минимум, по двум причинам. Во-первых, данный подход разрушает целостность вашего дизайна. Ваши пользователи – это раздражительные люди, и они не хотят на каждой новой странице искать положение правой навигационной кнопки. Они хотят оставаться на одном и том же месте от страницы к странице. Во-вторых, кнопки навигационной панели, находящиеся в самом низу, могут исчезнуть из области видимости. Пользователи не хотят выполнять прокрутку страницы, чтобы найти основную навигацию вашей страницы. Навигация должна быть понятной и тотчас же доступной на всем сайте.

Почему разрывается навигационная панель? Вините во всем объединение строк. Проверьте листинг 22.2 и обратите внимание, что основная область содержимого является единственной, кто имеет атрибут `rowspan="5"`. Другими словами, основная область содержимого охватывает пять ячеек для навигационных кнопок в макетной таблице, т.е. высоты ячеек для навигационных кнопок полностью зависят от высоты основной области содержимого. Естественно, если область содержимого изменяет размер, строки, которые она охватывает, – ячейки навигационной панели – изменяют размер пропорционально.

Вот в чем проблема. Вы можете попытаться устранить ее, указав высоты для каждой из пяти ячеек навигационной панели, однако, увы, этот подход не работает. Браузер автоматически раздувает ячейки навигационной панели, чтобы заставить их охватить область содержимого, независимо от того, какие значения вы присвоили. Как только

основная область содержимого выйдет за пределы предопределенной высоты, браузер игнорирует значение высоты для данной ячейки и высот, охваченных строками, и навигационная панель разрывается в любом случае.

Простейшим способом заставить макет работать является использование вложенных таблиц, или таблиц внутри других таблиц. Вернитесь к исходному макету, показанному на рисунке 21.1. Эта просторная ячейка слева может хранить любой тип содержимого: текст, изображение, медиа-файл или даже другую таблицу, как показано на рисунке 22.5.

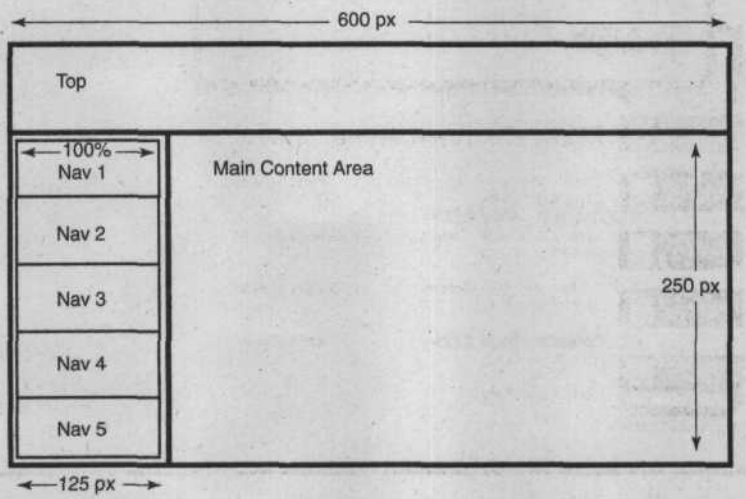


Рис. 22.5. Создайте вложенную таблицу для навигационной панели в ячейке слева

Совет

Как только содержимое будет добавлено в ячейку основной области содержимого, избавьтесь от атрибута `height`*, поскольку браузер делает высоту ячейки таблицы такой, какой она должна быть.*

Листинг 22.3. Просмотр исходного кода для рисунка 22.5

```
<table width="600">
   <tr>
      <td colspan="2">Top</td>
   </tr>
   <tr>
      <td width="125" valign="top">
<!— Начало вложенной таблицы. —>
```

```
        <table width="100%">
            <tr>
                <td>Nav 1</td>
            </tr>
            <tr>
                <td>Nav 2</td>
            </tr>
            <tr>
                <td>Nav 3</td>
            </tr>
            <tr>
                <td>Nav 4</td>
            </tr>
            <tr>
                <td>Nav 5</td>
            </tr>
        </table>
<!- Окончание вложенной таблицы. ->
    </td>
    <td height="250" valign="top">Main Content Area</td>
  </tr>
</table>
```

Обратите внимание, что значение атрибута `width` вложенной таблицы установлено в 100%. Это заставляет таблицу увеличиваться, чтобы заполнять всю ширину родительской ячейки. Обратите также внимание, что ячейка с вложенной таблицей имеет атрибут `valign="top"`. Данный атрибут заставляет браузер выравнивать содержимое ячейки по верху, что все время удерживает вложенную таблицу возле самого верхнего края ячейки, прямо там, где посетители ожидают найти ее.

Совет

Если после помещения навигационной панели во вложенную таблицу, она все еще выглядит разорванной, проверьте атрибуты тега `table` *вложенной таблицы. Возможно, вам необходимо установить значения атрибутов* `cellpadding`, `cellspacing` *и* `border` *в 0, наподобие следующей строки кода:*

```
<table width="100%" cellpadding="0" cellspacing="0"
border="0">
```

Теперь в основную ячейку можно добавить столько содержимого, сколько пожелаете. Поскольку мы не использовали объединение строк для создания ячеек навигационной панели, в основную ячейку можно добавить сколь угодно много содержимого, и навигационная панель не будет разорвана. Макет на рисунке 22.6 доказывает это. Ячейка, содержащая вложенную таблицу, выросла, однако размеры вложенной таблицы не изменятся до тех пор, пока вы не измените содержимое в ячейках вложенной таблицы.

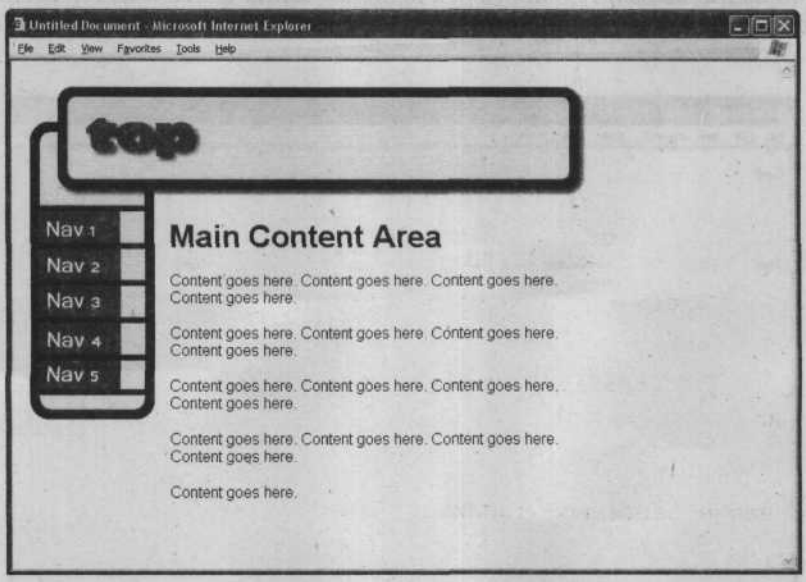


Рис. 22.6. Вложенная таблица исключает объединение строк, что сохраняет навигационную панель нетронутой независимо от того, какое количество содержимого будет добавлено

Совет

Если вы используете программное обеспечение, например, Macromedia Fireworks, Adobe Photoshop или Adobe ImageReady, для разделения дизайна на срезы, неожиданный разрыв навигационной панели присутствует всегда. Чтобы избежать этого, разделяйте на срезы каждую основную область дизайна по отдельности. За дополнительной информацией обратитесь к главе 19.

ГЛАВА 23.
Раскрашивание ячеек таблицы

Ячейки таблицы в языке HTML имеют атрибут `bgcolor`, управляющий цветом фона ячейки. Задание цветов фона для определенных ячеек в макете придает вашему дизайну большую выразительность без использования файлов изображений. Сравните макет, представленный на рисунке 23.1, с макетом на рисунке 23.2, и вы согласитесь с этим.

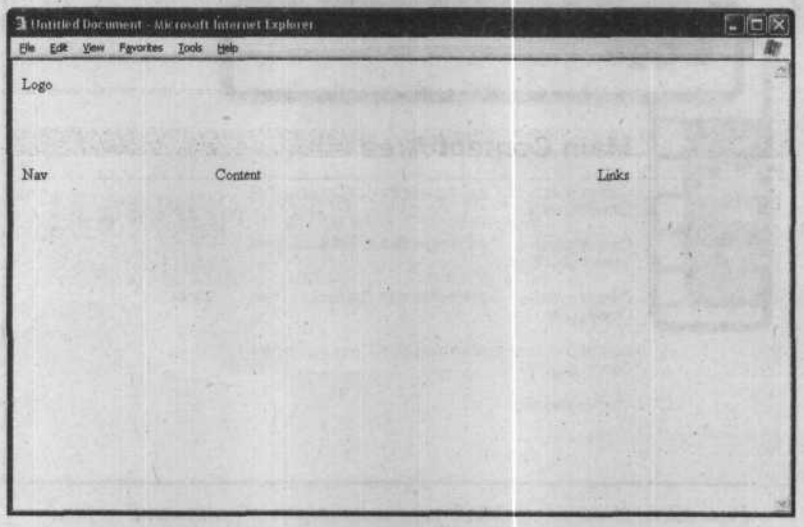


Рис. 23.1. Ячейки таблицы в данном макете не используют цвета фона

Листинг 23.1. Просмотр исходного кода для рисунка 23.1

```
<table width="760" border="0" cellpadding="0" cellspacing="0">
   <tr>
      <td width="760" valign="top">Logo</td>
   </tr>
   <tr>
      <td width="760" valign="top">
         <table width="760" border="0" cellpadding="0" cellspac-
      ing="0">
            <tr>
               <td width="200" valign="top">Nav</td>
               <td width="400" valign="top">Content</td>
               <td width="160" valign="top">Links</td>
            </tr>
```

```
        </table>
      </td>
    </tr>
</table>
```

Совет

Используйте вместо файлов изображений цвета фона везде, где это возможно. Поскольку цвета определяются в языке HTML, вашим посетителям не придется загружать отдельные файлы изображений, чтобы увидеть цвета.

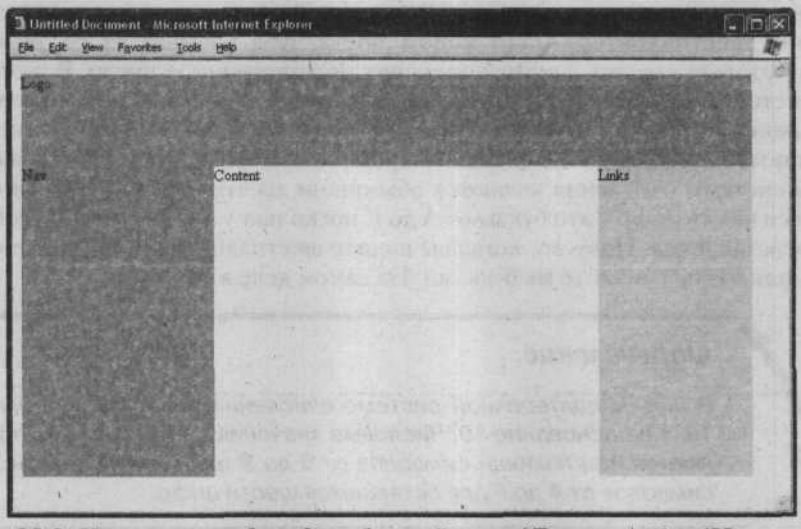


Рис. 23.2. Указав цвета фона для областей **Logo** *(Логотип),* **Nav** *(Навигация) и* **Links** *(Ссылки), вы создадите более выразительную страницу без использования файлов изображений*

Листинг 23.2. Просмотр исходного кода для рисунка 23.2

```
<table width="760" border="0" cellpadding="0" cellspacing="0">
  <tr>
    <td width="760" valign="top" bgcolor="#99CCFF">Logo</td>
  </tr>
  <tr>
    <td width="760" valign="top">
      <table width="760" border="0" cellpadding="0" cellspac-
    ing="0">
        <tr>
```

```
            <td  width="200"  valign="top"  bgcolor="#99CCFF">
        Nav</td>
            <td width="400" valign="top">Content</td>
            <td  width="160"  valign="top"  bgcolor="#99FF99"
        >Links</td>
        </tr>
      </table>
    </td>
  </tr>
</table>
```

Обратите внимание, что в листинге 23.2 желаемый цвет задается строкой символов, называемой **шестнадцатеричным кодом цвета**. Код начинается со знака решетки (#), за которым следует шестизначное шестнадцатеричное число. В шестнадцатеричной системе счисления используется основание 16, а не основание 10, как в нашей распространенной, ежедневно используемой десятичной системе счисления, что уже знают фанаты математики, находящиеся среди нас. Первые десять цифр шестнадцатеричной системы счисления являются обычными десятичными цифрами от 0 до 9. Оставшиеся шесть цифр – это буквы от A до F, поскольку у нас нет числовых символов для их представления. Поэтому, когда вы видите шестнадцатеричное значение, наподобие FC или FF, не считайте их буквами. На самом деле это числа.

Определение

В шестнадцатеричной системе счисления используется основание 16, а не основание 10. Числовые значения в этой системе представляются при помощи символов от 0 до 9 для первых десяти цифр, и символов от A до F для оставшихся шести цифр.

Совет

Убедитесь, что на фоне цвета ячейки таблицы текст виден отчетливо. Если контраст выглядит так, будто можно было пойти другим путем, значит вы допустили ошибку из-за невнимательности.

Шестнадцатеричный код цвета включает три набора шестнадцатеричных чисел по два разряда каждый. Первый двухзначный набор отвечает за количество красного в цвете. Второй набор управляет количеством зеленого, а третий – количеством синего. Смешивание различных уровней красного, зеленого и синего цветов создает каждый отдельный цвет, который вы видите на экране. Чем выше значение конкретного двухзначного набора, тем более заметным будет составляющий цвет в конечном оттенке.

Возьмем шестнадцатеричный код для красного цвета, который равен #FF0000. Если его разложить на составляющие, мы получим значение FF для красной составляющей,

значение 00 для зеленой составляющей и значение 00 для синей составляющей. Это абсолютно красный цвет, поскольку в позиции для красной составляющей находится значение FF – самое большое значение из всех возможных для двухзначного шестнадцатеричного числа. Для этого конкретного оттенка составляющие зеленого и синего цветов не снижают чистоту красного цвета. Подобным образом, код для зеленого цвета равен #00FF00 (ноль красного, полный зеленый, ноль синего), а код для синего цвета равен #0000FF (ноль красного, ноль зеленого, полный синего).

Совет

Цифры в каждом положении шестнадцатеричного кода числа не обязаны совпадать. Код #FC02D5 является отличным шестнадцатеричным цветом, где значение FC находится в позиции для красной составляющей, 02 – в позиции для зеленой составляющей, и D5 – в позиции для синей составляющей. Однако для получения наилучших результатов в Интернете придерживайтесь цифр, кратных трем: 0, 3, 6, 9, C и F. Следовательно, ближайшим Web-эквивалентом для кода #FC02D5 является код #FF00CC.

Чтобы создать цвет, например, фиолетовый, смешивается вместе пара составляющих. Красная и синяя составляющие создают фиолетовый цвет, поэтому одной из возможностей является смешивание полной красной и полной синей составляющих, как, например, #FF00FF, что дает очень яркий и чистый цвет. Но, возможно, вы захотите получить фиолетовый цвет с краснотой. Просто уменьшите уровень синей составляющей, присвоив ей меньшее значение, например, #FF0099 или #FF0066. Чтобы получить фиолетовый цвет с синевой, выполните противоположное действие. Оставьте полной синюю составляющую и уменьшите уровень красной составляющей, например #9900FF или #6600FF.

Совет

Если вы по-настоящему ненавидите шестнадцатеричные коды, можете вместо них всегда указывать название цвета, например, `bgcolor="lightgreen"`*, однако лучше все же использовать ближайшее шестнадцатеричное значение для Интернета, которое менее открыто для интерпретации браузером посетителя. Более того, названия могут вводить в заблуждение. Цвет с названием* ***darkgray*** *(темно-серый) на самом деле соответствует более светлому оттенку серого, чем цвет с названием* ***gray*** *(серый).*

Немного попрактиковавшись, вы можете научиться смешивать любые необходимые цвета, не утруждая себя запоминанием чего-либо из таблиц, похожих на таблицу 23.1, в которой приведены названия и шестнадцатеричные коды цветов для распространенных оттенков.

Табл. 23.1. Распространенные цвета и их коды

Цвет	Название	Шестнадцатеричный код	Ближайший Web-эквивалент
Зеленовато-голубой	aqua	#00FFFF	#00FFFF
Черный	black	#000000	#000000
Синий	blue	#0000FF	#0000FF
Синий, темный	darkblue	#00008B	#000099
Синий, светлый	lightblue	#ADD8E6	#CCCCFF
Синий, небесный	skyblue	#87CEEB	#99CCFF
Золотой	gold	#FFD700	#FFCC00
Серый	gray	#808080	#999999
Серый, темный	dimgray*	#696969	#666666
Зеленый	green	#008000	#009900
Зеленый, темный	darkgreen	#006400	#006600
Зеленый, светлый	lightgreen	#90EE90	#99FF99
Зеленый, морской	seagreen	#2E8B57	#339966
Индиго	indigo	#4B0082	#330099
Оранжевый	orange	#FF5A00	#FF9900
Розовый	pink	#FFC0CB	#FFCCCC
Пурпурный	purple	#800080	#990099
Красный	red	#FF0000	#FF0000
Красный, темный	darkred	#8B0000	#990000
Серебристый	silver	#C0C0C0	#CCCCCC
Фиолетовый	violet	#EE82EE	#FF99FF
Белый	white	#FFFFFF	#FFFFFF
Желтый	yellow	#FFFF00	#FFFF00

* Цвет с названием **darkgray** (темно-серый) на самом деле светлее, чем цвет с названием **gray** (серый). Вместо него используйте цвет **dimgray** (тускло-серый).

ГЛАВА 24.
Раскрашивание и форматирование CSS-разделов

В главе 23 было показано, как раскрашивать ячейки таблицы. Для разделов можно также указывать цвета фона, вместе с парой других атрибутов, касающихся их внешнего вида, которые HTML-таблицы просто не поддерживают. И вновь, как и во всех других случаях при использовании таблиц CSS, обязательно протестируйте свой дизайн на нескольких браузерах перед тем, как загрузить страницу в Интернет. Кто знает, как различные браузеры исказят вашу совершенно правильную CSS-разметку на этот раз.

Если макет, представленный на рисунке 24.1, кажется вам знакомым, то это потому, что вы уже видели нечто похожее на него до этого момента – если быть точными, в главе 18. В коде не используются цвета фона, что подтверждается соответствующим листингом. Как можно увидеть на рисунке 24.2, добавление атрибута `background-color` к определению стиля создает больший графический эффект.

Листинг 24.1. Просмотр исходного кода для рисунка 24.1

```
<div id="logo" style="position: absolute; width: 100%; height:
100px; top: 0px; left: 0px;">
   Logo
</div>

<div id="nav" style="position: absolute; width: 200px; top: 100px;
left: 0px;">
   Nav
</div>

<div id="links" style="position: absolute; width: 175px; top:
100px; right: 0px;">
   Links
</div>

<div id="content" style="position: absolute; top: 100px; left:
200px; right: 175px;">
   Content
</div>
```

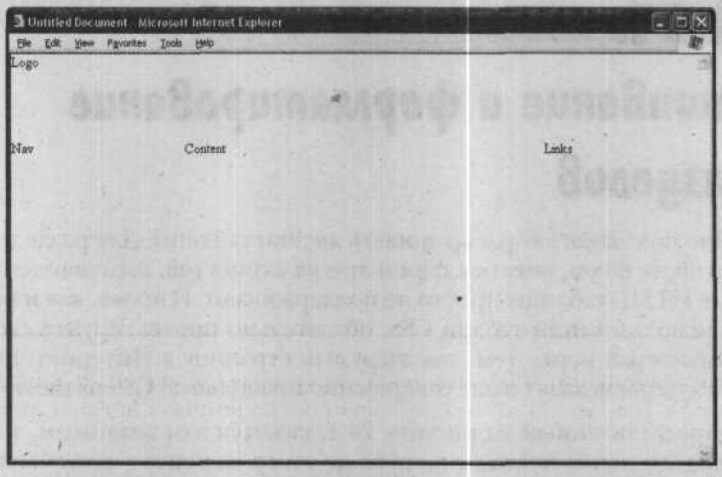


Рис. 24.1. В этом CSS-макете цвета фона не указываются ни для какого раздела

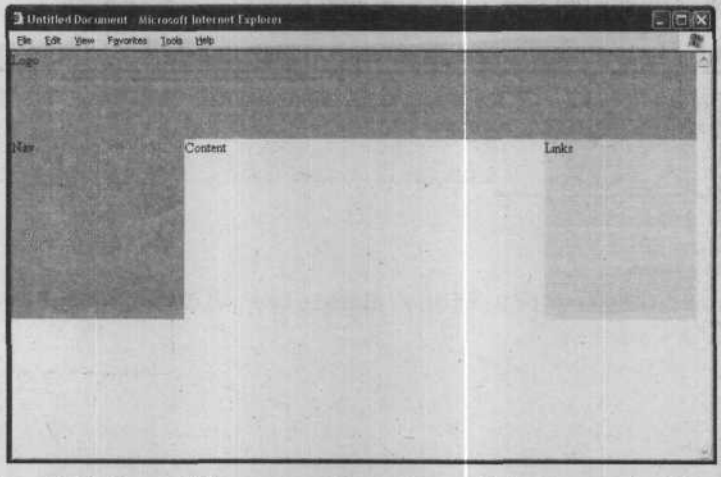


Рис. 24.2. Этот CSS-макет создает больший графический эффект, указывая цвета фона в разделах ***Logo*** *(Логотип),* ***Nav*** *(Навигация) и* ***Links*** *(Ссылки)*

Листинг 24.2. Просмотр исходного кода для рисунка 24.2

```
<div id="logo" style="position: absolute; width: 100%; height:
100px; top: 0px; left: 0px; background-color: #99CCFF;">
   Logo
</div>

<div id="nav" style="position: absolute; width: 200px; top: 100px;
left: 0px; background-color: #99CCFF;">
   Nav
</div>
```

```
<div id="links" style="position: absolute; width: 175px; top:
100px; right: 0px; background-color: #99FF99;">
  Links
</div>

<div id="content" style="position: absolute; top: 100px; left:
200px; right: 175px;">
   Content
</div>
```

Как и в случае с ячейками таблицы, цвет фона задается шестнадцатеричным кодом цвета. Обратитесь к главе 23 за списком распространенных цветов и их шестнадцатеричных значений.

Поскольку вы работаете с таблицами CSS, вы не обязаны останавливаться на раскрашивании фона. Так, вокруг раздела **Links** (Ссылки) можно добавить сплошную черную рамку, как показано на рисунке 24.3.

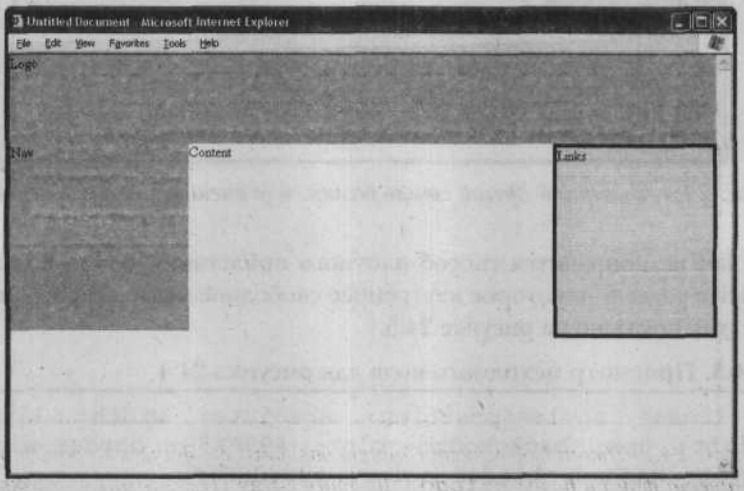


Рис. 24.3. Только ваши предпочтения и возможности клиентского браузера по работе с таблицами CSS могут помешать добавлению рамки вокруг раздела ***Links*** *(Ссылки)*

Совет

Можно определить различный внешний вид верхнего, левого, правого и нижнего сегментов рамки. За подробной информацией обратитесь к таблице 24.1 в конце данной главы.

Для создания рамки в определение стиля раздела добавляется три новых атрибута:

```
<div id="links" style="position: absolute; width: 175px; top:
```

```
100px; right: 0px; background-color: #99FF99; border-width: 4px;
border-style: solid; border-color: #000000;">
   Links
</div>
```

Возможно, вместо сплошной линии вы захотите использовать разрезную пунктирную линию или использовать голубую рамку вместо черной, или использовать более тонкую линию для рамки, как показано на рисунке 24.4.

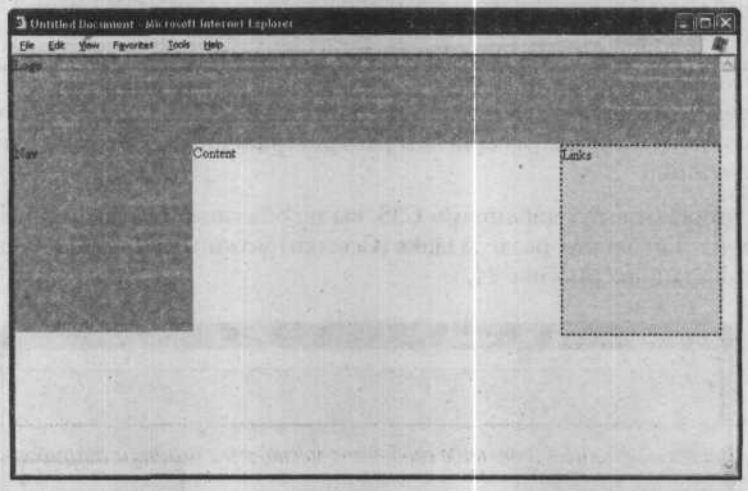


Рис. 24.4. Выберите другой стиль рамки, и ее внешний вид изменится

Возможно, вам не понравится способ плотного прилегания текста к углу раздела. Просто задайте разделу некоторое внутреннее свободное место при помощи атрибута padding, как показано на рисунке 24.5.

Листинг 24.3. Просмотр исходного кода для рисунка 24.4

```
<div id="links" style="position: absolute; width: 175px; top:
100px; right: 0px; background-color: #99FF99; border-width: 2px;
border-style: dashed; border-color: #0000FF;">
   Links
</div>
```

Листинг 24.4. Просмотр исходного кода для рисунка 24.5

```
<div id="links" style="position: absolute; width: 175px; top:
100px; right: 0px; background-color: #99FF99; border-width: 2px;
border-style: dashed; border-color: #0000FF; padding: 12px;">
   Links
</div>
```

Совет

Как и при работе со стилями рамки, можно сойти с ума, добавляя свободное место и определяя его различные значения для верхней, левой, правой и нижней стороны раздела. Все это собрано в таблице 24.1.

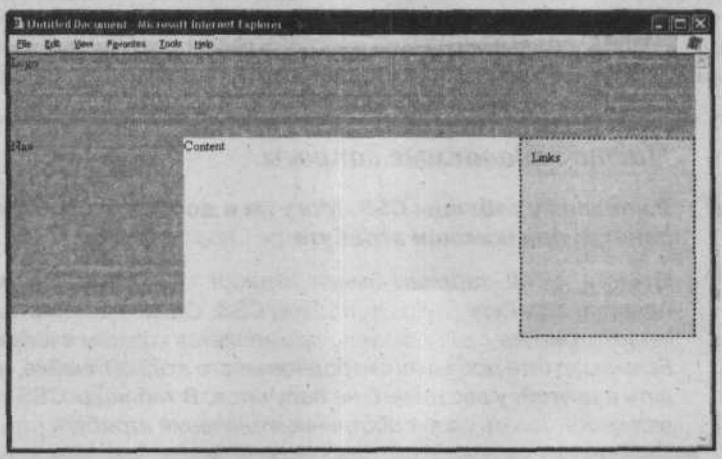


Рис. 24.5. Вставьте дополнительное свободное место при помощи атрибута `padding`

И, как вы уже видели (или не видели) в главе 10, на самом деле элементы раздела можно заставить исчезнуть, не удаляя при этом их код, как показано на рисунке 24.6.

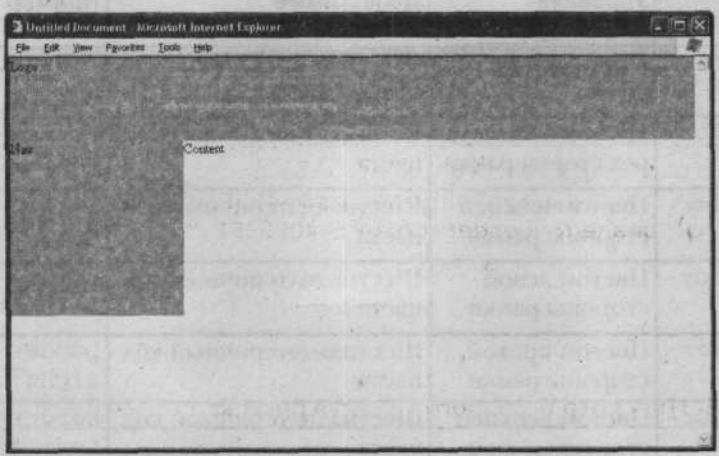


Рис. 24.6. Куда исчез раздел? Он все еще находится в исходном коде, однако его атрибут `visibility` *установлен в значение* `hidden`

Листинг 24.5. Просмотр исходного кода для рисунка 24.6

```
<div id="links" style="position: absolute; width: 175px; top:
100px; right: 0px; background-color: #99FF99; border-width: 2px;
```

```
 border-style: dashed; border-color: #0000FF; padding: 12px; visi-
bility: hidden;">
   Links
</div>
```

В таблице 24.1 приведены распространенные CSS-атрибуты для форматирования разделов макета. Помните, что каждый раздел может иметь свой собственный определенный стиль, поэтому таблицы CSS предоставляют гораздо больший диапазон возможностей, чем которые когда-либо предоставляли HTML-таблицы. Повторюсь, что HTML-таблицы являются более устойчивыми, следовательно, начав работу с таблицами CSS, вы приносите в жертву высокую совместимость между браузерами.

Часто задаваемые вопросы

Я ненавижу таблицы CSS. Могу ли я добавить свободное пространство при помощи атрибута `cellpadding`**?**

Отчасти. HTML-таблицы имеют атрибут `cellpadding`*, работающий подобно атрибуту* `padding` *таблиц CSS. Однако в HTML-таблице значение атрибута* `cellpadding` *применяется ко всем ячейкам таблицы. Если вы хотите добавить свободное место в одной ячейке, но не добавлять в другой, у вас ничего не получится. В таблицах CSS каждый раздел может иметь свое собственное значение атрибута* `padding`*.*

Табл. 24.1. Распространенные CSS-атрибуты для форматирования разделов макета

Атрибут	Управляет	Допустимые значения*	Пример
`background-color`	Цветом фона раздела	Шестнадцатеричный код цвета	`background-color: #FFCC00;`
`border-color`	Цветом всех четырех сторон рамки	Шестнадцатеричный код цвета	`border-color-top: #000000;`
`border-color-bottom`	Цветом нижней стороны рамки	Шестнадцатеричный код цвета	`border-color-bottom: #000000;`
`border-color-left`	Цветом левой стороны рамки	Шестнадцатеричный код цвета	`border-color-left: #000000;`
`border-color-right`	Цветом правой стороны рамки	Шестнадцатеричный код цвета	`border-color-right: #000000;`
`border-color-top`	Цветом верхней стороны рамки	Шестнадцатеричный код цвета	`border-color-top: #000000;`
`border-style`	Типом рамки всех четырех сторон	dashed, dotted, double, groove, inset, outset, ridge, solid	`border-style: dashed;`
`border-style-bottom`	Типом рамки снизу	dashed, dotted, double, groove, inset, outset, ridge, solid	`border-style-bottom: dotted;`

Атрибут	Управляет	Допустимые значения*	Пример
border-style-left	Типом рамки с левой стороны	dashed, dotted, double, groove, inset, outset, ridge, solid	border-style-left: double;
border-style-right	Типом рамки с правой стороны	dashed, dotted, double, groove, inset, outset, ridge, solid	border-style-right: groove;
border-style-top	Типом рамки сверху	dashed, dotted, double, groove, inset, outset, ridge, solid	border-style-top: inset;
border-width	Толщиной всех четырех сторон рамки	Любые размеры, поддерживаемые таблицами CSS, например: px, cm или in	border-width: 4px;
border-width-bottom	Толщиной нижней стороны рамки	Любые размеры, поддерживаемые таблицами CSS, например: px, cm или in	border-width-bottom: 0.03cm;
border-width-left	Толщиной левой стороны рамки	Любые размеры, поддерживаемые таблицами CSS, например: px, cm или in	border-width-left: 0.03in;
border-width-right	Толщиной правой стороны рамки	Любые размеры, поддерживаемые таблицами CSS, например: px, cm или in	border-width-right: 4px;
border-width-top	Толщиной верхней стороны рамки	Любые размеры, поддерживаемые таблицами CSS, например: px, cm или in	border-width-top: 0.03in;
padding	Размером свободного места по всем четырем сторонам раздела	Любые размеры, поддерживаемые таблицами CSS, например: px, cm или in	padding: 12px;
padding-bottom	Количеством свободного места снизу раздела	Любые размеры, поддерживаемые таблицами CSS, например: px, cm или in	padding-bottom: 0.33cm;
padding-left	Размером свободного места с левой стороны раздела	Любые размеры, поддерживаемые таблицами CSS, например: px, cm или in	padding-left: 0.33in;
padding-right	Размером свободного места с правой стороны раздела	Любые размеры, поддерживаемые таблицами CSS, например: px, cm или in	padding-right: 12px;
padding-top	Размером свободного места сверху раздела	Любые размеры, поддерживаемые таблицами CSS, например: px, cm или in	padding-top: 0.33cm;
visibility	Отображением или сокрытием раздела	hidden, visible	visibility: hidden;

* Некоторые браузеры могут поддерживать другие значения, помимо перечисленных, а некоторые браузеры могут не поддерживать даже приведенные значения.

ГЛАВА 25.
Управление пространством дизайна в языке HTML

По умолчанию Web-браузеры добавляют некоторое дополнительное пространство между интерфейсом браузера и началом страницы. Это удобно, когда у вас есть Web-страница, на которой расположен только текст, однако на множество типов дизайна – особенно на текучие макеты – дополнительное пространство плохо влияет.

Часто задаваемые вопросы

Что произойдет со значениями, которые браузер не понимает?

Браузеры IE и Netscape игнорируют атрибуты, которые они не понимают, поэтому, вообще говоря, вы не создадите проблем с совместимостью, установив все четыре атрибута.

Добавление дополнительного пространства можно отключить, установив значения пары атрибутов тега body в 0. От браузера зависит, какие атрибуты использовать. Для браузера Internet Explorer соответствующими атрибутами являются `leftmargin` и `topmargin`:

```
<body leftmargin="0" topmargin="0">
```

Для браузеров Netscape такими атрибутами являются `marginwidth` и `marginheight`:

```
<body marginwidth="0" marginheight="0">
```

Если вы находчивы, то используете все четыре атрибута:

```
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
```

С другой стороны, можно наоборот создавать дополнительное пространство, устанавливая эти атрибуты в ненулевые значения. Атрибуты `leftmargin` и `topmargin` в браузере IE управляют размером пространства слева и сверху окна браузера, соответственно, но ничего не делают справа и снизу, в то время, как атрибут `marginwidth` браузера Netscape обеспечивает равные пространства слева и справа, а атрибут `marginheight` – сверху и снизу.

Помните, что браузер IE не понимает атрибуты браузера Netscape, и наоборот, поэтому нельзя получить дополнительное правое поле, установив значение атрибута `marginwidth` в браузере IE. Подобным образом нельзя оказать влияние на левую сторону окна браузера, всего лишь установив значение атрибута `leftmargin` в браузере Netscape.

Для большего уровня управления полями, попытайтесь использовать таблицы CSS вместо языка HTML (обратитесь к главе 26).

ГЛАВА 26.
Управление пространством дизайна при помощи таблиц CSS

Как видно из главы 25, язык HTML предоставляет несколько атрибутов тега body для управления стандартными значениями пространства в окне браузера, однако эти атрибуты являются частными – они работают только в определенных браузерах. Браузеры полностью игнорируют атрибуты, которые они не понимают.

Вы получаете большее управление над полями при использовании каскадных таблиц стилей (CSS). Просто определите стиль для тега body на странице, наподобие следующей строки кода:

```
<body style="margin: 0px;">
```

Эта строка кода устанавливает все четыре поля – верхнее, левое, правое и нижнее – в 0 пикселов, тем самым исключая использование на странице любых пространств со значениями по умолчанию.

Совет

Ирония заключается том, вам нужно следить за управлением полями при помощи таблиц CSS, когда страница включает CSS-макеты. Симпатия к пространству полей может сбить положения ваших разделов. Вам может понадобиться откорректировать значения их позиций (слева, сверху, справа и снизу) после установки полей.

Таблицы CSS не ограничиваются только этим. При помощи атрибутов margin-top, margin-left, margin-right и margin-bottom можно легко определить различные размеры пространств для различных полей. Макет, представленный на рисунке 26.1, имеет верхнее поле, равное 200 пикселам, левое поле, равное 300 пикселам, правое поле, равное 400 пикселам, и нижнее поле, равное 100 пикселам.

Листинг 26.1. Просмотр исходного кода для рисунка 26.1

```
<body style="margin-top: 200px; margin-left: 300px; margin-right:
400px; margin-bottom: 100px;">
```

Совет

Для краткости можно задать размеры всех полей в атрибуте margin, выполнив обход по часовой стрелке (сверху, справа, снизу и слева), наподобие следующего:

```
<body style="margin: 200 400 100 300px;">
```

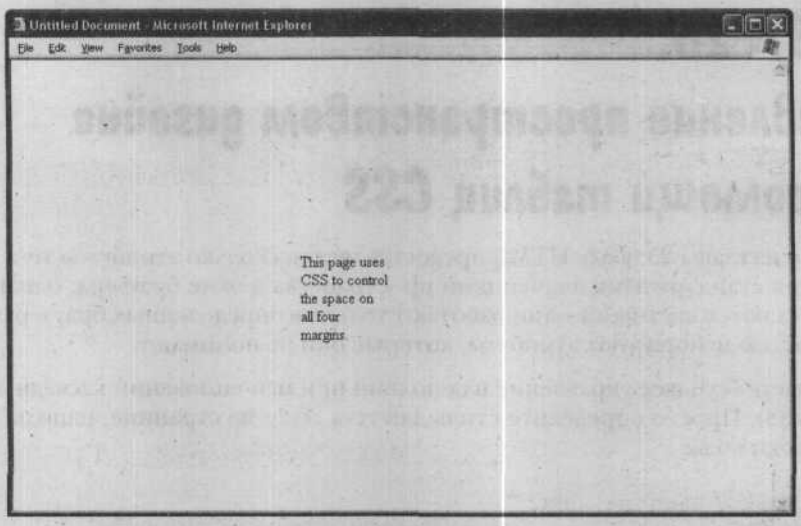


Рис. 26.1. Эта страница использует таблицы CSS для управления размером пространств для всех четырех полей

ЧАСТЬ 3.
Вопросы, касающиеся изображений

- Выбор правильного типа файла изображения
- Оптимизация Web-изображений
- Трансформация изображений с использованием атрибутов языка HTML
- Размещение внутристрочных изображений при помощи языка HTML
- Плавающие внутристрочные изображения с использованием таблиц CSS
- Разработка состояний кнопок
- Создание ролловерной графики
- Улучшение доступности изображений
- Создание карт ссылок
- Улучшение доступности карт ссылок
- Использование фоновых изображений в ячейках таблицы
- Использование фоновых изображений в CSS-разделах
- Использование фоновых изображений на Web-страницах

ГЛАВА 27.
Выбор правильного типа файла изображения

Файлы изображений используют множество различных форматов, однако только некоторые из этих форматов хорошо работают в Интернете. Браузеры, отображающие графику, стремятся поддерживать наиболее распространенных типов файлов, однако, как можно предположить, браузеры не договариваются между собой о том, какие точно типы графических файлов отображать. Браузер компании Microsoft, например, поддерживает растровые файлы операционной системы Windows (BMP), которые не поддерживает браузер Netscape.

Чтобы решить, какие форматы изображений использовать на своем сайте, ответьте на два вопроса:

1. Какие типы файлов поддерживаются всеми основными браузерами?
2. Какие из тех типов файлов создают файлы с наименьшими размерами, не принося в жертву качество изображения?

Ответ на первый вопрос прост. Все основные браузеры отображают изображения в форматах **GIF**, **JPEG** и **PNG**, поэтому нам нужно ограничить наши Web-изображения этими тремя форматами.

Совет

*Типы файлов **GIF** и **PNG** очень похожи. Они используются для одних и тех же видов изображений и дают практически одинаковый размер файла. Много лет назад основное отличие заключалось в том, что формат **GIF** был собственностью компании CompuServe, а формат **PNG** никогда не был чьей-либо собственностью. Однако во всем мире закончилось действие патента на использование формата **GIF** и теперь формат принадлежит миру.*

Ответом на второй вопрос является пугающее «Это зависит от ...». Вообще говоря, фотографии и другие изображения, содержащие большое количество информации о цвете, наилучшим образом работают с форматом **JPEG**, а штриховые иллюстрации и изображения с большими областями сплошного цвета лучше работают с форматами **GIF** или **PNG**. Почему? Файлы в формате **JPEG** не имеют встроенной **палитры** или таблицы цветов, которую имеют файлы в формате **GIF** и **PNG**. Палитра формата **GIF** или **PNG** определяет, какие цвета используются в изображении, максимум числа которых составляет 256 различных оттенков. При добавлении в палитру большего количества цветов общий размер файла увеличивается. Поскольку фотографии обычно имеют множество едва различимых оттенков и градаций, места для 256 цветов становятся занятыми довольно быстро. В итоге вы получаете файл в формате **GIF** или **PNG**, который выглядит зернистым, и его размер больше, чем размер соответствующего файла в формате **JPEG**, поскольку его палитра полностью заполнена.

Определение

*Встроенная в файл изображения таблица цветов. В файлы в формате **GIF** и **PNG** входит палитра, содержащая до 256 цветов, тогда как файлы в формате **JPEG** не имеют палитр и поэтому не ограничены 256 цветами.*

Однако если в вашем изображении используется меньше 256 цветов, файлы в формате **GIF** и **PNG** обычно имеют меньший размер, при этом сохраняя общее качество изображения.

Совет

*Непременно перемешивайте и сочетайте типы изображений на своем сайте. Не попадайте в западню, используя только изображения в формате **JPEG** или **GIF**. Выбирайте формат, который лучше всего подходит для каждого конкретного изображения. Профессиональное программное обеспечение для работы с графикой, например, Adobe Photoshop и Macromedia Fireworks, позволяет сравнивать рядом различные форматы ваших изображений, поэтому вам не придется гадать.*

Даже при этих условиях рассматривается не только число цветов, поскольку различные форматы также предоставляют различные возможности. Форматы **GIF** и **PNG** позволяют определять прозрачные цвета, а формат **GIF** позволяет создавать анимацию. Формат **JPEG** не позволяет ничего из этого, вы можете это увидеть в таблице 27.1. И некоторые фотографии отлично выглядят в формате **GIF** или **PNG** даже с потерей информации о цвете. Вам следует попробовать свои изображения во всех трех форматах, чтобы получить наилучший Web-файл из возможных.

Табл. 27.1. Распространенные типы файлов Web-изображений

Тип файла	Произносится	Обозначает	Палитра	Прозрачность	Анимация
GIF	**джифф** или **гифф**	Graphics Interchange Format (Формат графического обмена)	Есть; максимум 256 цветов	Есть; один уровень прозрачности	Есть
JPEG	**джэйпег**	Joint Photography Experts Group (Объединенная группа экспертов в области фотографии)	Нет; нет ограничения по цвету	Нет	Нет
PNG	**пинг**	Portable Network Graphics (Переносимая сетевая графика)	Есть; максимум 256 цветов	Есть; несколько уровней прозрачности	Нет

ГЛАВА 28.
Оптимизация Web-изображений

Оптимизация графического файла для Web означает максимально возможное уменьшение размера файла, при котором качество изображения сохраняется без изменений. Это – важный этап при создании Web-изображений, поскольку всегда следует помнить о времени загрузки страниц вашего сайта. И хотя все больше и больше людей используют высокоскоростные DSL- и кабельные линии для подключения к Интернету, обычные старые модемы для коммутируемого доступа не вышли из употребления, и графические файлы по-прежнему являются самым узким местом пропускной способности. Независимо от того, сколько изображений используется на вашем сайте, вы заметите улучшение производительности, оптимизировав изображения специально для Интернета.

В то же время, вам не нужно уменьшать размер файла настолько, что ухудшится качество изображения. Быстро загружающиеся изображения, которые выглядят отвратительно, хуже красивых изображений, которые загружаются слишком долго. Не сделайте ошибку: ваши посетители хотят видеть изображения. Посадите их перед чем-нибудь, что напоминает телевизор, и они будут ожидать просмотра картинок. Помните, что Интернет не мог превратиться в основное направление до тех пор, пока не перешел от страниц с педантичным HTML-текстом к графическому дизайну с полноцветными изображениями. Если ваши изображения выглядят ступенчатыми, низкого качества, зернистыми или неразборчивыми, посетители сделают соответствующие выводы о качестве информации на вашем сайте.

Определение

***Оптимизация** файла изображения для Web означает уменьшение размера файла для более быстрой загрузки, при котором качество изображения сохраняется без изменений.*

В этой главе предложено несколько способов оптимизации файлов изображений для Web с использованием практически любого самого элементарного приложения, предназначенного для редактирования графических файлов. Если ваше приложение для работы с графикой не позволяет выполнять представленные операции, пора подумать о новой версии программного обеспечения. Здесь нет необходимости брать банк. Программа Adobe Photoshop Elements, упрощенная версия дедушки Photoshop, является отличным инструментом для работы с Web-изображениями, и стоит всего $100. Еще одной хорошей программой является программа Jasc Paint Shop Pro, которая стоит почти столько же. Если разработка Web-сайтов – это ваша профессия, попытайтесь уговорить вашего работодателя на покупку полной версии программы Photoshop или Macromedia Fireworks. Цена на эти приложения больше, но они стоят того.

Совет

Более профессиональные приложения для работы с графикой предоставляют ряд других методов для Web-оптимизации, помимо методов, описанных в этой главе. За дополнительной информацией обратитесь к документации или справочной системе конкретного приложения.

Оптимизация разрешения

Разрешение файла изображения определяет размер его пикселов: чем выше разрешение, тем меньше размер пикселов и, следовательно, точнее изображение. Однако чем больше пикселов используется в изображении, тем больше становится его размер и тем больше времени необходимо для его загрузки.

В мире фотографических изображений более высокое разрешение всегда лучше, чем более низкое. Но в случае с Web-изображениями возможности отображения экранов мониторов накладывают жесткое ограничение на разрешение изображений. Мониторы компьютеров, работающих под управлением операционной системы Windows, отображают информацию с разрешением 96 ppi (пикселов на дюйм), что, на самом деле, является достаточно низким разрешением. Мониторы компьютеров Macintosh отображают информацию с еще более низким разрешением, 72 ppi. В духе кросс-платформенной совместимости, разрешение72 ppi стала стандартом для Web-изображений, хотя подавляющее большинство мониторов могут отображать информацию с немного более высоким значением, 96 ppi.

Определение

Разрешение файла изображения определяет размер его пикселов: чем больше разрешение, тем меньше пикселы и резче изображение.

Это означает следующее: на экране изображения с высоким разрешением выглядят не лучше изображений, имеющих низкое разрешение – 72 или 96 ppi. Если ваши Web-изображения имеют фотографическое качество, скажем, 300 или 600 ppi, вся эта дополнительная четкость является мертвым грузом. Посетители не заметят никакой разницы на экране, что и показано на рисунке 28.1. Зачем заставлять посетителей загружать изображения с высоким разрешением, когда с тем же успехом работают быстрее загружающиеся изображения с низким?

Часто задаваемые вопросы

Почему оптимизированные Web-изображения выглядят зернистыми при распечатке?

На экране изображение с разрешением 72 ppi выглядит резким и четким, однако это происходит потому, что мониторы являются устройствами вывода с достаточно низким разрешением. Старая добрая бумага на самом деле работает с устройством вывода с гораздо более высоким разрешением. На одном и том же пространстве на печатной странице можно уместить больше точек, поэтому изображения с низким разрешением более отчетливо показывают свою зернистость.

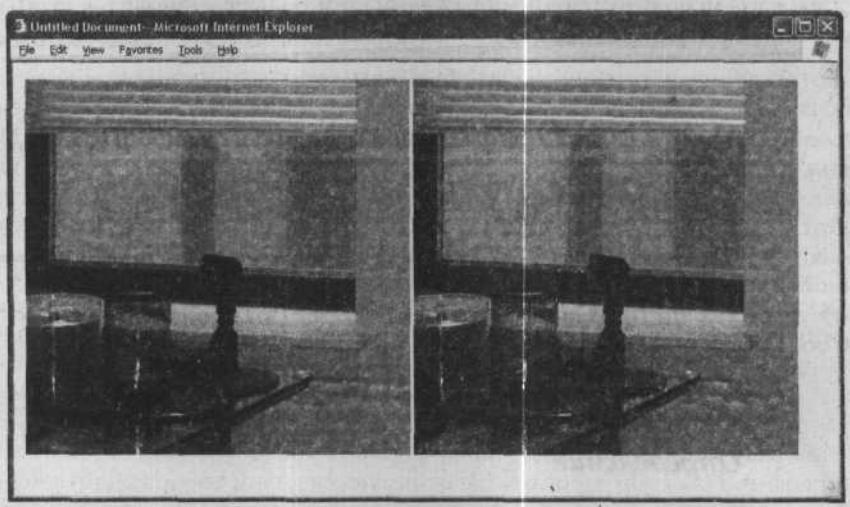


Рис. 28.1. Изображение слева имеет разрешение 600 ppi, а изображение справа – 72 ppi – более, чем в восемь раз, низкую. Однако оба изображения выглядят на экране совершенно одинаково, поскольку мониторы имеют относительно низкое разрешение

Совет

Изображения, специально предназначенные для печати посетителями, должны быть оптимизированы для печати, а не для экрана. Однако не все, кто посетит ваш сайт, будут распечатывать эти изображения. Чтобы улучшить производительность сайта, не встраивайте изображения с высоким разрешением в страницы. Вместо этого используйте изображения с низким разрешением, оптимизированные для экрана, и включите ссылки на изображения с высоким разрешением, имеющие фотографическое качество, для тех, кому они нужны. Изображения с высоким разрешением не выглядят по-другому на экране, однако они гораздо лучше перенесутся при печати, чем их копии с низким разрешением.

Используя графический редактор, уменьшите разрешение всех Web-изображений до стандартного разрешения 72 ppi. Это самый простой и наиболее эффективный метод оптимизации изображений для вывода на монитор.

Оптимизация размера изображения

Пространство экрана напоминает физическое пространство в центре Манхэттена – бессмысленно дорогое. Вы получите больше пользы от своего Web-сайта, если будете использовать изображения с меньшими физическими размерами (т.е. меньшими значениями ширины и высоты). Меньшие изображения загружаются быстрее, чем большие. Кроме того, используя меньшие изображения, на экране можно разместить больше информации, перед тем как пользователи пролистают страницу, и сайт будет лучше выглядеть на компактных дисплеях портативных устройств.

Совет

При уменьшении изображения пропустите изображение с измененными размерами через фильтр, увеличивающий резкость, чтобы показать подробности.

И хотя меньшие изображения лучше, не используйте браузер для изменения размеров изображения. В языке HTML вполне выполнимо превратить изображение с размерами 300 на 150 пикселов в, скажем, изображение с размерами 60 на 30 пикселов, настроив значения атрибутов `width` и `height` тега `img` (за более подробной информацией обратитесь к главе 29). Но если вы желаете отобразить изображение с размерами 60 на 30 пикселов, запустите свой графический редактор и сделайте копию изображения с такими размерами. Иначе браузеру все равно придется загружать больший файл изображения с размерами 300 на 150 пикселов, и вы ничуть не уменьшите время загрузки.

Аналогично, не вставляйте в файл изображения чрезвычайно большие объемы пустого пространства. Вместо этого обрежьте изображение до размеров, которые вы желаете отобразить, и включите пустое пространство в дизайн страницы.

Оптимизация палитр файлов в форматах GIF и PNG

Как вы уже знаете из главы 27, файлы в форматах **GIF** и **PNG** имеют встроенную палитру, в которой может храниться до 256 цветов. Чем больше цветов хранится в палитре, тем больше размер файла изображения. Если вашему изображению в формате **GIF** или **PNG** необходимы все 256 цветов, так тому и быть. Однако для большинства файлов в форматах **GIF** и **PNG** можно избавиться от некоторых цветов, незначительно повлияв на качество изображения.

Будьте осторожны с изображениями, содержащими закругленные или криволинейные фигуры. В таких изображениях часто используется метод, называемый **сглаживанием**, который создает иллюзию плавности путем вставки слегка размытых пикселов вдоль криволинейного края. Сглаживание минимизирует естественное блочное отображение пикселов, имеющих квадратную форму, и обманывает глаз, который видит гладкие, плавные кривые. Для каждого размытого оттенка необходимо место в палитре изображений в формате **GIF** или **PNG**. Удаление большого количества цветов из палитры снижает эффективность сглаживания.

Определение

Сглаживание – это метод создания видимости гладких, плавных кривых путем вставки слегка размытых пикселей вдоль криволинейного края.

Оптимизации сжатия изображений в формате JPEG

Изображения в формате **JPEG** не имеют палитр, которые имеют изображения в формате **GIF** и **PNG**. Вместо этого формат **JPEG** сжимает информацию в изображении, делая ее более эффективной и отбрасывая информацию, которую человеческий глаз скорее всего не заметит. Слишком большое сжатие избавляется от огромного количества информации, и качество изображения заметно ухудшается.

Хорошие графические приложения позволяют указывать степень сжатия файла в формате **JPEG** путем настройки его качества. Не существует твердого правила для настройки наилучшего качества. Все зависит от количества информации в файле в формате **JPEG**.

Определение

*Сжатие в графическом формате, например, **JPEG**, уменьшает размер файла изображения, отбрасывая часть визуальной информации. Слишком большое сжатие избавляет от огромного количества информации и качество изображения заметно ухудшается.*

Для оптимизации файла в формате **JPEG** поиграйте с уровнем сжатия и посмотрите, какое значение качества подходит лучше всего. Обычно существует точка, за которой качество изображения существенно ухудшается. Когда вы достигли этой точки, просто вернитесь назад на один уровень сжатия.

ГЛАВА 29.
Трансформация изображений с использованием атрибутов языка HTML

Браузеры, поддерживающие изображения, могут делать больше, чем просто отображать графику. Они также могут изменять их физические размеры.

При добавлении изображения на страницу, как правило, указываются точные размеры изображения с использованием атрибутов `width` и `height` тега `img`, как показано на рисунке 29.1.

Листинг 29.1. Просмотр исходного кода для рисунка 29.1

```
<img src="logo.jpg" width="150" height="75">
```

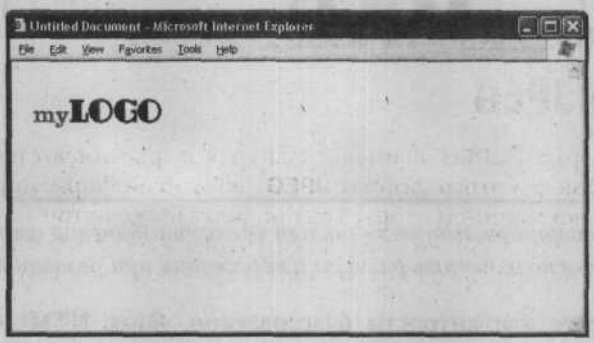


Рис. 29.1. Укажите точные размеры изображения в теге `img`, и изображение появится на странице в точности, как вы разработали его в приложении, предназначенном для работы с графикой

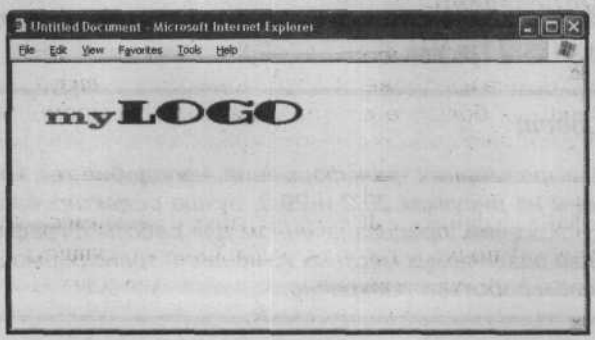


Рис. 29.2. Изменив значения атрибутов `width` и `height`, вы трансформируете изображение при помощи языка HTML

Но кто сказал, что вы обязаны указывать реальные размеры изображения? Можно трансформировать изображение, растянув его ширину или высоту, как показано на рисунке 29.2.

Листинг 29.2. Просмотр исходного кода для рисунка 29.2

```
<img src="logo.jpg" width="300" height="35">
```

И пропорционально увеличив или уменьшив значения атрибутов `width` и `height`, можно изменить размеры изображения, как показано на рисунке 29.3.

Листинг 29.3. Просмотр исходного кода для рисунка 29.3

```
<img src="logo.jpg" width="300" height="150">
```

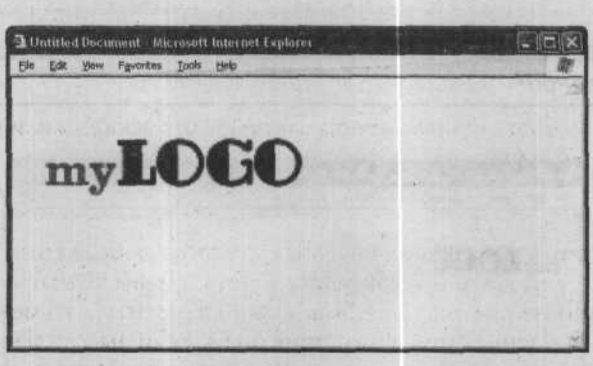


Рис. 29.3. Пропорционально увеличив или уменьшив значения атрибутов `width` *и* `height`*, можно изменить размеры изображения при помощи языка HTML*

Однако используйте эти хитрости благоразумно. Язык HTML – это программа Photoshop для бедных, и вы определенно получите то, за что заплатили. Как правило, результаты трансформации изображения в специализированной программе для работы с графикой выглядят лучше. И если сделать изображение меньшим при помощи языка HTML, это не изменит времени его загрузки, как и если сделать изображение большим, это не увеличит времени загрузки. Браузер всегда загружает исходный файл с нормальными размерами, независимо от того, как вы трансформируете это изображение при помощи языка HTML.

Совет

Для чрезмерных трансформаций, наподобие тех, которые представлены на рисунках 29.2 и 29.3, лучше редактировать изображения в приложении, предназначенном для работы с графикой. Однако для едва различимых местных изменений трансформации языка HTML в целом работают прекрасно.

ГЛАВА 30.
Размещение внутристрочных изображений при помощи языка HTML

Внутристрочные изображения – это изображения, отображаемые внутри блока текста. Ими в основном являются изображения с содержимым, например, диаграммы, графики и фотографии, в противоположность логотипам, кнопкам навигационной панели и другим элементам графического интерфейса, которые обычно имеют свои собственные ячейки таблиц или CSS-разделы в макете страницы.

Совет

***Внутристрочные изображения** – это изображения, отображаемые внутри блока HTML-текста.*

Простейшим способом добавления внутристрочного изображения в языке HTML является вставка тега `img` как раз в том месте текста, где вы хотите отобразить изображение, как показано на рисунке 30.1. Как можно заметить, это метод является работоспособным, но не очень эстетически приятным. Если высота изображения больше высоты окружающего текста в пунктах, браузер разделит строки текста самым непривлекательным способом, который можно себе представить.

Листинг 30.1. Просмотр исходного кода для рисунка 30.1

```
<h2>A complete brain-trust solution</h2>

<p>When you consult with us, you'll be amazed at the results. We offer
the most interesting, creative, out-of-the-box, and forward-thinking
ideas <img src="images/bulb.gif" width="47" height="84"> in the
industry. You won't believe what we'll come up with next. We have won
several international awards for our ideas alone, including the pres-
tigious Nobel Prize. You simply won't find better ideas anywhere.</p>
```

Атрибут `align` тега `img` помогает разместить изображение более эффективно, как показано на рисунке 30.2. Просто выберите значение атрибута `align` из таблицы 30.1. Можно сделать так, чтобы изображения **плавало** с левой или правой стороны всего блока текста, или можно выровнять изображение вертикально по верху, центру или низу строки текста, в которой отображается внутристрочное изображение.

Определение

Сделать внутристрочное изображение плавающим означает разместить его с левой или правой стороны блока текста, чтобы текст обтекал это изображение.

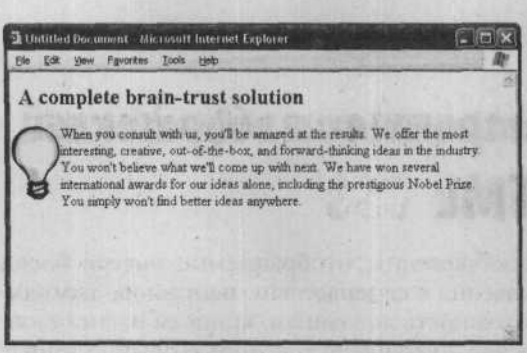


Рис. 30.1. Размещение тега `img` внутри блока текста создает внутристрочное изображение, однако браузер разделяет строки текста без всякой искусной утонченности

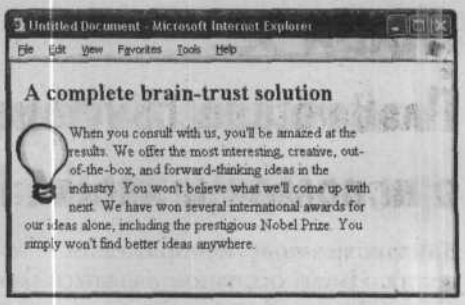


Рис. 30.2. Использование атрибута `align` тега изображения сообщает браузеру, как разместить изображение относительно окружающего текста

Листинг 30.2. Просмотр исходного кода для рисунка 30.2

```
<h2>A complete brain-trust solution</h2>

<p><img src="images/bulb.gif" width="47" height="84" align="left">
When you consult with us, you'll be amazed at the results. We offer
the most interesting, creative, out-of-the-box, and forward-think-
ing ideas in the industry. You won't believe what we'll come up with
next. We have won several international awards for our ideas alone,
including the prestigious Nobel Prize. You simply won't find bet-
ter ideas anywhere.</p>
```

Табл. 30.1. Значения атрибута `align` тега `img`

Значение	Результат
`bottom`	Вертикально выравнивает низ изображения по низу или по базовой линии строки текста
`left`	Делает изображение плавающим с левой стороны блока текста
`middle`	Вертикально выравнивает изображение по центральной строке текста
`right`	Делает изображение плавающим с правой стороны блока текста
`top`	Вертикально выравнивает верх изображения по верху строки текста

Совет

Чтобы внутристрочное изображение плавало с левой или правой стороны блока текста, переместите тег `img` в начало блока текста, как показано в листинге 30.2.

ГЛАВА 31.
Плавающие внутристрочные изображения с использованием таблиц CSS

Как и со всеми вопросами, связанными с дизайном и форматированием, в наши дни политически корректным методом для создания плавающих внутристрочных изображений является не присваивание значений `left` или `right` атрибуту `align` тега `img` (за дополнительной информацией обратитесь к главе 30), а использование каскадных таблиц стилей (CSS). Положительным моментом является то, что таблицы CSS предоставляют больше возможностей, чем атрибут `align`. Отрицательным моментом является то, что браузеры регулярно теряют сознание от таблиц CSS везде, где находят их, поэтому необходимо проверять свои CSS-стили на множестве браузеров.

Первый шаг заключается в превращении внутристрочного изображения в плавающее рядом с блоком текста, как показано на рисунке 31.1. Как можно догадаться, определение CSS-стиля `float: left;` размещает изображение с левой стороны блока текста, а определение `float: right;` размещает изображение с правой стороны.

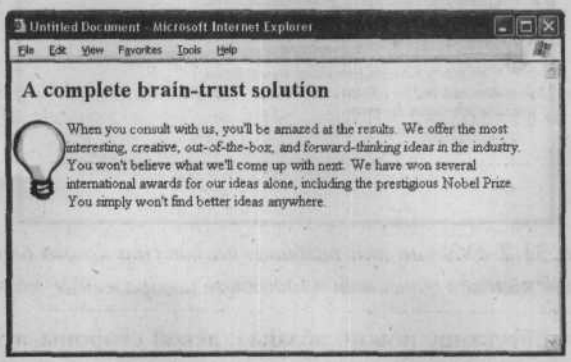


Рис. 31.1. Достаточно просто превратить внутристрочное изображение в плавающее при помощи таблиц CSS

Листинг 31.1. Просмотр исходного кода для рисунка 31.1

```
<h2>A complete brain-trust solution</h2>

<p><img src="images/bulb.gif" width="47" height="84" style="float:
left;">When you consult with us, you'll be amazed at the results.
We offer the most interesting, creative, out-of-the-box, and for-
ward-thinking ideas in the industry. You won't believe what we'll
come up with next. We have won several international awards for our
ideas alone, including the prestigious Nobel Prize. You simply won't
find better ideas anywhere.</p>
```

Пока все хорошо, но таблицы CSS не дали нам ничего, чего бы мы не смогли сделать при помощи атрибута `align` тега `img`. Веселье начинается, когда сплошной блок текста разбивается на две или более части, как показано на рисунке 31.2. Вы сможете изменить способ обтекания плавающего изображения дополнительными блоками текста.

Листинг 31.2. Просмотр исходного кода для рисунка 31.2

```
<h2>A complete brain-trust solution</h2>

<img src="images/bulb.gif" width="47" height="84" style="float:
left;">

<p>When you consult with us, you'll be amazed at the results. You
won't believe what we'll come up with next.</p>

<p>We have won several international awards for our ideas alone,
including the prestigious Nobel Prize. You simply won't find bet-
ter ideas anywhere.</p>
```

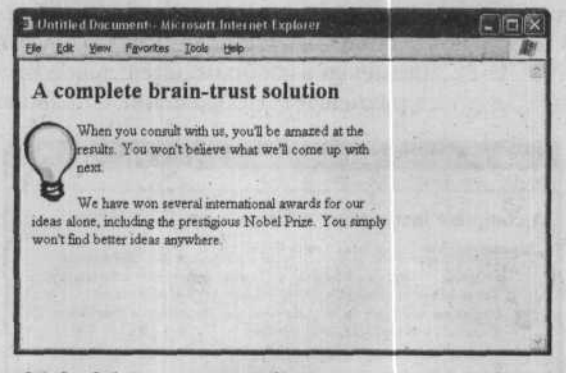


Рис. 31.2. Обычно при разбивании текста на два блока, второй блок красиво обтекает плавающее изображение, как и первый

Так, если запретить обтекание нового абзаца с левой стороны, то он будет расположен прямо под левым плавающим изображением вместо того, чтобы обтекать это изображение, как показано на рисунке 31.3.

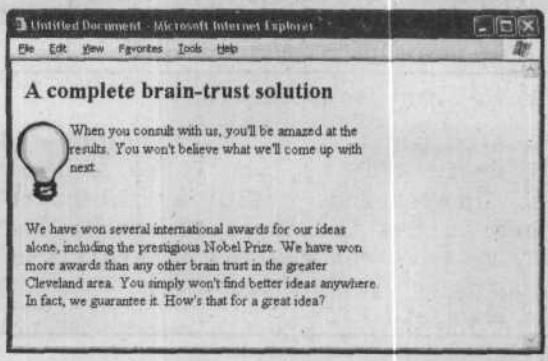


Рис. 31.3. Если запретить обтекание второго абзаца с левой стороны, текст будет вынужден располагаться прямо под левым плавающим изображением

Листинг 31.3. Просмотр исходного кода для рисунка 31.3

```
<h2>A complete brain-trust solution</h2>

<img src="images/bulb.gif" width="47" height="84" style="float:
left;">

<p>When you consult with us, you'll be amazed at the results. You
won't believe what we'll come up with next.</p>

<p style="clear: left;">We have won several international awards for
our ideas alone, including the prestigious Nobel Prize. You simply
won't find better ideas anywhere.</p>
```

Чтобы увидеть, что здесь происходит, добавим еще одно внутристрочное изображение, и на этот раз сделаем его плавающим с правой стороны, как показано на рисунке 31.4. Второй абзац начинается прямо под левым плавающим изображением, однако он обтекает правое плавающее изображение.

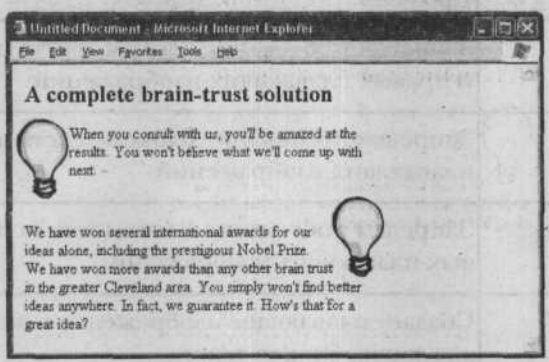


Рис. 31.4. После добавления левого плавающего изображения второй абзац обтекает его, поскольку было указано, что абзац не должен обтекать только изображения слева

Листинг 31.4. Просмотр исходного кода для рисунка 31.4

```
<h2>A complete brain-trust solution</h2>

<img src="images/bulb.gif" width="47" height="84" style="float:
left;">

<p>When you consult with us, you'll be amazed at the results. You
won't believe what we'll come up with next.</p>

<img src="images/bulb.gif" width="47" height="84" style="float:
right;">

<p style="clear: left;">We have won several international awards for
our ideas alone, including the prestigious Nobel Prize. We have won
more awards than any other brain trust in the greater Cleveland
area. You simply won't find better ideas anywhere. In fact, we guar-
antee it. How's that for a great idea?</p>
```

Совет

Чтобы запретить обтекание правого плавающего изображения, используйте определение `clear: right;`.

Если вы хотите для второго абзаца запретить обтекание плавающих изображений справа и слева, используйте определение стиля `clear: both;`.

В таблице 31.1 приведены CSS-стили для создания плавающих изображений и для запрещения обтекания.

Табл. 31.1. Создание плавающих изображений и запрещение обтекания при помощи таблиц CSS

Стиль	Результат
`clear: both;`	Запрещает обтекание блоком текста предыдущих левых и правых плавающих изображений
`clear: left;`	Запрещает обтекание блоком текста предыдущих левых плавающих изображений
`clear: right;`	Запрещает обтекание блоком текста предыдущих правых плавающих изображений
`float: left;`	Создает плавающее изображение с левой стороны последующих блоков текста
`float: right;`	Создает плавающее изображение с правой стороны последующих блоков текста

ГЛАВА 32.
Разработка состояний кнопок

Состояния кнопки – это различные представления, которые принимает графическое изображение кнопки в зависимости от вида взаимодействия посетителя с кнопкой. Для Web-сайта, основанного на языке DHTML, обычно определено два состояния кнопки: состояние по умолчанию, определяющее, как выглядит кнопка в обычных условиях; и состояние `onMouseOver`, определяющее, как выглядит кнопка, когда посетитель наводит на нее указатель мыши. (Ролловерная функция, написанная на языке JavaScript, производит переключение между этими двумя состояниями.) Сайты, построенные на базе технологии Macromedia Flash, обычно включают третье состояние: состояние `onClick`, определяющее, как выглядит кнопка, когда посетитель щелкает на ней.

Определение

***Состояния кнопки** – это различные представления, которые принимает графическое изображение кнопки в зависимости от вида взаимодействия посетителя с кнопкой.*

Состояния кнопки обеспечивают бесценную визуальную обратную связь на обычной, наводненной ссылками, Web-странице. Когда такой большой объем содержимого переполняет типичное окно браузера, выполнение действий в ответ на перемещение указателя мыши или щелчки помогают посетителям понять, как использовать ваш сайт.

Часто задаваемые вопросы

Что за странная терминология для состояний кнопки?

Виноваты язык JavaScript. Имена для событий кнопки `onMouseOver` и `onClick` произошли от соответствующих событий в языке JavaScript.

Когда наступает время разрабатывать состояния для своих кнопок, мы хотим извлечь выгоду из психологии того, что происходит. Другими словами, мы хотим, чтобы визуальные эффекты соответствовали направлению, в котором работает разум посетителя. Не нужно противоречить психологии – делать обратное тому, что ожидает разум посетителя, – иначе ваш сайт будет посылать сбивающие с толку сообщения, и визуальная обратная связь будет создавать путаницу, а не улучшать удобство использования.

Вот теория: разрабатывайте состояния кнопок с увеличением энергии. Каждое последующее состояние кнопки должно «подниматься на ступеньку выше», следуя логической последовательности, в которой посетитель взаимодействует с кнопкой. Следовательно, состояние по умолчанию должно иметь наименьшее количество энергии, поскольку посетитель совсем не взаимодействует с ним. Состояние

`onMouseOver` должно подняться на ступеньку выше, как будто кнопка впитывает кинетическую энергию, которую генерирует посетитель, перемещая мышь. И состояние `onClick` (если у вас есть такое состояние) должно подниматься еще на одну ступеньку, соответствующую дополнительной кинетической энергии, получаемой при щелчке мышью.

Все, что мы делаем здесь, – это графически дублируем способ функционирования реального объекта, наподобие механизма: появляется энергия и что-то происходит. Светодиод компьютера не гаснет, когда включается питание, и продолжает гореть всю ночь, пока не будет выключено питание. Это не создает наглядного восприятия. Сколько графических кнопок вы видели в Интернете до сих пор, которые как будто выключались при наведении на них мыши? Если вы разрабатываете кнопки, которые опускаются на ступеньку ниже, когда посетитель взаимодействует с ними, кажется, что кнопки поглощают энергию и противоречат законам физики, как плохая территория в фильме «Star Trek».

Когда появляется энергия, что-то происходит. Что точно будет происходить, зависит от вас, но в таблице 32.1 собраны некоторые предложения, а на рисунке 32.1 показано несколько последовательностей состояний кнопок.

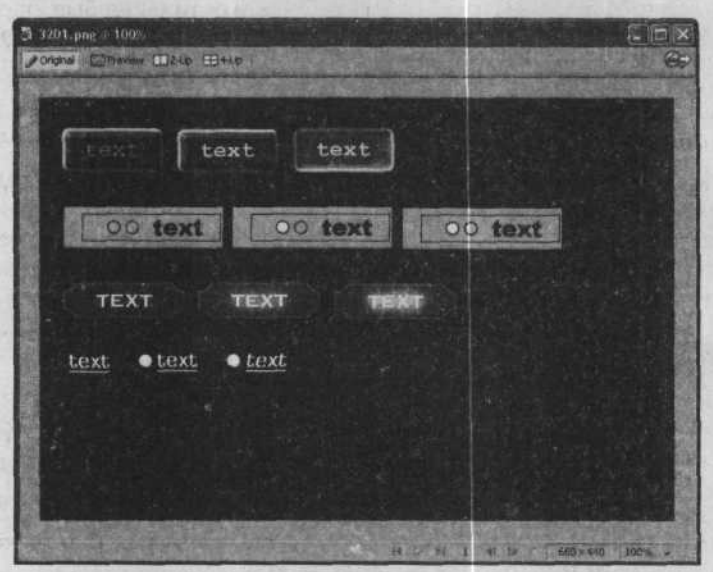


Рис. 32.1. В этих последовательностях состояний кнопок используется переход от низкой энергии к более высокой, чего интуитивно и ожидает посетитель

Табл. 32.1. Дизайнерские предложения для состояний кнопок

Состояние onMouseOver (Подсвечивание)	Состояние onClick (Активация)
• Сделать ярче цвет текстовой метки; • Сделать ярче цвет лицевой части кнопки; • Добавить рамку или свечение вокруг кнопки; • Подтолкнуть всю кнопку целиком или только текстовую метку внутри кнопки; • Подчеркнуть текстовую метку; • Добавить маркер с левой стороны текстовой метки; • Изменить форму или размер маркера текстовой метки (если маркер присутствует в состоянии по умолчанию); • Увеличить размер кегля текста метки.	• Сделать так, чтобы кнопка выглядела нажатой; • Сделать так, чтобы кнопка выглядела полностью включенной (если состояние `onMouseOver` выглядит не полностью включенным); • Создать анимацию быстрой вспышки цвета в элементе, который изменили или добавили в состояние `onMouseOver`; • Добавить маркер к текстовой метке (если он не был еще добавлен в состоянии `onMouseOver`); • Добавить подчеркивание текстовой метки (если оно не было добавлено в состояние `onMouseOver`).

Совет

Подойдите творчески к разработке дизайна для состояния `onMouseOver`, поскольку то, что мы хотим сделать, заключается в подсвечивании кнопки. Серьезно отнеситесь к состоянию `onClick` и используйте типичную кнопочную метафору нажатия – т.е. сделайте так, чтобы кнопка выглядела нажатой, и не важно к какой оригинальной идее вы пришли для состояния `onMouseOver`. Имитация нажатия кнопки является самым наглядным способом сообщить посетителям, что они только что активировали кнопку.

Совет

Не используйте дизайн нажатой кнопки для состояния `onMouseOver`, поскольку он наводит посетителя на мысль, что кнопка была активирована (т.е. на ней щелкнули), хотя на самом деле она была только подсвечена. Поберегите дизайн нажатых кнопок для состояния `onClick` для сайтов, построенных по технологии Flash.

ГЛАВА 33.
Создание ролловерной графики

Ролловерная графика – это изображения, которые изменяют свой вид, когда посетитель наводит на них указатель мыши. Этот эффект на самом деле является неким волшебным фокусом, поскольку само изображение не изменяется. Вместо этого существует два отдельных файла изображения, и JavaScript-функция вытягивает предыдущее состояние, меняя местами эти два изображения в зависимости от позиции указателя мыши.

Определение

***Ролловерная графика** – это изображение, которое изменяет свой вид, когда посетитель наводит на него указатель мыши.*

Итак, для создания ролловерной графики необходимы три вещи:

1. Файл изображения по умолчанию или нормальное состояние ролловера;
2. Файл изображения для состояния `onMouseOver` или изображение, которое появляется, когда посетитель наводит указатель мыши на ролловер;
3. JavaScript-функция, выполняющая смену изображений.

Для ролловера можно использовать любые два файла изображений. Они не обязательно должны выглядеть похожими. Они даже не обязаны иметь одинаковые физические размеры. И все-таки, ролловеры работают лучше всего, когда графические файлы являются различными версиями одного и того же изображения с одинаковыми размерами. Ролловерный эффект создает таким образом некий вид анимации, и вы получаете лучшую последовательность, поскольку у посетителя не создается впечатление, что изображение по умолчанию пропадает или радикально изменяется.

Совет

Ролловерная графика отлично подходит для кнопок навигационной панели. Дополнительная визуальная обратная связь, благодаря ролловерному эффекту, усиливает ощущение, что на кнопке можно щелкнуть.

JavaScript-функция является секретным ингредиентом, заставляющим работать ролловер. В основе функции лежит чрезвычайно полезный метод `getElementById`, который поддерживается самыми последними версиями наиболее популярных браузеров. Все достаточно просто: функция находит изображение, которое вы желаете сменить на другое, по его коду идентификатора, и затем изменяет источник этого изображения с версии по умолчанию на версию для состояния `onMouseOver`, или с версии для состояния `onMouseOver` обратно на версию по умолчанию.

Совет

Для получения наилучших результатов всегда указывайте значения атрибутов `width` и `height` тега `img` ролловерной графики. В таком случае, если изображение для состояния `onMouseOver` будет иметь другие размеры, чем изображение по умолчанию, браузер изменит размеры изображения для состояния `onMouseOver`, чтобы они соответствовали размерам изображения по умолчанию. Иначе браузер вставит изображение для состояния `onMouseOver` как есть и перерисует весь экран целиком. И еще более неприятно, если изменение размеров привело к тому, что указатель мыши оказался за пределами области изображения для состояния `onMouseOver`; ролловерная функция автоматически возвратит обратно изображение по умолчанию, и вы получите стробоскопический эффект.

Ролловерную функцию вызывают два различных события языка JavaScript. Событие `onMouseOver` переключает состояние по умолчанию на состояние, когда над изображением наведен указатель мыши, а событие `onMouseOut` переключает состояния обратно. Если бы не было события `onMouseOut` для возврата к обычному состоянию, ролловерное изображение застряло бы на состоянии, когда указатель мыши наведен над изображением, независимо от того, куда посетитель переместил указатель мыши.

Инструментарий

Ролловерная графика

Данный DHTML-документ демонстрирует создание ролловерного эффекта. `<html>`

```
<head>
<title>Rollover Graphics</title>
<script language="JavaScript">
function doRollover(imageID, imageSource) {
/* Эта функция принимает два параметра: imageID (ID
изображения, которое вы хотите заменить) и
imageSource (путь к файлу изображения, на которое
вы хотите произвести замену). Единственная
инструкция языка JavaScript использует оба параметра для выполнения трюка. */
  document.getElementById(imageID).src = imageSource;
}
</script>
</head>
<body>
<!— Для этой демонстрации здесь используется только
одно ролловерное изображение, однако тело документа
можно заполнить любым желаемым содержимым. —>
```

Инструментарий

```
<img id="imageid"
   src="imagepath"
   width="imagewidth"
   height="imageheight"
   onMouseOver="doRollover('rolloverid',
'mouseoverimagepath');"
   onMouseOut="doRollover('rolloverid', 'default-
imagepath');">
<!— Примечания:
imageid - это уникальная строка, идентифицирующая
изображение, например, img01 или happyface. Каждо-
му изображению на странице присваивайте различные
идентификаторы.
imagepath - это путь к изображению, например,
images/icecream.jpg. Этот путь может быть путем от-
носительно документа, путем относительно корня или
абсолютным путем.
imagewidth и imageheight - это ширина и высота
изображения в пикселах.
rolloverid - это идентификатор целевого изображе-
ния, которое является изображением для замены с ис-
пользованием ролловерной функции. Для обычного
ролловера, rolloverid совпадает с imageid. Для
разъединенного ролловера, rolloverid является
идентификатором любого изображения, которое вы хо-
тите изменить. Обязательно заключайте это значение
в одинарные кавычки.
mouseoverimagepath - это путь к версии изображения
для состояния, когда указатель мыши наведен на
изображение, например, images/icecream_ro.jpg.
Путь может быть путем относительно документа, пу-
тем относительно корня или абсолютным путем. За-
ключайте весь путь в одинарные кавычки.
defaultimagepath - это путь к версии изображения по
умолчанию. Опять-таки, путь может быть путем отно-
сительно документа, путем относительно корня или
абсолютным путем, и обязательно заключайте весь
путь в одинарные кавычки. Для обычного ролловера,
defaultimagepath совпадает с imagepath. Для разъе-
диненного ролловера, defaultimagepath совпадает со
значением атрибута src целевого изображения.
—>
</body>
</html>
```

Совет

Эту же функцию можно использовать для создания разъединенного ролловера, или изображения, которое инициирует роловерный эффект не для себя, а абсолютно другого изображения. Чтобы сделать это, определите события `onMouseOver` *и* `onMouseOut` *в инициирующем изображении, но в качестве параметра функции укажите идентификатор изображения, которое вы хотите изменить, а не собственный идентификатор инициирующего изображения.*

Совет

Несколько слов по терминологии: окажите себе услугу и назовите изображение для состояния `onMouseOver` *так же, как и изображение по умолчанию, добавив суффикс, например,* ***_ro*** *или* ***_over****. Так, если изображение по умолчанию называется* ***button.gif****, изображение для состояния* `onMouseOver` *будет называться* ***button_ro.gif*** *или* ***button_over.gif****. Это поможет вам отслеживать изображения.*

Конечно, различные дизайнеры имеют различные предпочтения по именованию своих изображений. Один из редакторов данной книги предпочитает систему, подобную следующей: имя ***NavHomeOff.gif*** *для изображения по умолчанию и имя* ***NavHomeOn.gif*** *для ролловера. Используйте любую систему, которая имеет для вас смысл. Но все дизайнеры согласны, что необходимо использовать либо ту, либо другую систему.*

Об идентификаторах: если вы не можете определить, какие идентификаторы использовать, просто воспользуйтесь общим префиксом, например, ***img****, и пронумеруйте ваши изображения подобным образом:* ***img01****,* ***img02****,* ***img03*** *и так далее. В качестве идентификаторов можно использовать имена файлов изображений, но только в том случае, если такое изображение встречается на странице всего один раз. Помните, каждое изображение должно иметь уникальный идентификатор.*

ГЛАВА 34.
Повышение доступности изображений

Добросовестные Web-дизайнеры делают свои Web-сайты максимально совместимыми со средствами специальных возможностей, например, читателями экрана, преобразователями текст-речь и преобразователями текст-шрифт Брайля, чтобы помочь инвалидам по зрению.

Сделать HTML-текст доступным просто – вам вообще ничего не нужно делать для этого. Средства специальных возможностей достаточно сообразительны, чтобы отыскать текст на странице и преобразовать его в произносимые слова или шрифт Брайля. Однако пока не существует технологии для таких средств, позволяющей исследовать содержимое изображения и создать доступное описание. Вам, создателю Web-сайтов, выпадает участь писать такие описания самому и вставлять их в HTML-код, чтобы читатели экрана и подобные им могли найти эти описания.

Описание помещается в значение атрибута `alt` тега `img`, как в следующем коде:

```
<img src="neptune.jpg" width="300" height="150" alt="Изображение планеты Нептун, полученное с помощью космического телескопа Hubble. Поверхность Нептуна кажется кобальтово-синей яркими полосами серо-голубых облаков.">
```

Совет

В качестве дополнительного бонуса, как могут сказать в отделе рекламы, некоторые поисковые машины специально ищут текст в атрибуте `alt`, чтобы более точно индексировать ваш сайт. Поэтому добавление осмысленного текста для атрибута `alt` может помочь в повышении ранга вашего сайта среди результатов поиска.

Обратите внимание, что в описании не урезаются детали. Идея заключается в предоставлении инвалидам по зрению сопоставимого впечатления об изображении. Это означает, что мы хотим передать в описании ту же информацию, которую визуально представляет изображение. Поэтому, когда вы будете добавлять текстовые описания для атрибута `alt` ваших изображений, вообразите, что пишете сочинение. Сделайте так, чтобы ваше описание вызывало изображение в мысленном взгляде.

В предыдущем примере вы видели, что текст выполняет достаточно хорошую работу для подачи визуальной информации. Вам нет смысла добавлять описание, подобное следующему:

```
<img src="neptune.jpg" width="300" height="150" alt="Планета Нептун">
```

Такой немногословный текст может вызвать споры относительно точности данного описания, поскольку в нем недостаточно деталей для вызова изображения Нептуна в мысленном взгляде. Инвалиды по зрению, которые уже слышали это описание,

вынуждены знать, как выглядит Нептун, чтобы иметь сопоставимое впечатление об изображении, что доставляет им неудобство, поскольку само изображение не создает предварительных предположений. Кто-то, кто никогда до этого не видел планету Нептун, может посмотреть на изображение и узнать, как выглядит эта планета, поэтому ваше описание должно делать то же самое.

Совет

Обязательно предоставляйте текстовые описания для всех изображений на странице. Для чисто функциональных изображений, например, логотипов или кнопок, детальные описания не обязательны: описания кнопок «О нас» или «Начальная страница» прекрасно справятся с задачей. Однако непременно предоставляйте очень подробные описания для иллюстраций, диаграмм, графиков и других изображений с содержимым, или для изображений, которые имеют смысл или раскрывают суть текста на странице.

ГЛАВА 35.
Создание карт ссылок

Карта ссылок – это изображение, содержащее одну или несколько гипербластей, называемых **гиперобъектами**. Каждый гиперобъект функционирует, как отдельная ссылка. Ссылки могут указывать на различные страницы или выполнять различные скрипты, что делает карты ссылок более полезными, чем обычные старые добрые изображения со ссылкой, в которых изображение целиком функционирует, как одна ссылка.

Определение

***Карта ссылок** – это изображение, содержащее один или несколько гиперобъектов или гипербластей.*

В карте ссылок сам файл изображения не представляет ничего особенного. Можно начать с любого Web-изображения. Превращает изображение в карту ссылок блок HTML-кода, определяющий форму, положение и функционирование гиперобъектов. Итак, карта ссылок состоит из двух частей:

1. Самого файла изображения;
2. Блока кода, определяющего гиперобъекты.

Объедините их, и вы получите что-то похожее на рисунок 35.1. Обратите внимание, что в теге `img` используется атрибут `usemap`, который сообщает браузеру, какое определение карты применить. На странице можно иметь любое число карт ссылок и даже можно использовать одно и то же определение для абсолютно разных изображений.

Совет

Поскольку гиперобъекты определяются в HTML-коде, а не в самом изображении, если разделить изображение и его код, то карта ссылок превратится обратно в обычное старое доброе изображение без ссылки. Перемещение изображения на другую страницу, без перемещения при этом блока кода, определяющего гиперобъекты, приведет к отображению неактивной карты ссылок.

Листинг 35.1. Просмотр исходного кода для рисунка 35.1

```
<img src="images/mars.jpg" width="500" height="300" usemap="#mars">
<!- Начало определений гиперобъектов ->
<map name="mars">
      <area shape="rect" coords="9,3,164,148" href="mars01.htm">
      <area shape="rect" coords="172,2,328,149" href="mars02.htm">
```

```
    <area shape="rect" coords="334,5,487,149" href="mars03.htm">
    <area shape="rect" coords="9,151,163,294" href="mars04.htm">
    <area shape="rect" coords="171,153,327,296" href=
"mars05.htm">
    <area shape="rect" coords="334,152,488,295" href=
"mars06.htm">
</map>
<!- Конец определений гиперобъектов ->
```

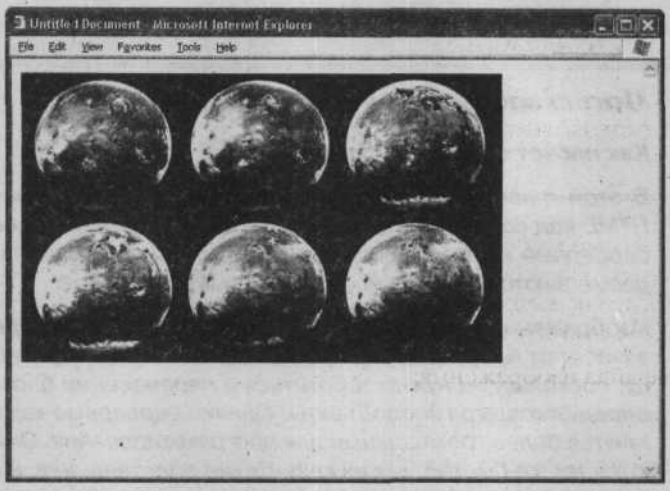


Рис. 35.1. Карты ссылок состоят из файла изображения и блока HTML-кода, определяющего гиперобъекты

Совет

Теги `map` *можно размещать в любом месте HTML-документа, однако наиболее подходящим местом является конец HTML-документа. В таком случае вы всегда будете знать, где искать определения карт ссылок, если понадобится их изменить или переместить на другую страницу.*

Само определение находится между тегами `map`. Значение атрибута `name` тега `map` соответствует значению атрибута `usemap` тега `img`, за одним исключением, которое бросается в глаза: в значении атрибута `usemap` перед именем определения ставится символ числа (#), а значение атрибута `name` не имеет префикса.

Каждый гиперобъект карты ссылок получает свой собственный тег `area`. Таким образом, если у нас есть шесть гиперобъектов, как показано на рисунке 35.1, нам понадобится шесть тегов `area`. Эти теги `area` могут идти в любом порядке. Как можно предположить, атрибуты тега `area` определяют, где и как будут располагаться гиперобъекты:

- Атрибут `shape` определяет, какая форма гиперобъекта из трех возможных используется: прямоугольник (`rect`), окружность (`circle`) или многоугольник (`poly`).
- Атрибут `coords` определяет положение и размер гиперобъекта. В зависимости от значения атрибута `shape`, значения атрибута `coords` имеют разный смысл (обратитесь к таблице 35.1). Левый верхний угол любого изображения имеет координаты (0,0). Правый нижний угол изображения с размерами 500 на 300 пикселов имеет координаты (500,300).
- Атрибут `href` содержит ссылку гиперобъекта. Его значением может быть путь к новой странице, или оно может открывать пустое окно электронного сообщения (`href="mailto:mars@email.eml"`) или запускать скрипт (`href="javascript:doMars();"`).

Часто задаваемые вопросы

Как насчет серверных карт ссылок?

В этой главе описываются клиентские карты ссылок, для которых HTML-код содержит определения гиперобъектов. Также существуют серверные карты ссылок, для которых определения гиперобъектов размещаются в отдельном файле на Web-сервере.

Изображения с определениями, размещенными на сервере, меньше зависят от браузера, и их проще переносить со страницы на страницу, поскольку не нужно заботиться о перенесении блока HTML-кода, определяющего гиперобъекты. Однако серверные карты ссылок являются более громоздкими для программирования. Они не выполняются так же быстро, как их клиентские родственники, и их невозможно сделать доступными.

И хотя раньше серверные карты ссылок были стандартом, они почти исчезли, и теперь все основные браузеры поддерживают клиентские карты ссылок.

Совет

Название карты ссылок не обязательно должно соответствовать названию файла изображения, однако использование одного имени является удобным способом отслеживания, какая карта ссылок соответствует какому изображению.

Совет

Следите, чтобы ваши области гиперобъектов не перекрывались. Когда случается такая ситуация, браузеры сбиваются с толку и карта ссылок, вероятно, будет работать неправильно.

Итак, если вы хотите использовать круглые гиперобъекты вместо прямоугольных для изображения **mars.jpg**, просто измените значение атрибута shape и обновите координаты:

```
<img src="images/mars.jpg" width="500" height="300" usemap="#mars">
<!- Начало определений гиперобъектов ->
<map name="mars">
      <area shape="circle" coords="85,76,60" href="mars01.htm">
      <area shape="circle" coords="245,76,70" href="mars02.htm">
      <area shape="circle" coords="410,76,70" href="mars03.htm">
      <area shape="circle" coords="85,225,70" href="mars04.htm">
      <area shape="circle" coords="245,225,70" href="mars05.htm">
      <area shape="circle" coords="410,225,70" href="mars06.htm">
</map>
<!- Конец определений гиперобъектов ->
```

Совет

В одной карте ссылок можно смешивать и сочетать формы. Гиперобъекты не обязательно должны иметь одинаковую форму.

Табл. 35.1. Смысл значений атрибута coords

Форма	Число координат	Смысл	Пример
circle	Всегда 3	Координата *x* центральной точки, координата *y* центральной точки, радиус окружности в пикселах	coords="10,12,20"
poly	По меньшей мере, 6 и всегда кратно 2	Координата *x* точки фигуры, координата *y* точки фигуры (повторяется столько раз, сколько точек фигуры необходимо для описания многоугольника)	coords="100,150, 200,100,50,150" coords="275,50, 300,150,350,100, 400,150,450,50"
rect	Всегда 4	Координата *x* левого верхнего угла, координата *y* левого верхнего угла, координата *x* правого нижнего угла, координата *y* правого нижнего угла	coords="0,0,100, 150"

ГЛАВА 36.
Повышение доступности карт ссылок

Все знают, что для улучшения доступности изображения пишется подробное текстовое описание. Средства специальных возможностей, например, читатели экрана, особым образом выбирают текст, и посетитель, имеющий проблемы со зрением, слышит вербальное описание того, что изображено на картинке.

Однако карта ссылок является особым случаем, поскольку это не просто изображение. Карта ссылок объединяет изображение с гиперобластями, которые функционируют, как ссылки. Поэтому понятно, что для более доступной карты ссылок требуется как общее текстовое описание, так и текстовые описания для каждой гиперобласти.

Используйте атрибут `alt` тега `img`, чтобы задать текстовое описание для всего изображения, точно так же, как вы сделали бы это для любого другого изображения на сайте:

```
<img src="images/mars.jpg" width="500" height="300" usemap="#mars"
alt="An image map showing the decline of surface water on the plan-
et Mars.">
```

Атрибут `alt` тега `area` прекрасно подойдет для текстового описания гиперобластей:

```
<map name="mars">

      <area  shape="rect"  coords="9,3,164,148"  href="mars01.htm"
alt="Phase 1">

      <area shape="rect" coords="172,2,328,149" href="mars02.htm"
alt="Phase 2">

      <area shape="rect" coords="334,5,487,149" href="mars03.htm"
alt="Phase 3">

      <area shape="rect" coords="9,151,163,294" href="mars04.htm"
alt="Phase 4">

      <area    shape="rect"    coords="171,153,327,296"    href=
"mars05.htm" alt="Phase 5">

      <area    shape="rect"    coords="334,152,488,295"    href=
"mars06.htm" alt="Phase 6">

</map>
```

Наш любимый комитет стандартов, консорциум World Wide Web (W3C), советует включать дублирующие текстовые ссылки для гиперобластей карты ссылок с тем, чтобы текстовые браузеры или другие устройства, которые не отображают изображения, могли без труда получить доступ к содержимому карты ссылок, как показано на рисунке 36.1. В любом случае добавление небольшого количества текстовых инструкций не повредит.

Совет

Всегда разделяйте ссылки в горизонтальном списке каким-нибудь печатным символом, например, круглыми скобками, квадратными скобками или символами окантовки (вертикальной чертой). Это поможет вашим посетителям, бегло просматривающим страницы, отличить ссылки, а устройствам чтения экрана поможет определить, где заканчивается одна ссылка и начинается другая.

Другими словам, не делайте так:

Phase 1 Phase 2 Phase 3 Phase 4

Попробуйте так:

[Phase 1] [Phase 2] [Phase 3] [Phase 4]

Или так:

(Phase 1) (Phase 2) (Phase 3) (Phase 4)

Конечно, квадратные и круглые скобки придают вид блочного, технического терминала для фанатов математики. Для чего-то более эстетического, попробуйте использовать символы окантовки:

Phase 1 | Phase 2 | Phase 3 | Phase 4

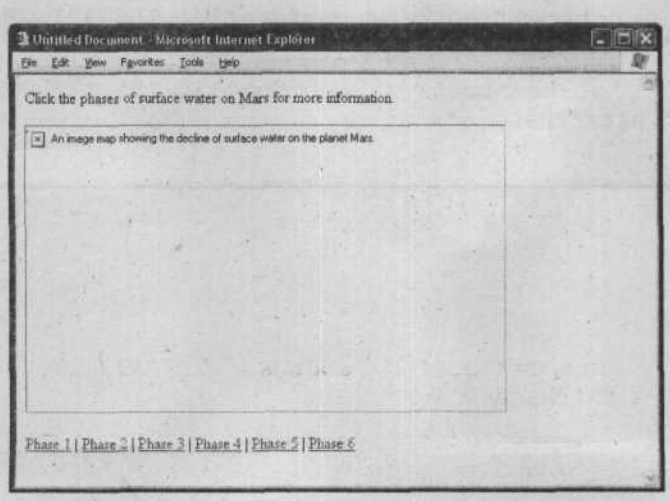


Рис. 36.1. Эта страница с отсутствующим изображением может никогда и не выиграет приза, но ее карта ссылок полностью доступна для использования

Листинг 36.1. Просмотр исходного кода для рисунка 36.1

```
<p>Click the phases of surface water on Mars for more information.</p>

<img src="images/mars.jpg" width="500" height="300" border="0" usemap="#mars" alt="An image map showing the decline of surface water on the planet Mars.">

<p><a href="mars01.htm">Phase 1</a> | <a href="mars02.htm">Phase 2</a> | <a href="mars03.htm">Phase 3</a> | <a href="mars04.htm">Phase 4</a> | <a href="mars05.htm">Phase 5</a> | <a href="mars06.htm">Phase 6</a></p>

<map name="mars">

    <area shape="rect" coords="9,3,164,148" href="mars01.htm" alt="Phase 1">

    <area shape="rect" coords="172,2,328,149" href="mars02.htm" alt="Phase 2">

    <area shape="rect" coords="334,5,487,149" href="mars03.htm" alt="Phase 3">

    <area shape="rect" coords="9,151,163,294" href="mars04.htm" alt="Phase 4">

    <area shape="rect" coords="171,153,327,296" href="mars05.htm" alt="Phase 5">

    <area shape="rect" coords="334,152,488,295" href="mars06.htm" alt="Phase 6">

</map>
```

ГЛАВА 37.
Использование фоновых изображений в ячейках таблицы

Вероятно вы знаете, что внутри ячейки таблицы можно разместить изображение следующим образом:

```
<td><img src="mercury.jpg" width="200" height="200"></td>
```

Возможно, вы также знаете, что можно задать цвет фона ячейки таблицы следующим образом:

```
<td bgcolor="#FF0000">Behold, the planet Mercury</td>
```

Но знали ли вы и то, что можно использовать изображение в качестве фона ячейки таблицы? Это работает следующим образом:

```
<td background="mercury.jpg" width="200" height="200">Behold, the
planet Mercury</td>
```

Использование фоновых изображений в ячейках таблицы позволяет наложить HTML-содержимое, как показано на рисунке 37.1.

Совет

Атрибуты `width` и `height` тега `td` описывают размер ячейки таблицы, а не размер изображения. Старые браузеры могут уменьшать размеры ячейки, чтобы они соответствовали размерам содержимого ячейки, даже если вы укажете точные значения атрибутов `width` и `height`, что может привести к обрезанию фонового изображения.

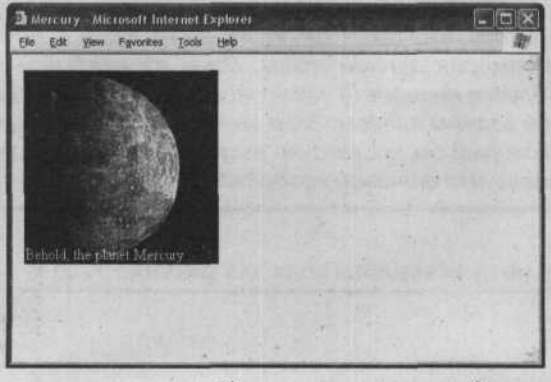


Рис. 37.1. При использовании изображения в качестве фона ячейки таблицы можно накладывать HTML-содержимое

Листинг 37.1. Просмотр исходного кода для рисунка 37.1

```
<table>
   <tr>
<!- Установите значение атрибута valign в bottom, чтобы прижать
текст к низу ячейки таблицы. ->
      <td valign="bottom" width="200" height="200" background=
   "images/mercury.jpg">
         Behold, the planet Mercury
      </td>
   </tr>
</table>
```

Определение

Мозаичное заполнение *– это повторение изображения для заполнения заданной области.*

Этот метод работает отлично в том случае, когда маловероятно, что объем содержимого, которое вы хотите наложить, приведет к расширению ячейки таблицы за значения атрибутов `width` и `height` изображения. В противном случае браузер использует мозаичное заполнение или повторение изображения, чтобы заполнить дополнительную область, как показано на рисунке 37.2, в результате чего может получиться не тот эффект, которого мы хотели достичь.

Совет

Убедитесь, что текст удобочитаем на фоне изображения, в отличие от примеров данной главы. Одна из наиболее распространенных проблем в Интернете заключается в плохом контрасте между передним и задним планом. Если вы не можете найти цвет или стиль текста, который бы достаточно выделялся на фоновом изображении, не используйте фоновое изображение.

Листинг 37.2. Просмотр исходного кода для рисунка 37.2

```
<table>
   <tr>
<!- Здесь нет необходимости использовать атрибут valign, поскольку
текст заполнит ячейку с размерами 200 на 200 пикселов. ->
      <td width="200" height="200" background="images/mercury.jpg">
```

```
    <p>Behold, the planet Mercury. It is one of the hottest
places in the solar system due to its proximity to the sun.</p>

    <p>The planet Mercury is a small, arid, lifeless, sun-baked
excuse for a heavenly body. Hardly anyone bothers to study
it.</p>

    <p>Its only notable quality is the speed with which it orbits
the sun, from which it derives its name. In Roman myth, Mercury,
with wings on his heels, was the messenger of the gods, a divine
postman and errand runner, the swiftest of deities.</p>

    </td>

  </tr>
</table>
```

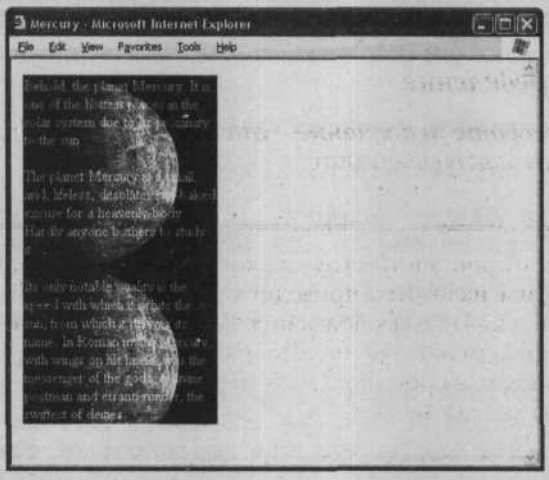


Рис. 37.2. Если размеры ячейки превышают размеры фонового изображения, браузер использует мозаичное заполнение изображения, чтобы заполнить дополнительную область

К сожалению, язык HTML не предоставляет дополнительные атрибуты для точной настройки поведения фоновых изображений, что позволяют таблицы CSS (обратитесь к главе 38).

ГЛАВА 38.
Использование фоновых изображений в CSS-разделах

Фоновые изображения в ячейках таблицы либо есть, либо их нет, с чем вы уже познакомились в главе 37. Вы либо определяете их в HTML-коде, либо не определяете совсем, а браузер позаботится обо всем остальном. Если объем содержимого ячейки таблицы превышает физические размеры изображения, браузер использует мозаичное заполнение для изображения. Если вам подходит такой эффект, то вам повезло. Если вам не подходит такой эффект – дело плохо.

Иначе обстоит дело с фоновыми изображениями в CSS-разделах. Таблицы CSS предоставляют ряд возможностей для дизайна. Если хотите, можно использовать множество способов мозаичного заполнения для изображения, или же можно отобразить изображение один и только один раз. Можно также указать, где должно располагаться изображение в разделе.

Начните с типичного раздела, наподобие следующего:

```
<div id="mercury" style="width: 200px;">

    <p>Behold, the planet Mercury. It is one of the hottest places
    in the solar system due to its proximity to the sun.</p>

    <p>The planet Mercury is a small, arid, lifeless, sun-baked
    excuse for a heavenly body. Hardly anyone bothers to study
    it.</p>

    <p>Its only notable quality is the speed with which it orbits
    the sun, from which it derives its name. In Roman myth, Mercury,
    with wings on his heels, was the messenger of the gods, a divine
    postman and errand runner, the swiftest of deities.</p>

</div>
```

Для добавления фонового рисунка к разделу используйте атрибут `background-image` следующим образом:

```
<div id="mercury" style="width: 200px; background-image: url
(images/mercury.jpg);">
```

Обратите внимание, что путь к изображению заключается в конструкцию `url(...)`.

Добавление одного CSS-атрибута `background-image` приводит к созданию раздела, который ведет себя почти так же, как ячейка таблицы, в которой используется мозаичное заполнение для фонового рисунка, как показано на рисунке 38.1. Остальные определения CSS-стилей позволяют настроить поведение фонового изображения.

Если вы не хотите, чтобы использовалось мозаичное заполнение для фонового изображения, воспользуйтесь атрибутом `background-repeat`, как показано на рисунке 38.2.

Листинг 38.1. Просмотр исходного кода для рисунка 38.2

```
<div id="mercury" style="width: 200px;
   background-image: url(images/mercury.jpg);
   background-repeat: no-repeat;">
```

Чтобы расположить фоновое изображение внутри раздела, укажите значение для атрибута `background-position`, как показано на рисунке 38.3.

Совет

Если вы хотите использовать мозаичное заполнение для фонового изображения только в горизонтальном направлении, установите значение атрибута `background-repeat` *в* `repeat-x`. *Используйте значение* `repeat-y`, *чтобы использовать мозаичное заполнение для фонового изображения только в вертикальном направлении.*

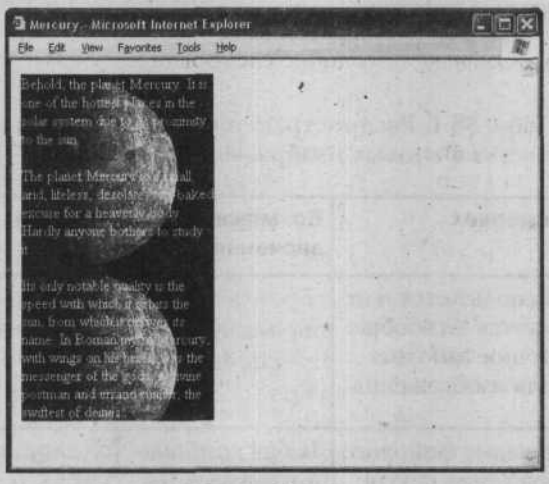


Рис. 38.1. Если указать только CSS-атрибут `background-image` *и ничего больше, для фонового изображения будет использовано мозаичное заполнение*

Листинг 38.2. Просмотр исходного кода для рисунка 38.3

```
<div id="mercury" style="width: 200px;
   background-image: url(images/mercury.jpg);
   background-repeat: no-repeat;
   background-position: center;">
```

В таблице 38.1 приведены распространенные значения для CSS-атрибутов, касающихся фоновых изображений. Помните, что необходимо основательно протестировать страницы перед тем, как выложить их в Интернете, поскольку браузеры вытворяют странные вещи, когда они встречают определения стилей.

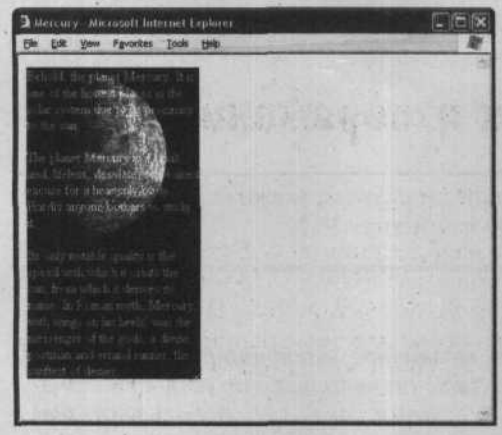


Рис. 38.2. Чтобы предотвратить использование мозаичного заполнения для фонового изображения, установите значение атрибута `background-repeat` *в* `no-repeat`

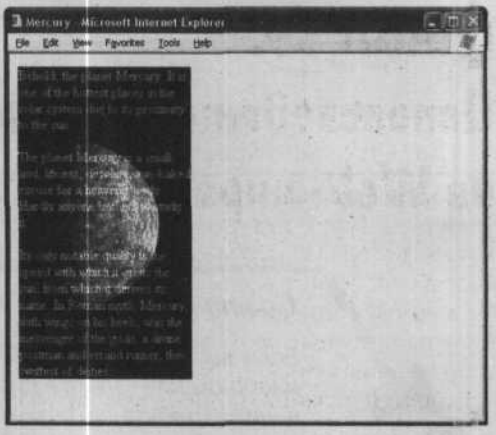


Рис. 38.3. Расположите фоновое изображение при помощи атрибута `background-position`

Табл. 38.1. Распространенные CSS-атрибуты для фоновых изображений в разделах

Стиль	Определяет	Возможные значения	Пример
`background-repeat`	Как используется и используется ли вообще мозаичное заполнение для изображения	`repeat-x`, `repeat-y`, `no-repeat`, `repeat`	`background-repeat: repeat-y;` `background-repeat: repeat;`
`background-position`	Положение фонового изображения внутри раздела	Любая комбинация следующих значений: `left`, `right`, `center`, `top`, `bottom`	`background-position: left;` `background-position: right bottom;` `background-position: center left;` `background-position: center;`

ГЛАВА 39.
Использование фоновых изображений на Web-страницах

Существует два способа добавления фонового изображения на Web-страницу. Первый способ заключается в использовании атрибутов языка HTML. Второй заключается в использовании определений стилей CSS. При использовании языка HTML мы получаем ограниченные возможности дизайна, но высокую надежность при использовании на различных браузерах. При использовании таблиц CSS мы получаем больше возможностей дизайна, но меньшую надежность.

В языке HTML мы добавляем атрибут `background` в тег `body`, как в следующей строке кода:

```
<body background="images/ufo.gif">
```

Совет

Остерегайтесь фоновых изображений, которые подавляют содержимое страницы. Придерживайтесь бесцветных изображений, не влияющих на четкость и разборчивость вашего текста.

Браузер использует мозаичное заполнение или повторение изображения по всему доступному пространству окна браузера, как показано на рисунке 39.1.

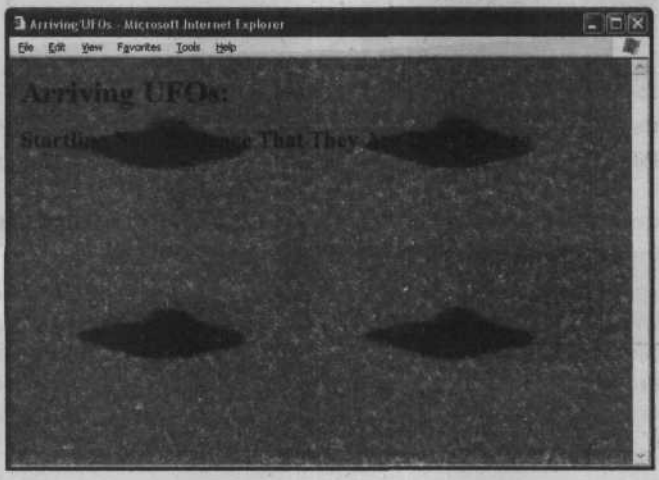


Рис. 39.1. При добавлении фонового изображения на страницу при помощи языка HTML браузер автоматически использует мозаичное заполнение, чтобы заполнить страницу

При использовании каскадных таблиц стилей получается такой же результат:

```
<body style="background-image: url(images/ufo.gif);">
```

В предыдущем примере создается фоновое изображение, которое использует мозаичное заполнение, точно такое же, как показанное на рисунке 39.1. Однако таблицы CSS дают дополнительные возможности для точного управления поведением фонового изображения, включая стили `background-repeat`, `background-position` и `background-attachment`.

Используйте стиль `background-repeat` для управления тем, как будет (и будет ли вообще) использоваться мозаичное заполнение для фонового изображения, как показано на рисунке 39.2.

Листинг 39.1. Просмотр исходного кода для рисунка 39.2

```
<body style="background-image: url(images/ufo.gif);
             background-repeat: repeat-y;">
```

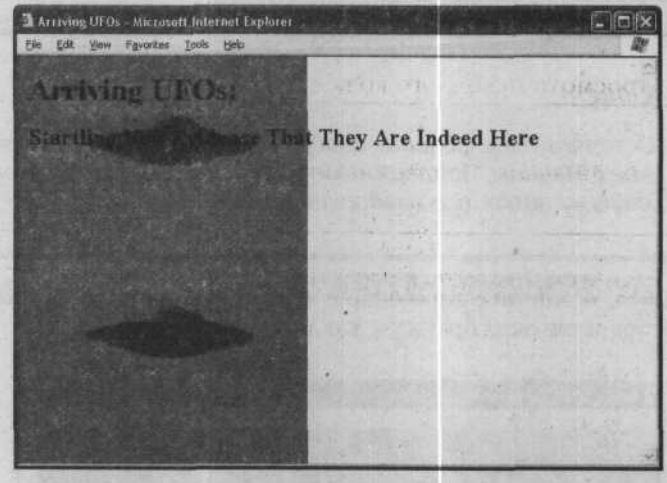


Рис. 39.2. При использовании таблиц CSS для фоновых изображений можно указывать только вертикальное мозаичное заполнение

Место расположения фонового изображения на странице определите при помощи атрибута `background-position`, как показано на рисунке 39.3.

Атрибут `background-attachment` определяет, как фоновое изображение реагирует на прокрутку. Установите атрибут `background-attachment` в значение `fixed`, чтобы фоновое изображение не реагировало на прокрутку; т.е. фоновое изображение остается привязанным к своему положению в окне браузера. Содержимое страницы может прокручиваться, однако фоновое изображение будет находиться на прежнем месте, как показано на рисунке 39.4.

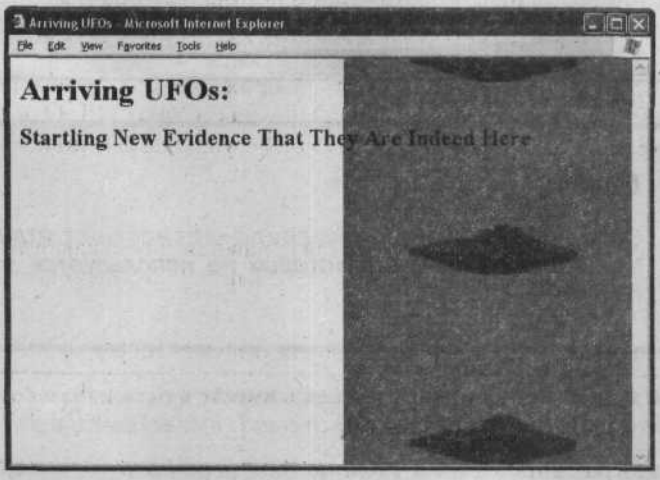


Рис. 39.3. Перемещайте фоновое изображение по всему экрану, включив в определение стиля атрибут `background-position`

Листинг 39.2. Просмотр исходного кода для рисунка 39.3

```
<body style="background-image: url(images/ufo.gif);
             background-repeat: repeat-y;
             background-position: right;">
```

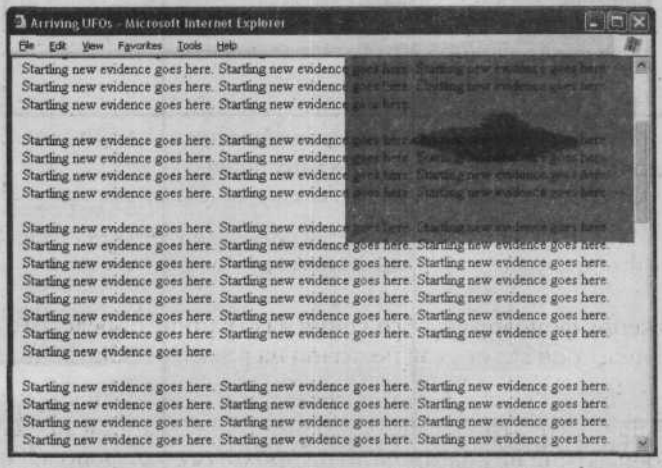


Рис. 39.4. При использовании таблиц CSS можно даже сделать так, чтобы фоновое изображение не реагировало на прокрутку

Листинг 39.3. Просмотр исходного кода для рисунка 39.4

```
<body style="background-image: url(images/ufo.gif);
             background-repeat: no-repeat;
```

```
background-position: right;
background-attachment: fixed;">
```

Совет

Определение стиля `background-attachment` *отлично работает с изображениями, для которых не используется мозаичное заполнение.*

Чтобы фоновое изображение прокручивалось вместе с остальным содержимым страницы, установите значение атрибута `background-attachment` в `scroll`.

Для удобства программирования в таблице 39.1 перечислены возможности таблиц CSS, касающиеся фоновых изображений.

Табл. 39.1. Распространенные возможности таблиц CSS для фоновых изображений

Стиль	Определяет	Возможные значения	Пример
`background-repeat`	Как используется и используется ли вообще мозаичное заполнение для изображения	`repeat-x`, `repeat-y`, `no-repeat`, `repeat`	`background-repeat: repeat-y;` `background-repeat: repeat;`
`background-position`	Положение фонового изображения внутри раздела	Любая комбинация следующих значений: `left`, `right`, `center`, `top`, `bottom`	`background-position: left;` `background-position: right bottom;` `background-position: center left;` `background-position: center;`
`background-attachment`	Как фоновое изображение реагирует на прокрутку	`fixed`, `scroll`	`background-attachment: fixed;` `background-attachment: scroll;`

ЧАСТЬ 4.
Текст

- Правильное использование текстовых элементов
- Правильное использование элементов фраз
- Добавление каскадных таблиц стилей
- Замена тегов форматирования каскадными таблицами стилей
- Создание специальных стилей форматирования
- Изменение вида текстового элемента
- Определение стилей классов
- Управление типом шрифта
- Управление размером шрифта
- БЛОГ: Подведение итогов по заданию размеров
- Управление интервалами
- Управление стилями с выключкой и без выключки
- Управление отступами
- Создание списков
- Выделение текста цветом
- Восстановление горизонтальных линеек
- Добавление строки даты и метки времени
- Разработка таблиц данных
- Использование псевдоэлементов
- Определение стилей описания элементов

ГЛАВА 40.
Правильное использование текстовых элементов

Практически каждому Web-дизайнеру приходится время от времени делать это: с помощью HTML-тегов размечать текст, основываясь на внешнем виде текста в браузере, а не на функциональном назначении используемого тега. Необходимо быстро получить строку полужирного текста? Кто среди нас не поддавался искушению воспользоваться тегом заголовка, даже если рассматриваемый текст не является заголовком? Требуется создать отступ? Этот фокус можно проделать с помощью тега `<blockquote>`, даже когда вам не надо форматировать цитату с отступами.

Теперь, когда каскадные таблицы стилей (CSS – Cascading Style Sheets) поддерживаются на приемлемом уровне большинством браузеров, прежние аргументы о допустимости всего, что работает, уже больше не выдерживают критики. Каскадные таблицы стилей позволяют придать любому HTML-элементу такой вид, какой хочет получить дизайнер, поэтому нет необходимости заимствовать тег на основании его внешнего вида в браузере – это означает, что HTML-теги могут вернуться обратно к своему исходному назначению: определять структуру документа. Другими словами, когда вы размечаете что-нибудь тегом заголовка, это действительно должно функционировать как заголовок на Web-странице. Когда вы размечаете что-нибудь тегом `<blockquote>`, это должен быть текстовый блок с отступами, а не изображение, для которого требуется вставить немного дополнительного пустого пространства.

Если вы достаточно долго разрабатываете Web-сайты, возможно у вас выработались некоторые вредные привычки, и вы, вероятно, уже забыли, для разметки чего предназначались первоначально текстовые теги HTML. Ознакомьтесь с таблицей 40.1.

Табл. 40.1. Распространенные текстовые теги HTML

Тег	Определяет	Пример
`address`	Почтовый адрес, номер телефона, адрес электронной почты или другая форма контактной информации	`<address>Здесь располагается ваш адрес</address>`
`blockquote`	Цитата с отступами	`<blockquote>Здесь располагается цитата с отступами</blockquote>`
`h1`	Заголовок первого уровня	`<h1>Здесь располагается заголовок</h1>`
`h2`	Заголовок второго уровня	`<h2>Здесь располагается заголовок</h2>`
`h3`	Заголовок третьего уровня	`<h3>Здесь располагается заголовок</h3>`

Тег	Определяет	Пример
h4	Заголовок четвертого уровня	Здесь располагается заголовок
h5	Заголовок пятого уровня	Здесь располагается заголовок
h6	Заголовок шестого уровня	Здесь располагается заголовок
p	Абзац	Здесь располагается абзац
q	Внутристрочная цитата	<q>Здесь располагается внутристрочная цитата</q>

Работа с тегами заголовков

Как показано на рисунке 40.1, язык HTML предоставляет шесть уровней заголовков — с h1 по h6 — в порядке убывания структурной и визуальной важности. Заголовки первого уровня отображаются полужирным и крупным шрифтом и являются основными заголовками на первой странице газеты. Заголовки второго уровня не такие крупные и похожи на дополнительные заголовки или заголовки разделов в газете. Так заголовки изменяются до заголовков шестого уровня, которые похожи на полужирный текст рубричной рекламы: большей частью по внешнему виду это не заголовок, но, тем не менее, выполняет функцию заголовка.

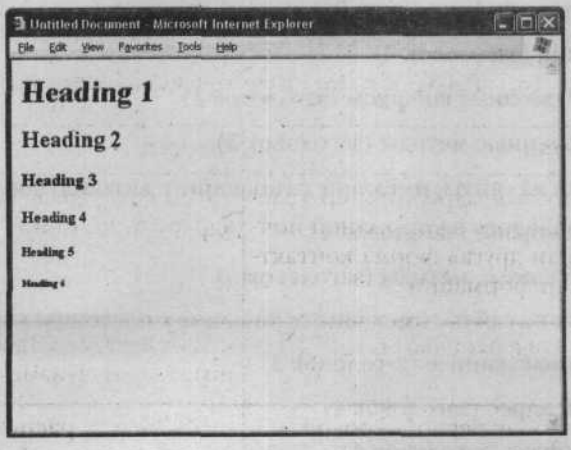


Рис. 40.1. В языке HTML определяются шесть уровней заголовков

В размеченном надлежащим образом HTML-документе должны содержаться теги заголовка для обозначения логической структуры страницы. Это означает, что основной заголовок Web-страницы должен использовать тег <h1>. Подраздел, расположенный под данным заголовком, оформляется тегом <h2>; под-подраздел заключается в тег <h3>, и т. д. В идеале, если удалить повествовательный текст со страницы,

надлежащим образом структурированный набор тегов заголовков дает полную и точную схему информации Web-страницы, подобно следующему примеру:

Совет

Если содержимое Web-страницы не разбивается в соответствии с некоторой логической иерархией, тогда все заголовки на странице будут одинаковыми.

Добро пожаловать на мой домашний сайт (заголовок 1)

Что находится на моем сайте (заголовок 2)

О странице с играми (заголовок 3)

О странице с новостями (заголовок 3)

О моем интерактивном магазине подарков (заголовок 3)

Информация обо мне (заголовок 2)

Мое резюме (заголовок 3)

Текущее место работы (заголовок 4)

Предыдущее место работы (заголовок 4)

Образование (заголовок 4)

Дипломная работа (заголовок 5)

Университет (заголовок 5)

Мои увлечения (заголовок 3)

Искусство вязания шнурком (заголовок 4)

Излюбленные методы (заголовок 5)

Ссылки на сайты, имеющие отношение к вязанию шнурком (заголовок 5)

Плетение корзин (заголовок 4)

Излюбленные методы (заголовок 5)

Ссылки на сайты, имеющие отношение к плетению корзин (заголовок 5)

Мои контактные данные (заголовок 3)

Почтовый адрес (заголовок 4)

Телефон/факс (заголовок 4)

Адрес электронной почты (заголовок 4)

Помните, что не важно какой эффект оказывают эти теги на вид текста, поскольку можно использовать каскадные таблицы стилей для придания тексту любых требуемых атрибутов. Важно обеспечить согласованное использование тегов и их соответствие соглашениям, принятым в HTML.

Маркировка абзацев

Тег абзаца является, возможно, самым пренебрегаемым тегом во всем HTML. Кто хочет замедлять производственный процесс и включать эти раздражающие символы и в начале и в конце каждого одиночного блока основного текста? Тем не менее, преимущества дополнительных усилий по вставке этих тегов могут быть исключительными. Каскадные таблицы стилей обеспечивают широкий диапазон возможностей по оформлению основного текста, из которых даже самые незначительные существенно улучшают вид текста по сравнению со сжатыми, непривлекательными блоками плохо сверстанных строк, которые выдает HTML (Рис. 40.2). Ключевое условие надлежащего применения каскадных таблиц стилей – верная разметка документа, а это означает разметку тегами абзацев всех элементов, которые, как предполагается, должны быть абзацами текста.

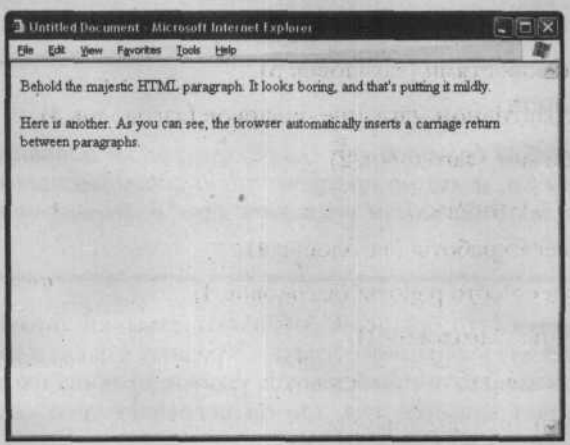


Рис. 40.2. Абзацы HTML столь непривлекательны, что трудно упрекнуть вас в ненадлежащем маркировании текста. Но старайтесь всегда маркировать абзацы, чтобы использовать каскадные таблицы стилей для улучшения внешнего вида текста

Использование элементов цитат

В HTML существует два вида тегов для разметки цитат: <quote> и <blockquote>. Тег <quote> предназначен для внутристрочных цитат, подобных диалогу в повести, тогда как тегом <blockquote> маркируются выделенные цитаты, как на рисунке 40.3.

Консорциум World Wide Web (W3C) рекомендует указывать язык цитаты в атрибуте lang тега <quote>, как в следующем примере:

```
<p>Her lip trembled. <q lang="en-us">Plastic,</q> she replied, wondering even as she said it if she should have asked for paper.</p>
```

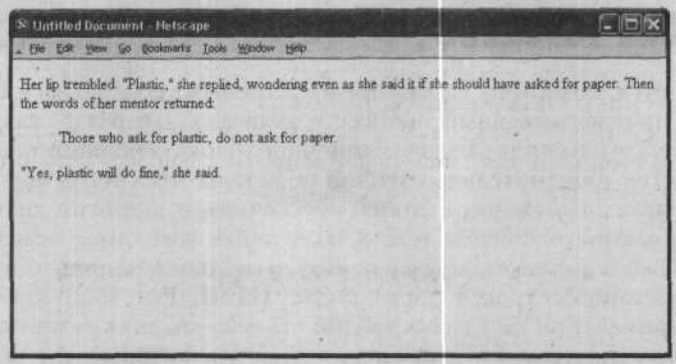


Рис. 40.3. Тег <quote> предназначен для внутристрочных цитат, тогда как тег <blockquote> используется для цитат с отступами

Совет

В любом браузере тег <blockquote> не добавляет автоматически кавычки, и это не предусмотрено спецификациями HTML, поэтому нет необходимости указывать язык в открывающем теге <blockquote>.

Зачем? Предполагается, что браузеры добавляют кавычки автоматически в зависимости от соглашений для выбранного языка. (В разных языках в качестве кавычек используются разные символы и применяются разные правила их расстановки.) Браузер Netscape добавляет кавычки там, где он встречает теги <quote>, но браузер Internet Explorer этого не делает.

Отображение адреса

Заключите контактную информацию о себе и своей организации в теги <address>, подобно следующему примеру:

```
<address>
   The Old Software Company<br>
   100 Main St. Ste. 500<br>
   Youngstown, OH 92123
</address>
```

Обратите внимание на теги
 в конце каждой строки, предназначенные для принудительного возврата каретки. Получаемый результат выглядит, как показано на рисунке 40.4.

Для форматирования любого прежнего адреса на Web-странице не следует использовать теги <address>. Попробуйте зарезервировать эти теги специально для контактной информации, относящейся к вашему местонахождению, поскольку механизмы поиска информации и им подобные приложения могут искать теги <address> для извлечения сведений о владельце сайта. Если необходимо отформатировать несколько адресов, аналогичных контактной информации для вашего сайта, или нечто подобное, следует избегать тега <address> и определять специальный стиль в каскадных таблицах стилей.

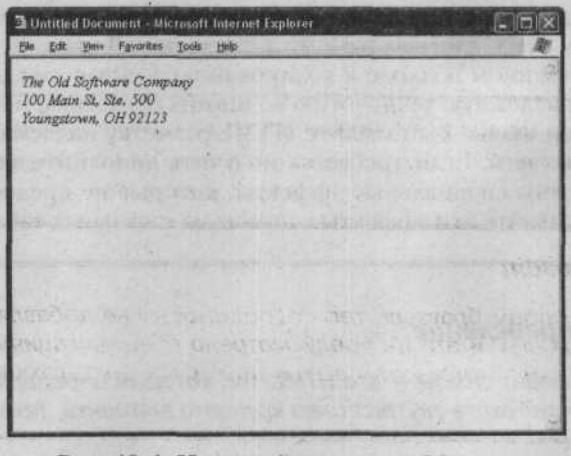


Рис. 40.4. Используйте теги <address> для маркировки контактной информации, относящейся к вам или к вашей организации

ГЛАВА 41.

Правильное использование элементов фраз

Элементы фразы – это серия HTML-тегов для идентификации определенной структуры сегмента текста внутри более крупных элементов, таких, как абзац.

Элементы фразы обычно создают специальные текстовые эффекты, например, полужирное и курсивное начертание. Именно таким образом дизайнеры склонны использовать их для своих нужд, а не по прямому назначению. В главе 40 вам встретился совет о более ответственном подходе к кодированию. Сейчас каскадные таблицы стилей предоставляют надежную технологию во многих браузерах, и этот аргумент справедлив и для данной главы. Выполняйте HTML-разметку надлежащим образом, расставляя правильные теги. Если требуется получить дополнительные текстовые эффекты или если нужны специальные эффекты, которые не предоставляются тегами по умолчанию, создавайте эти эффекты с помощью каскадных таблиц стилей.

Определение

Элемент фразы – это HTML-тег, который определяет структуру сегмента текста внутри более крупного элемента, такого, как абзац.

Не забывая о вышесказанном, ознакомьтесь с полным списком элементов фраз, приводимом в таблице 41.1.

Назначение большинства элементов фраз ясно из их названия, но некоторые теги в виде аббревиатур и акронимов нуждаются в дополнительных пояснениях.

Сначала кое-что напомним. **Аббревиатура** – это группа букв, которая заменяет слово или несколько слов. Каждая буква в некоторых аббревиатурах произносится отдельно, даже если аббревиатура в целом удобна для произношения. Хорошим примером аббревиатуры является **ООН**, которая расшифровывается как **Организация Объединенных Наций**. Эта аббревиатура произносится не «он», а «о, о, н».

Табл. 41.1. Элементы фразы в HTML

Тег	Определение	Стандартный вид	Пример
`em`	Текст, на котором акцентируется внимание	Курсив	`<em>`Что`</em>` вы сказали?
`strong`	Действительно выделенный текст	Полужирный	Вы не можете быть `<strong>`серьезным!`</strong>`
`cite`	Цитата; ссылка или источник	Курсив	Что случилось, док? `<cite>`– Багз Банни`</cite>`

Тег	Определение	Стандартный вид	Пример
dfn	Термин, который определяется в следующем тексте	Курсив	<dfn>Тег</dfn> является в HTML элементом разметки
code	Компьютерный код	Моноширинный текст (с одинаковой шириной букв)	Тег <code>em</code> идентифицирует выделенный текст
samp	Выходные данные из компьютерной программы, функции и т. д.	Моноширинный текст	Когда это происходит, компьютер выводит сообщение <samp>Ошибка: Перезагрузитесь.</samp>
kbd	Клавиши, нажимаемые пользователем или вводимый им текст	Моноширинный текст	Для продолжения нажмите клавишу <kbd>Ввод</kbd>
var	Переменная в компьютерной программе, уравнении и т. д.	Курсив	Прежде чем продолжить, проверьте значение <var>x</var>.
abbr	Аббревиатура	Обычный текст	Ваш сайт выглядит фантастически, но консорциум <abbr title="World Wide Web Consortium">W3C</abbr> не одобрит, как обычно
acronym	Акроним	Обычный текст	Пошлите свой протест в <acronym title="National Aeronautics and Space Administration">НАСА</acronym>.

В браузере Netscape аббревиатура отображается с пунктирным подчеркиванием. Когда указатель мыши находится над термином, появляется совет, предоставляющий полный эквивалент из атрибута заголовка. Браузер Internet Explorer не содержит подобной функции.

Некоторые аббревиатуры не произносятся вообще, вместо этого они заменяются реальными словами, которые представляют. Рассмотрите аббревиатуру для штата Калифорния – CA. Обычно, когда встречается эта аббревиатура, произносят «Калифорния» вместо «си, эй».

Акроним, наоборот, является аббревиатурой, которая произносится, как целое слово. Когда встречается акроним NASA (National Aeronautics and Space Administration – Национальное агентство по аэронавтике и исследованию космического пространства), он не произносится по буквам «эн, эй, эс, эй», а читается как одно слово «наса»,

и крайне редко подставляется реальное название агентства. Вы просто произносите акроним и продолжаете действовать дальше по своему плану.

Определение

Аббревиатура — это группа букв, которая заменяет слово или серию слов. Акроним — это аббревиатура, которая произносится так, как если бы она была отдельным словом.

Часто задаваемые вопросы

Как можно добавить новое слово в мой родной язык?

Предложите действительно удачный акроним. Некоторые акронимы функционируют настолько хорошо как слова, что они превратились в реальные слова, особенно акронимы, описывающие технические понятия, которые большинство людей и не пытается расшифровать. Лазер (laser), радар (radar) и скуба (scuba) были когда-то акронимами.

В чем проявляется различие между маркировкой акронимов и аббревиатур для посетителей сайтов? Абсолютно ни в чем. Как видно из таблицы 41.1, теги `abbr` и `acronym` не добавляют специального шрифтового форматирования к размеченному тексту (по крайней мере, в Internet Explorer, наиболее популярном браузере на сегодняшний день). Большинство посетителей сайта никогда не узнает, что вам пришлось набирать National Aeronautics and Space Administration для всех встречавшихся акронимов NASA. Но посетители, использующие устройства чтения с экрана и другие специальные средства, будут благодарны вам за проявленную заботу.

Устройства чтения с экрана и подобные им устройства не понимают, когда считывается аббревиатура, а когда акроним, поэтому это сообщается им с помощью разметки. Когда устройству чтения с экрана встречается фраза, размеченная как акроним, оно пытается воспроизвести текст как одно слово. Когда встречается аббревиатура, устройство чтения с экрана воспроизводит его побуквенно или подставляет полный текст, который присутствует в атрибуте `title` этого тега. Некоторые поисковые механизмы каталогизируют страницы в соответствии со словами в атрибуте `title` и по аббревиатуре или акрониму, так что сайт будет найден как теми, кто ведет поиск по ключевой фразе «HTML», так и теми, кто ищет строку «HyperText Markup Language».

ГЛАВА 42.
Добавление каскадных таблиц стилей

Каскадные таблицы стилей (CSS – Cascading Style Sheets) описывают стиль или внешний вид элементов на Web-странице. Каскадные таблицы стилей могут использоваться для переопределения способа отображения браузером HTML-тега или серии тегов, либо с их помощью можно создать полностью новые стили и применить их к любым частям содержимого страниц. С этой точки зрения каскадные таблицы стилей чрезвычайно удобны для разделения **структуры** документа – чисто технического описания Web-страницы, которое определяет отдельные части содержимого как конкретные элементы или типы объектов: абзацы, изображения, заголовки, цитаты и так далее – от представления документа – внешнего вида всех элементов в браузере.

Определение

Каскадные таблицы стилей описывают стиль или внешний вид элементов на Web-странице. Структура страницы — это ее техническое описание, определяющее отношения между частями содержимого и типами так называемых элементов. Представление страницы определяет внешний вид страницы в браузере. Теоретически, HTML обрабатывает структуру страницы, а каскадные таблицы стилей определяют представление, хотя HTML содержит ряд тегов представления, которые ничего не делают со структурой. Это приводит к недовольству Web-разработчиков, ориентированных на полное кодирование тегами.

Когда HTML появился впервые, предполагалось, что это будет структурный язык, причем внешний вид элементов должен был определяться программным обеспечением браузера. Роль Web-разработчика сводилась к определению того, к каким типам объектов относились те или иные части содержимого – т.е. к описанию структуры страницы – а браузер отвечал за представление данных. Для разбирающихся в графике Web-разработчиков этот способ работы был неприемлем. Они требовали все больше и больше управления видом Web-страниц, пока сам код HTML не становился непонятной смесью из структурных элементов и элементов представления, которые не имели вообще ничего общего со структурой. Для Web-разработчиков, погруженных в кодирование, как и для рафинированных специалистов в консорциуме W3C, этот способ работы был одинаково неприемлемым.

Вводите каскадные таблицы стилей, которые, по крайней мере, теоретически удовлетворяют требованиям обоих типов специалистов, занятых Web-разработками. Головы, озабоченные графикой, получают расширенный и улучшенный набор вариантов разработки для управления видом Web-страниц на уровне пикселов, а головы, занятые исключительно кодированием, получают способ использовать HTML как чисто структурный язык. Обратная сторона такого подхода заключается в том, что хотя каскадные таблицы стилей существуют уже несколько лет, они по-прежнему остаются не до конца развитой технологией. «Графические» головы медленно осваивают ее, поскольку каскадные таблицы стилей в большой степени используют кодирование, они интуитивно менее понятны, и их труднее использовать, чем HTML-разметку. Каскадные таблицы стилей предполагают определенный уровень технического понимания,

которого многие люди, ориентированные на визуальное восприятие, не имеют и не желали приобретать в бурные дни возникновения интернет-компаний. Более того, создатели браузеров, такие, как корпорация Microsoft и компания Netscape, не утруждались обеспечением согласованной и полной поддержки каскадных таблиц стилей.

Определение

Таблица стилей — это блок CSS-кода, который помещается между тегами стилей в HTML-документе или в отдельном CSS-файле. Она состоит из правил стилей, или инструкций представления, предназначенных для браузера. Правило стиля имеет селектор стиля, который представляет собой форматируемые элементы, и определение стиля, которое сообщает браузеру, как надо отображать селектор.

Совет

Когда таблица стилей содержится в своем собственном файле, не требуется расставлять теги стилей до и после CSS-кода. Используйте теги стилей только в тех случаях, когда таблица стилей встраивается в HTML-документ.

Время этих веяний медленно уходит в прошлое. Теперь каскадные таблицы стилей получили достаточно широкий уровень поддержки в основных браузерах, поэтому их эффективное применение стало, по крайней мере, реальным. Однако нигде каскадные таблицы стилей не демонстрируют такой стабильности, как подвергаемые критике HTML-теги представления. Поэтому тот, кто пытается использовать их, должен тестировать, тестировать и еще раз тестировать свои Web-страницы в различных браузерах. Это необходимо, чтобы устранить неизбежные проблемы, ошибки и несоответствия, закравшиеся в проект.

Написание каскадных таблиц стилей

Существует два способа добавления каскадных таблиц стилей на Web-страницу: написание таблицы стилей или помещение CSS-кода в атрибут стиля тега, который требуется отформатировать.

Написание таблицы стилей

Таблица стилей — это блок CSS-кода. Она располагается в HTML-документе между тегами стилей или существует в качестве отдельного файла, который импортируется в HTML-документ. Таблица стилей содержит одно или несколько **правил стилей,** или инструкций презентации, предназначенных для браузера, и **определения стиля**, предоставляющего пары атрибут/значение, которые сообщают браузеру, как надо отображать селектор.

Таблица стилей выглядит во многом как приводимый ниже пример кода:

```
<style type="text/css">

    h1 {
        font-family: Arial;
        font-weight: bold;
        font-size: 24px;
    }

    p {
        font-family: "Times New Roman";
        font-size: 16px;
    }

</style>
```

Эта таблица стилей содержит два правила стилей: одно для селектора `h1`, другое для селектора `p`. Помните, что селекторы – это элементы, к которым применяется стиль. В данном случае стиль применяется к тегам `h1` и `p` для заголовков первого уровня и абзацев, соответственно. Код между фигурными скобками (`{}`), следующими после каждого селектора, является определением для этого селектора.

Определение стиля содержит пары атрибут/значение. В HTML пары атрибут/значение принимают следующий вид:

```
имяатрибута="значение"
```

где имя атрибута представляется одним словом, атрибут отделяется от значения с помощью знака равно (=) и значение заключается в кавычки. В каскадных таблицах стилей соглашение по кодированию отличается от принятого в HTML:

```
имя-атрибута: значение;
```

где части имени атрибута разделяются символом тире (-) вместо их совмещения в одном слове; для отделения атрибута от значения используется двоеточие (:), а не знак равенства; и в конце каждой строки ставится знак точки с запятой (;). Обычно значение не требуется заключать в кавычки, хотя из этого правила существует несколько исключений, например, когда имя шрифта содержит более одного слова, как `Times New Roman` в предыдущем примере.

Пусть этот стиль кодирования не пугает вас. Просто, знакомясь с определением, вы можете догадаться, что заголовки первого уровня отображаются полужирным шрифтом **Arial** с размером 24 пиксела, тогда как для отображения текста абзацев используется шрифт **Times New Roman**, 16 пикселов. Результат применения этих стилей похож на изображение в браузере на рисунке 42.1.

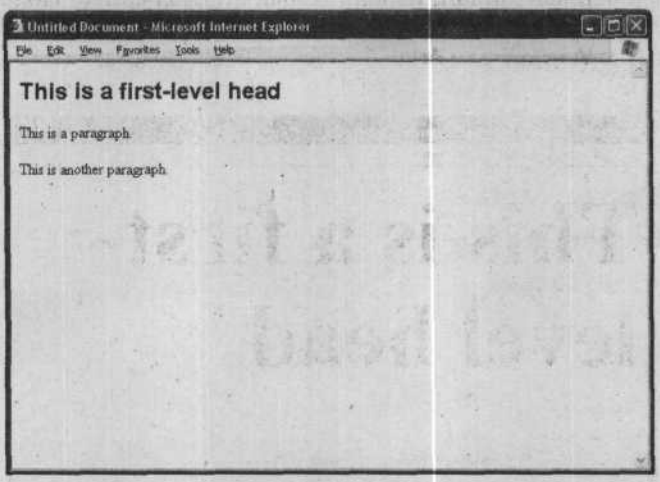


Рис. 42.1. Данный HTML-документ содержит таблицу стилей, в которой определяются стили для тегов h1 и p

Если таблицу стилей изменить следующим образом:

```
<style type="text/css">

   h1 {
      font-family: "Times New Roman";
      font-weight: bold;
      font-size: 96px;
   }

   p {
      font-family: Verdana;
      font-size: 8px;
   }

</style>
```

представление Web-страницы изменится и будет выглядеть, как на рисунке 42.2.

Определения стилей могут быть сложными или простыми в зависимости от решаемых задач. Другими словами, нет необходимости указывать значения для всех возможных атрибутов в определении. В предыдущих примерах определения не содержат указаний об интервалах или цвете текста, которые, конечно же, могут задаваться в каскадных таблицах стилей. При пропусках информации в определении браузер заполняет их своими параметрами, используемыми по умолчанию. Поэтому, если специально не указано, что текст абзаца должен отображаться красным шрифтом, браузером

будет выводиться черный шрифт, используемый по умолчанию. Подобным образом, если в предыдущих примерах браузеру не сообщить специально о том, что надо использовать шрифты **Verdana** или **Arial**, браузером будет подставляться шрифт по умолчанию.

Рис. 42.2. При изменении определения стиля внешний вид Web-страницы изменяется

Таблицы стилей называются каскадными благодаря тому, что дочерние элементы наследуют стили своих родителей, также, как человеческие дети наследуют гены своих родителей.

Родительский элемент – это HTML-тег, содержащий другой HTML-тег. **Дочерний элемент** – это HTML-тег, который содержится в родительском элементе. Возьмем данный блок HTML-кода:

```
<p>
   Welcome back, my friends to the <em>Show That Never Ends</em>!
</p>
```

Тег абзаца (<p>) является родительским, а тег выделения () – дочерним, поскольку тег выделения находится между открывающим и закрывающим тегами абзаца. А в этом случае:

```
<body>
   <p>
      Welcome back, my friends to the <em>Show That Never Ends</em>!
</p>
</body>
```

тег выделения по-прежнему является дочерним тегом абзаца, но теперь и тег абзаца и тег выделения являются дочерними элементами тега основного текста (<body>).

Каскадирование действует следующим образом: дочерний элемент наследует стиль родительского элемента. Таким образом, любой стиль, присвоенный тегу <body>:

```
<style type="text/css">

body {
    font-family: Arial;
}

</style>
```

автоматически применяется ко всем его дочерним элементам, которыми, как показано на рисунке 42.3, в данном примере являются тег абзаца и тег выделения. Даже если по умолчанию в браузере используется шрифт **Times New Roman**, не надо сообщать браузеру о необходимости использовать шрифт **Arial** из родительского тега <body>.

Совет

Некоторые атрибуты каскадных таблиц стилей не переходят от родительских элементов к дочерним, но это нормально, поскольку не требуется, чтобы дочерние элементы наследовали эти свойства. Возьмем атрибут верхнего поля, который определяет размер верхнего поля страницы. Этот атрибут обычно указывается в теге <body>. Вряд ли вы захотите, чтобы все абзацы, заголовки, изображения и другие дочерние элементы тега <body> имели одинаковое верхнее поле или чтобы содержимое страницы скучивалось вверху окна браузера.

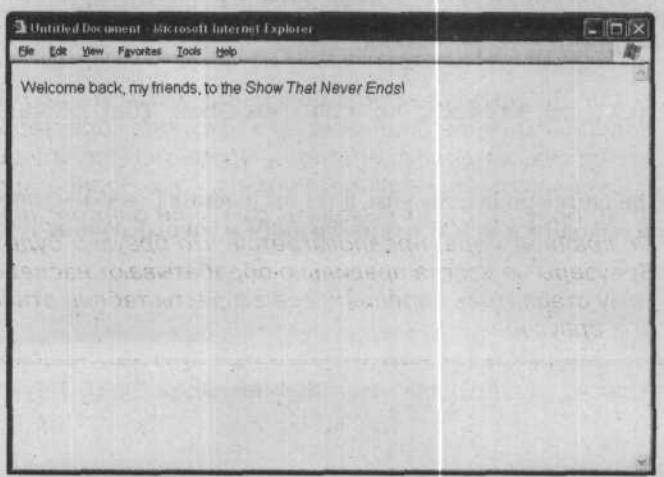


Рис. 42.3. В каскадных таблицах стилей дочерний элемент наследует стиль своих родительских элементов. Браузеру не надо указывать об отображении текста абзацев шрифтом **Arial**, *если в определении стиля для тега <body> уже задан шрифт* **Arial**

Каскадная природа CSS (CSS – Cascading Style Sheets) означает, что определения стилей дочерних элементов дополняют или уточняют стили родительских элементов. В этой таблице стилей:

```
<style type="text/css">

body {
    font-family: Arial;
}

p {
    font-style: italic;
}

</style>
```

текст абзацев отображается курсивным шрифтом **Arial**, тогда как другие основные элементы – например, теги `h1` – отображаются обычным шрифтом **Arial**. Заголовки первого уровня являются дочерними элементами тега `<body>`, как абзацы, а все дочерние элементы тега `<body>` отображаются шрифтом **Arial**. Однако в этой таблице стилей только дочерние элементы абзаца имеют дополнительное уточнение об отображении курсивом, поэтому браузером это дополнительное определение стиля применяется к тегам `<p>`, а остальные дочерние теги остаются без изменения. На рисунке 42.4 показан результат применения данной таблицы стилей.

Совет

Когда определение стиля дочернего элемента противоречит определению стиля родительского элемента, браузером используется стиль дочернего элемента. Например, если в родительском определении указан шрифт **Verdana**, *а в дочернем определении —* **Arial**, *браузер будет выводить дочерний элемент шрифтом* **Arial**. *По крайней мере, предполагается, что браузер будет так делать. Браузеры не всегда правильно обрабатывают наследование, поэтому старайтесь проверять все эффекты таблиц стилей в различных браузерах.*

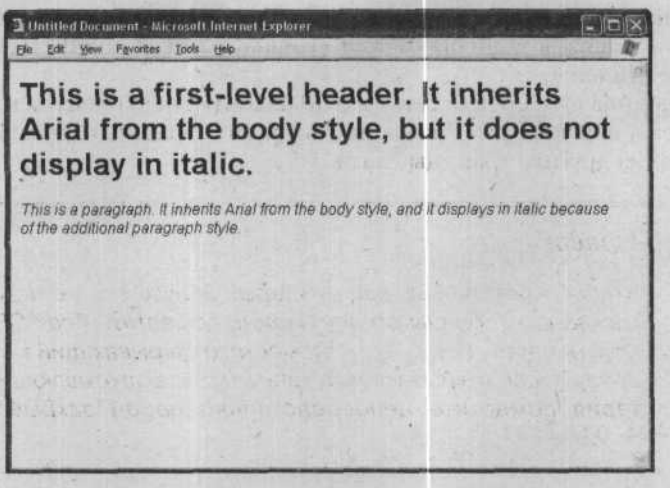


Рис. 42.4. Используйте дополнительные определения стилей для уточнения и расширения стилей, которые наследуются дочерними элементами от своих родителей

Внедрение таблиц стилей

Чтобы внедрить таблицу стилей на HTML-страницу, расположите CSS-код между тегами стиля и поместите весь блок стиля внутрь раздела заголовка страницы, подобно следующему примеру:

```
<html>
   <head>
      <title>Моя HTML-страница</title>
      <style type="text/css">

         body {
            font-family: Arial;
         }
         p {
            font-style: italic;
         }
      </style>
   </head>
   <body>

      <!—Содержимое страницы располагается здесь. —>
```

```
    </body>
</html>
```

Встроенная таблица стилей действует в отношении всех элементов на той странице, где она внедрена, и не действует на элементы других страниц, если только не вставить таблицу стилей на другие страницы сайта.

Совет

Чтобы спрятать CSS-код от старых браузеров, не поддерживающих каскадные таблицы стилей, можно поместить блок CSS-кода в теги комментария (<!–, –>). Поместите открывающий тег комментария сразу после открывающего тега `style`*, а закрывающий тег комментария поместите непосредственно перед закрывающим тегом* `style`*.*

Импорт таблиц стилей

Чтобы не включать одинаковые таблицы стилей на каждой странице сайта, можно поместить таблицу стилей в отдельный файл, сохранить этот файл с расширением **.css**, а затем импортировать данную таблицу стилей на все страницы, где она нужна. Применяя такой способ, требуется только один раз написать таблицу стилей, и вам не надо будет копировать и вставлять один и тот же код. Если потребуется изменить стиль, вам нужно будет лишь изменить его в одном месте – в CSS-файле. Если же вы внедряете таблицу стилей, необходимо будет обновлять CSS-код на всех страницах сайта, где эта таблица вставлена.

Чтобы импортировать внешний CSS-файл, пользуйтесь правилом `@import` вместо внедренной таблицы стилей:

```
<html>
    <head>
        <title>Моя HTML-страница</title>
        <style type="text/css">

            @import url("styles/mystyles.css");

        </style>
    </head>
    <body>

        <!– Здесь располагается содержимое страницы. –>
```

```
    </body>
</html>
```

После правила `@import` укажите адрес в Интернете таблицы стилей, используя для этого конструкцию `url("путь")`. Путь может указываться относительно документа, корневой папки или же может приводиться абсолютный путь, подобно пути для ссылки.

Можно импортировать несколько таблиц стилей:

```
<style type="text/css">

    @import url("styles/styles01.css");
    @import url("styles/styles02.css");
    @import url("styles/styles03.css");

</style>
```

Если импортированные таблицы стилей требуется дополнить дополнительными, характерными только для конкретной страницы, стилями, используйте следующий код:

```
<style type="text/css">

    @import url("styles/styles01.css");
    @import url("styles/styles02.css");
    @import url("styles/styles03.css");

    p {
        color: #ff0000;
    }

</style>
```

Если одна из импортированных таблиц стилей содержит стиль абзаца и необходимо, чтобы браузер использовал вместо него стиль, специфический для данной страницы, задайте стиль для страницы следующим образом:

```
<style type="text/css">

    @import url("styles/styles01.css");
    @import url("styles/styles02.css");
    @import url("styles/styles03.css");
```

```
    p {
        color: #ff0000;
        !important;
    }

</style>
```

Совет

Браузеры старых версий не понимают правило стиля `@import`, но этот недостаток можно обратить в достоинство. Внедрите базовые определения стилей в HTML-код страницы, а затем используйте `@import` для внесения расширенных стилей, которые не работают в старых браузерах. Только не маркируйте основные, внедренные стили объявлением `!important`, иначе общие стили заменят усовершенствованные версии, добавленные командой `@import`.

Объявление `!important` сообщает браузеру, что в случае возникновения конфликта надо использовать данный стиль. Браузер игнорирует любые стили абзацев в импортированных таблицах стилей только на этой странице.

Использование HTML-атрибута стиля

Написание таблицы стилей оправданно, когда имеется глобальный стиль, который применяется ко всем элементам определенного типа. А когда существует одноразовый стиль, который применяется к конкретному экземпляру элемента на странице и нигде больше, имеет смысл поместить определение в атрибут стиля для тега элемента.

Скажем, вы хотите, чтобы определенный абзац отображался полужирным, красным шрифтом, тогда как остальные абзацы должны отображаться шрифтом, который используется браузером по умолчанию. Вы не хотите создавать таблицу стилей для тега абзаца, поскольку это приведет к изменению шрифта всех абзацев на полужирный, красный шрифт. Эта задача решается с помощью следующего кода:

```
<p>This is a normal paragraph, neither bold nor red.</p>

<p>This is another normal paragraph, neither bold nor red.</p>

<p style="font-weght: bold; color: #FF0000;"">This is a special
paragraph, both bold and red.</p>

<p>Back to normal paragraph style here, neither bold nor red.</p>
```

Атрибут стиля в теге абзаца применяет текущее определение каскадных таблиц стилей к этому конкретному абзацу, а также ко всем дочерним элементам абзаца, и, как показано на рисунке 42.5, представление возвращается к нормальному стилю, как только браузеру встречается закрывающий тег абзаца.

Совет

Обратите внимание, что определение внутри атрибута стиля соответствует правилам написания каскадных таблиц стилей, даже если в окружающем коде соблюдаются синтаксические правила HTML.

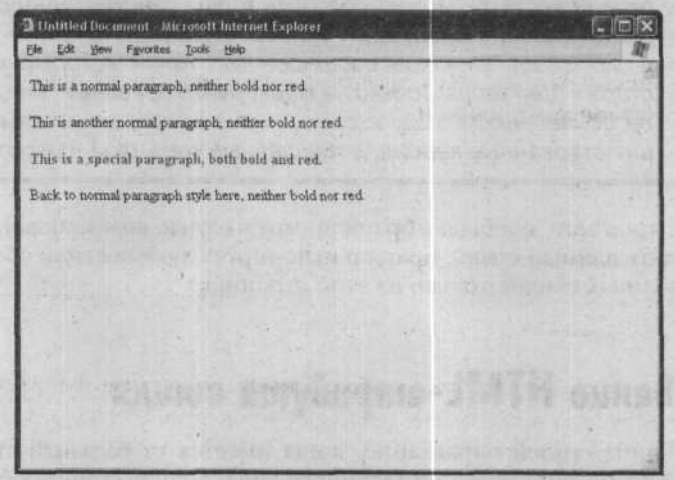


Рис. 42.5. Помещайте CSS-определение внутрь атрибута стиля HTML-тега, чтобы создать стиль одноразового употребления для этого тега

ГЛАВА 43.
Замена тегов форматирования каскадными таблицами стилей

Теги форматирования – это HTML-теги, которые в явном виде задают шрифтовые эффекты, такие, как полужирное начертание, подчеркивание и курсив. В отличие от элементов фразы, теги форматирования не определяют структуру размечаемого текста. Они просто показывают, что определенная строка текста должна отображаться полужирным шрифтом, курсивом или с подчеркиванием.

К тегам форматирования относятся: `b` (для полужирного шрифта), `i` (для курсива), `u` (для подчеркивания) и `s`, или `strike`, (для перечеркивания). На рисунке 43.1 показано действие этих тегов.

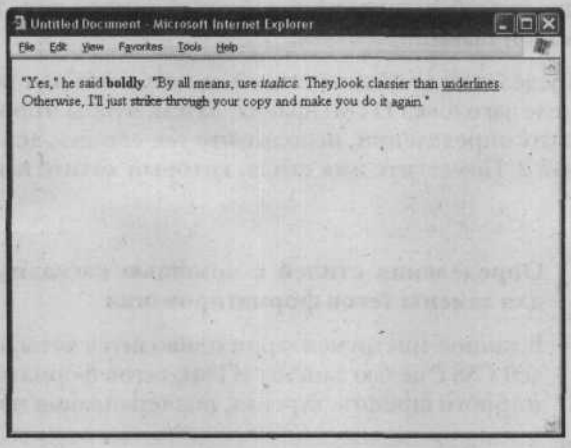


Рис. 43.1. В этом абзаце используются теги форматирования для получения полужирного начертания, курсива, подчеркивания и перечеркивания

Определение

Тег форматирования — это HTML-тег, который описывает внешний вид блока содержимого, но не идентифицирует ни структуру, ни назначение содержимого.

Совет

Избегайте использования подчеркивания, независимо от способа его реализации. Подчеркивание практически всегда указывает на гиперссылки. Если на странице подчеркивается текст, не являющийся ссылкой, посетители сайта будут пытаться щелкнуть на этом тексте, полагая, что это ссылка. Если вы выделили подчеркнутый текст другим цветом, возможно, вам придется просить посетителей направлять свои электронные сообщения, полные возмущений, Web-мастеру. Лучше полностью исключить подчеркивания текста, не являющегося ссылками, и использовать для выделения текста курсив вместо подчеркивания.

Поскольку теги форматирования ничего не должны делать со структурой, и т.к. считается, что HTML должен быть языком разметки только структуры, форматирование с помощью этих тегов теперь считается ошибкой разметки. Вместо этого следует применять определения стилей в каскадных таблицах стилей так, как это описано в инструментарии данной главы.

Скопируйте эти определения в таблицу стилей своего сайта или поместите их между тегами стиля в разделе заголовка HTML-файла. Затем, чтобы отформатировать часть текста с помощью этих определений, используйте тег , действие которого показано на рисунке. 43.2. Поместите имя стиля, который хотите использовать, в атрибут класса.

Инструментарий

Определения стилей с помощью каскадных таблиц стилей для замены тегов форматирования

В данном инструментарии приводится четыре определения стилей CSS с целью замены HTML-тегов форматирования для полужирного шрифта, курсива, подчеркивания и перечеркивания.

```
.b {
    font-weight: bold;
}
.i {
    font-style: italic;
}
.u {
    text-decoration: underline;
}
.s {
    text-decoration: line-through;
}
```

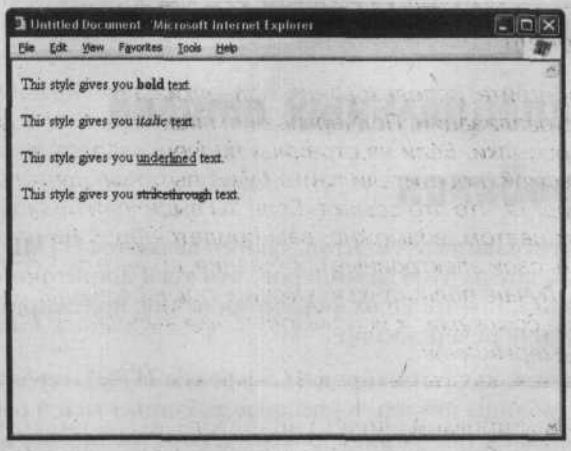


Рис. 43.2. Вместо тегов форматирования используйте определения стилей каскадных таблиц стилей. Такие определения действуют также хорошо, как теги форматирования, и позволяют вновь использовать HTML как чисто структурный язык

Листинг 43.1. Исходный код страницы, вид которой в браузере показан на рисунке 43.2

```
<p>This style gives you <span class="b">bold</span> text.</p>

<p>This style gives you <span class="i">italic</span> text.</p>

<p>This style gives you <span class="u">underlined</span> text.</p>

<p>This style gives you <span class="s">strikethrough</span>
text.</p>
```

Совет

Сокращенные имена классов стилей — `b`, `i`, `u`, `s` — образованы от имен соответствующих тегов форматирования с тем, чтобы облегчить Web-дизайнерам старой школы переход на использование каскадных таблиц стилей. Если предпочтение отдается более содержательным именам, скажем **bold** *или* **italic**, *ничто не мешает поступить в соответствии со своими предпочтениями. Проверьте, чтобы в атрибутах классов для тегов `<span>` приводились измененные имена классов.*

ГЛАВА 44.
Создание специальных стилей форматирования

В качестве предпочтительных элементов, какими являются HTML-теги форматирования в определенных кругах Web-дизайнеров, эти теги достаточно ограничены. Они предоставляют лишь основные виды форматирования: полужирное начертание, курсив, подчеркивание и перечеркивание.

В главе 43 описывается, как сымитировать эффекты HTML-тегов форматирования с помощью простой таблицы стилей. Каскадные таблицы стилей открывают дорогу ко многим другим возможностям форматирования, перечисляемым в таблице 44.1.

Совет

Тег форматирования — это HTML-тег, который описывает внешний вид блока содержимого, но не идентифицирует ни структуру, ни назначение содержимого. К тегам форматирования в HTML относятся `b` *(полужирный),* `i` *(курсив),* `u` *(подчеркивание) и* `s` *или* `strike` *(перечеркивание).*

Табл. 44.1. Стандартные CSS-атрибуты форматирования

CSS-атрибуты	Возможные значения	Пример
`font-weight`	`bold, bolder, lighter, normal`	`font-weight: bolder;`
`font-style`	`italic, oblique, normal`	`font-style: oblique;`
`font-variant`	`normal, small-caps`	`font-variant: small-caps;`
`text-decoration`	`underline, overline, line-through, none`	`text-decoration: overline;`
`text-transform`	`capitalize, uppercase, lowercase, none`	`text-transform: capitalize;`

Совет

Текст с большей или меньшей плотностью шрифта вряд ли будет отличаться от обычного текста на экране, но, возможно, разница будет заметна на распечатке. Кроме того, в существующих браузерах наклонный шрифт (oblique) выглядит также, как курсивный шрифт (italic), хотя в будущих версиях браузеров наклонный текст, возможно, будет выглядеть по-другому.

Хотите черту сверху вместо подчеркивания? Нет проблем – см. рис. 44.1.

Листинг 44.1. Исходный код страницы, вид которой в браузере показан на рисунке 44.1

```
<style type="text/css">

   .o {
      text-decoration: overline;
   }

</style>

<p>
   <q>I thought we went <span class="o">over</span> this,</q>  said
Captain Steward.
</p>
```

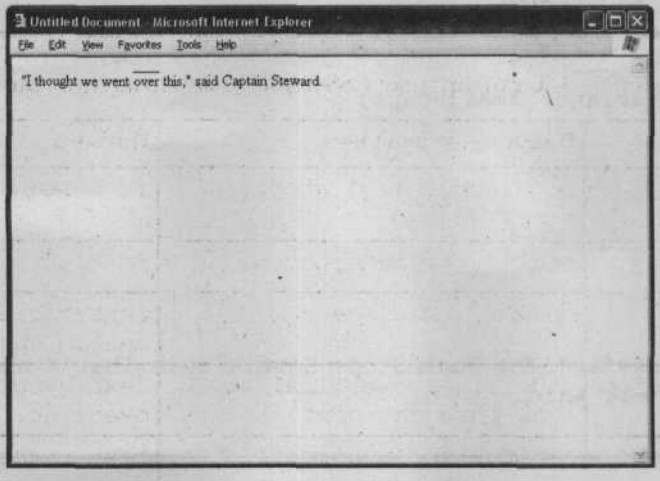


Рис. 44.1. Использование каскадных таблиц стилей (CSS) для создания черты сверху вместо подчеркивания

Строку текста можно также визуализировать в стиле **small-caps** (капитель), который преобразует строчные буквы исходного кода в уменьшенные прописные буквы в окне браузера (рис. 44.2).

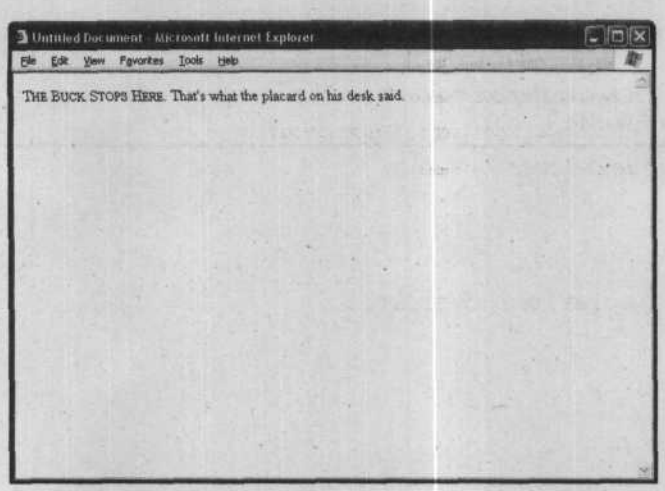


Рис. 44.2. Строчные буквы исходного кода преобразуются в окне браузера в заглавные

Листинг 44.2. Исходный код страницы, вид которой в браузере показан на рисунке 44.2

```
<style type="text/css">

   .sc {
      font-variant: small-caps;
   }

</style>

<p>
   <span class="sc">The Buck Stops Here.</span>That's what the placard on his desk said.
</p>
```

Подобным образом, используйте значения `capitalize`, `uppercase` и `lowercase` с атрибутом `text-transform`, чтобы изменить регистр букв исходного кода. В тексте со значением атрибута `capitalize` первая буква каждого слова выводится на экран как заглавная. Как показано на рис. 44.3, текст со значением атрибута `uppercase` отображается в браузере со всеми заглавными буквами, а текст со значением атрибута `lowercase` – со всеми строчными буквами, независимо от регистра букв в исходном коде.

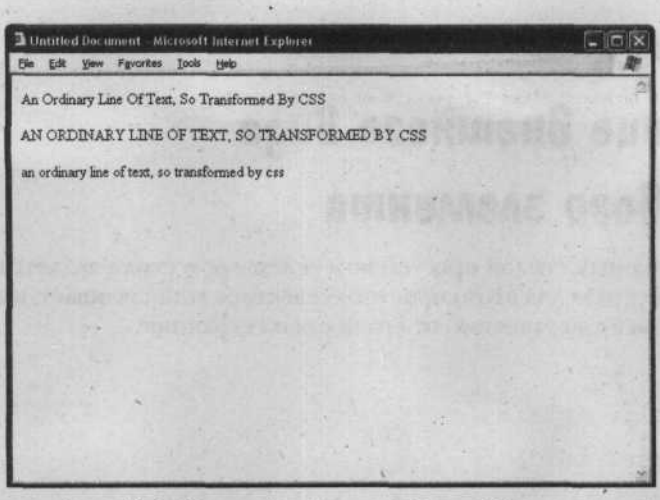


Рис. 44.3. Изменяйте регистр букв предложения с помощью значений атрибута `text-transform`

Листинг 44.3. Исходный код страницы, вид которой в браузере показан на рисунке 44.3

```
<p style="text-transform: capitalize;">an ordinary line of text, so
transformed by CSS</P>

<p style="text-transform: uppercase;">an ordinary line of text, so
transformed by CSS</P>

<p style="text-transform: lowercase;">an ordinary line of text, so
transformed by CSS</P>
```

ГЛАВА 45.
Изменение внешнего вида текстового элемента

В каскадных таблицах стилей простейшим селектором стиля является имя HTML-тега. Определение стиля для этого простого селектора устанавливает, как браузер будет отображать элемент, независимо от его места на странице:

```
h1 {
    font-family: Arial;
    font-weight: bold;
}

p {
    font-family: Verdana;
}
```

Таким образом, выполняется переопределение общего внешнего вида HTML-тегов текста. Как показано в таблице 45.1, каскадные таблицы стилей позволяют выбирать гораздо более специфические селекторы для всей серии тегов, чтобы получить требуемый шрифтовой эффект.

Определение контекстно-зависимых селекторов

Контекстно-зависимый селектор – это селектор стиля, который идентифицирует все HTML-теги определенного типа, находящиеся где-нибудь внутри тега другого типа. Этот тип селектора используется для установки стиля всех тегов выделения, которые встречаются, например, внутри таблицы, либо всех тегов `<strong>`, встречающихся внутри упорядоченного списка. В этих примерах браузер игнорирует теги выделения, которые не появляются в таблице, или теги `<strong>`, которые не находятся в упорядоченном списке, если только для этих случаев не предоставлены дополнительные правила стилей.

Дочерний и братский селекторы не всегда работают хорошо в браузере Internet Explorer.

Табл. 45.1. CSS-селекторы для HTML-тегов

Селектор	Описание	Пример синтаксиса	Описание примера
Простой	Все HTML-теги одного типа	`p`	Все абзацы
Контекстно-зависимый	Все HTML-теги одного типа, которые находятся где-нибудь внутри тега другого типа	`table em`	Все теги выделения, которые находятся где-нибудь внутри таблицы
Дочерний	Все HTML-теги одного типа, которые имеют в качестве родителя какой-нибудь тег другого типа	`td > p`	Все абзацы, для которых ячейки таблицы являются их прямыми родителями
Братский	Все HTML-теги одного типа, которые следуют за тегом другого типа, но не являются дочерними элементами этого другого тега	`h1 + p`	Все абзацы, которые следуют сразу за заголовками первого уровня

Определение

Контекстно-зависимый селектор — это селектор стиля, идентифицирующий все HTML-теги определенного типа, которые находятся где-нибудь внутри тега другого типа, например, все теги `<strong>`, находящиеся внутри упорядоченного списка.

Синтаксис для этого типа селектора выглядит следующим образом:

```
тег-контейнер целевой-тег
```

Как можно видеть, имена тегов разделяются с помощью одиночного пробела. Таким образом, чтобы определить стиль для всех тегов `<strong>`, располагающихся где-нибудь внутри упорядоченного списка, правило стиля должно выглядеть подобно следующему примеру:

```
ol strong {
    font-style: italic;
}
```

Только те теги `<strong>`, которые находятся внутри упорядоченного списка, отображаются браузером полужирным курсивом. Возможно, у вас возникнет вопрос, почему полужирным, если в стиле ничего не указывает на полужирное начертание шрифта? Потому что по умолчанию браузеры воспроизводят текст, размеченный тегами `<strong>`, полужирным шрифтом. Как видно из рисунка*45.1, остальной текст на странице, размеченный тегами `<strong>`, отображается полужирным шрифтом без дополнительного эффекта курсива.

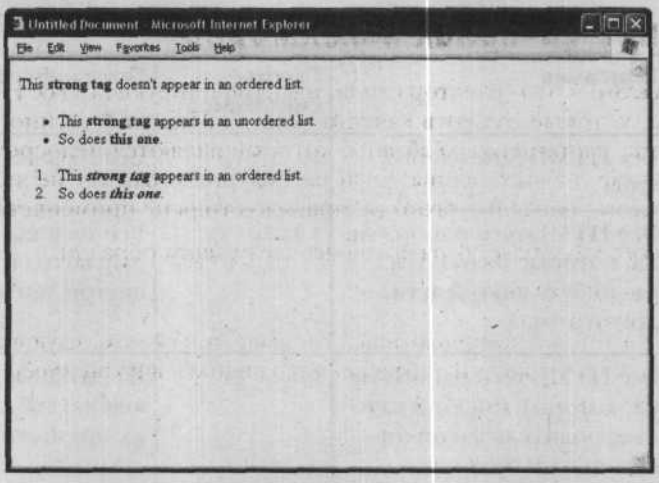


Рис. 45.1. При использовании контекстно-зависимого тега дополнительный стиль применяется только к тем тегам `<strong>`*, которые находятся внутри упорядоченного списка*

С помощью контекстно-зависимых селекторов можно действовать действительно избирательно. Скажем, вы хотите применить стиль только к тем тегам выделения, находящимся внутри тегов абзацев, которые в свою очередь находятся внутри ячеек таблицы, расположенных внутри элементов `div`:

```
div td p em {
    font-weight: bold;
}
```

Как показано на рисунке 45.2, браузер следует инструкциям буквально.

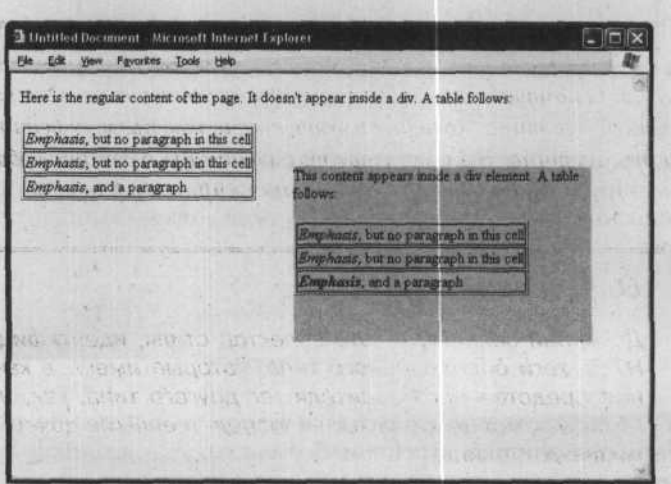


Рис. 45.2. Соберите вместе ряд из нескольких тегов для создания специфичного контекстно-зависимого селектора. Этот стиль воздействует только на теги `<em>`*, расположенные внутри тегов абзацев, внутри ячеек таблиц, внутри элементов* `div`

Определение дочерних селекторов

Дочерний селектор – это селектор стиля, идентифицирующий все HTML-теги определенного типа, которые имеют в качестве своего непосредственного родителя тег другого типа, как, например, все абзацы, которые являются непосредственными дочерними объектами ячеек таблицы. Если рассматриваемый тег не является прямым дочерним объектом родительского тега, правило стиля не применяется.

Синтаксис данного типа селектора выглядит следующим образом:

```
родительский-тег > дочерний-тег
```

Разделяйте имена тегов с помощью знака «больше чем» (>). Следующее правило стиля действует только в отношении абзацев, находящихся внутри ячеек таблицы:

```
td > p {
   font-weight: bold;
}
```

Как показано на рисунке 45.3, все остальные абзацы отображаются браузером невыделенными.

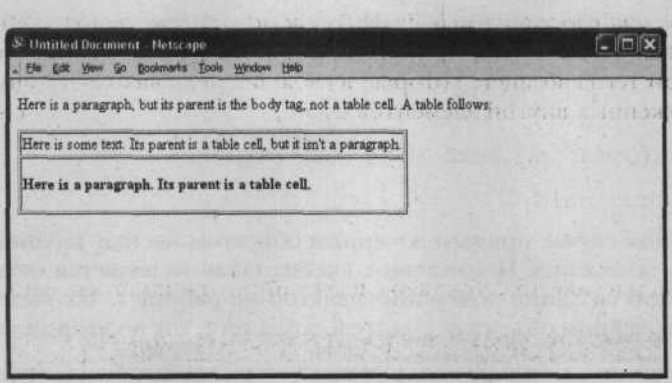


Рис. 45.3. С помощью дочернего селектора выделяются только те теги определенного типа, которые имеют в качестве прямых родителей теги другого типа. В этом случае только абзацы – дочерние объекты ячеек таблицы, выделяются полужирным шрифтом

Определение

Дочерний селектор — это селектор стиля, идентифицирующий все HTML-теги определенного типа, которые имеют в качестве своего непосредственного родителя тег другого типа, как, например, все абзацы, которые являются непосредственными дочерними объектами ячеек таблицы.

Совет

Браузер Microsoft Internet Explorer некорректно обрабатывает пример, показанный на рисунке 45.3, поэтому очень тщательно проверяйте дочерние селекторы в Internet Explorer.

Здесь важна точная генеалогия тега. Чтобы дочерний селектор работал, рассматриваемый тег должен быть **прямым** дочерним объектом своего родителя. Таким образом, правило стиля, подобное следующему примеру:

```
td > em {
    font-weight: bold;
}
```

не оказывает воздействия на следующий блок HTML-кода:

```
<td>
  <p>This is a paragraph with <em>emphasis</em>.</p>
</td>
```

но работает корректно в данном случае:

```
<td>
  This is unformatted text with <em>emphasis</em>.
</td>
```

Почему? В первом случае прямым дочерним объектом ячейки таблицы является тег абзаца, а не тег выделения. Несомненно, ячейка таблицы является «дедом» тега выделения, но в такой ситуации дочерний селектор не работает. Тег выделения должен быть прямым дочерним объектом родительского тега, как во втором случае.

Чтобы воздействовать на все теги выделения внутри ячеек таблиц, независимо от точной генеалогии, воспользуйтесь вместо дочернего селектора контекстно-зависимым селектором:

```
td em {
    font-weight: bold;
}
```

Это правило стиля действует в обоих предыдущих случаях, поскольку в обоих этих случаях тег выделения находится где-то внутри ячейки таблицы.

Вернемся к дочерним селекторам. Можно задавать полную генеалогию тега в качестве селектора для получения действительно особенных результатов. Это правило стиля выбирает только те теги выделения, которые являются дочерними объектами тегов <h1>, являющихся в свою очередь дочерними объектами ячеек таблицы:

```
td > h1 > em {
    font-weight: bold;
}
```

Определение братских селекторов

Братский селектор – это селектор стиля, идентифицирующий HTML-теги определенного типа, за которыми следует тег другого типа, но при этом идентифицируемые теги не являются дочерними объектами этого тега. Рассмотрим тег абзаца, следующий за заголовком первого уровня внутри ячейки таблицы. Абзац не является дочерним объектом заголовка, поскольку тег абзаца не находится внутри тега заголовка. Эти теги являются братскими, подобно брату и сестре. Их общий родитель – ячейка таблицы, которая содержит оба эти тега:

```
<td>
   <!- This table cell is the proud parent of these two children.
->
    <h1>I am the older sibling. I am way too serious for my own
good.</h1>
<p>I am the younger sibling. I always get my own way.</p>
</td>
```

Таким образом, для выбора всех абзацев, следующих за заголовками первого уровня, используйте следующий синтаксис:

```
старший-брат + младший-брат
```

для правила стиля, которое выглядит подобно следующему примеру:

```
h1 + p {
   font-weight: bold;
}
```

Как показано на рисунке 45.4, абзацы, следующие за заголовками первого уровня, отображаются полужирным шрифтом, тогда как к другим абзацам это выделение не применяется.

Определение

Братский селектор — это селектор стиля, идентифицирующий все HTML-теги определенного типа, которые следуют за тегом другого типа, но не являются дочерними элементами этого тега, как, например, абзацы, следующие за заголовками первого уровня.

Обратите внимание, что второй абзац под заголовком первого уровня не выделяется полужирным шрифтом. Почему? Потому что фактически этот абзац следует за абзацем, а не за заголовком первого уровня. Братский селектор выбирает только тег, следующий непосредственно за ним, и только тогда, когда второй тег не является дочерним объектом первого тега. Таким образом, приводимое ниже правило стиля:

```
td + em {
   font-weight: bold;
}
```

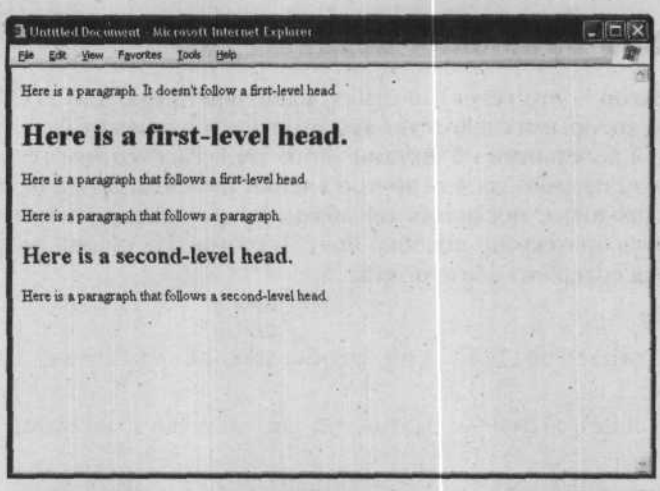


Рис. 45.4. Используйте братский селектор для воздействия на теги, которые следуют за другими тегами, но не являются дочерними объектами этих тегов

не оказывает воздействия на следующий блок HTML-кода:

```
<td>
    This is unformatted text with <em>emphasis</em>.
</td>
```

Как и с другими типами селекторов, братские селекторы позволяют достигать высочайшего уровня точности в управлении тегами. Это правило стиля действует только в отношении абзацев, следующих за заголовками четвертого уровня, которые следуют за заголовками третьего уровня и т.д.:

```
h1 + h2 + h3 + h4 + p {
    font-weight: bold;
}
```

Смешивание селекторов разных типов

Каскадные таблицы стилей обладают достаточной гибкостью, позволяющей применять одно и то же определение стиля к нескольким селекторам разных типов. Просто разделите селекторы с помощью запятых. Рассмотрим следующий пример:

```
p1, h1, h2 + h3, td > strong {
    color: #FF0000;
}
```

Данное правило стиля применяется ко всем абзацам, всем заголовкам первого уровня, всем заголовкам третьего уровня, являющимся братьями заголовков второго уровня, и ко всем тегам `<strong>`, являющимся дочерними элементами ячеек таблицы. Согласно правилу все эти элементы окрашиваются в красный цвет.

Следующая таблица стилей оказывает тот же самый эффект, но она не отличается лаконичностью:

```
p {
    color: #FF0000;
}

h1 {
    color: #FF0000;
}

h2 + h3 {
    color: #FF0000;
}

td > strong {
    color: #FF0000;
}
```

Конечно, используйте длинную форму записи, если для вас она представляет более понятное описание стилей. Но после того как вы привыкните к синтаксису каскадных таблиц стилей, не забудьте, что можно написать очень эффективные правила стилей, совершенно не жертвуя точностью определений.

ГЛАВА 46.
Определение стилей классов

В главе 45 объяснялось, как с высокой точностью переопределять внешний вид текста, размеченного HTML-тегами. А что происходит, если требуется определить стиль, который не обязательно должен применяться к какому-либо определенному HTML-тегу или логической последовательности тегов? Не стоит бояться. Каскадные таблицы стилей выручат вас. Просто определите свой собственный селектор стиля, называемый **классом**.

Определение

Класс — это специально создаваемый селектор стиля, который не обязательно должен применяться к какому-либо определенному HTML-тегу или к конкретной последовательности тегов.

Правило стиля для типичного класса выглядит похожим на следующий пример:

```
.bolditalic {
    font-weight: bold;
    font-style: italic;
}
```

Придумайте имя для своего класса и поставьте точку перед ним, как это сделано в предыдущем фрагменте кода для имени класса `.bolditalic`.

Совет

При заполнении атрибута класса для HTML-тега не включайте точку в начале имени класса. Просто приведите имя без точки.

Класс каскадных таблиц стилей – это что-то типа клуба, в который может «вступить» любой HTML-тег, но членство допускается только по приглашению. Чтобы «пригласить» HTML-тег в клуб, заполните для тега атрибут класса, как в листинге 46.1.

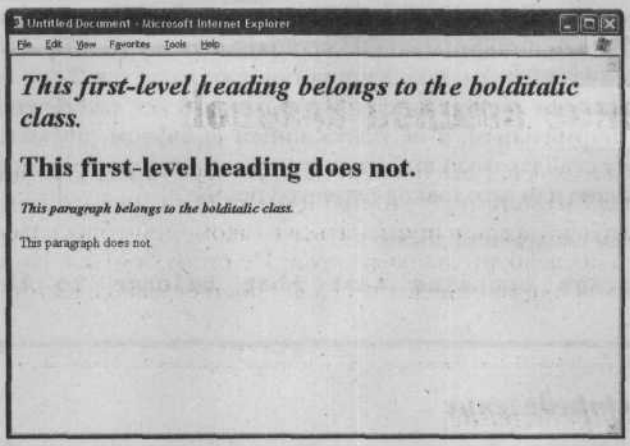


Рис. 46.1. После определения стиля класса поместите его имя в атрибут класса того тега, к которому хотите применить класс

Листинг 46.1. Исходный код страницы, вид которой в браузере показан на рисунке 46.1

```
<style type="text/css">

.bolditalic {
   font-weight: bold;
   font-style: italic;
}

</style>

<h1 class="bolditalic">This first-level heading belongs to the
bolditalic class.</h1>

<h1>This first-level heading does not.</h1>

<p class="bolditalic">This paragraph belongs to the bolditalic
class.</p>

<p>This paragraph does not.</p>
```

В коде листинга 46.1 следует отметить пару интересных моментов. Во-первых, стиль класса можно использовать для любого типа тега. Помните, что класс подобен клубу, в который может «вступить» любой тег. Во-вторых, стиль класса не переопределяет используемый по умолчанию внешний вид тегов в браузере, если только стиль прямо

не противоречит стандартным настройкам браузера. Заголовки первого уровня отображаются полужирным шрифтом по умолчанию, поэтому тег <h1>, входящий в клуб, просто становится курсивным; полужирным он уже является. Если бы стилем класса было задано нормальное начертание шрифта, тогда текст, размеченный тегом <h1>, отображался бы нормальным, а не полужирным шрифтом, поскольку определение стиля опровергает стандартный выбор полужирного начертания, используемого браузером по умолчанию для заголовков первого уровня.

Теперь рассмотрим следующий абзац:

```
<p>This paragraph contains text that belongs to the bolditalic
class.</p>
```

Определение

Диапазон — это сегмент содержимого, идентифицируемый в HTML-документе тегами .

Как сделать так, чтобы только слово «text» вошло в клуб bolditalic (полужирный курсив)? Атрибут класса нельзя определить, т.к. тогда весь текст абзаца стал бы полужирным и курсивным, а не только слово «text». Для решения этой задачи потребуется тег , поставленный в нужном месте:

```
<p>This paragraph contains <span>text</span> that belongs to the
bolditalic class.</p>
```

Тег не оказывает видимого эффекта в окне браузера. Его единственное назначение — это разметка сегмента или **диапазона** (span) содержимого. Теперь, когда тег установлен в надлежащем месте, можно сделать запрос на вхождение тега в клуб:

```
<p>This paragraph contains <span class="bolditalic">text</span>
that belongs to the bolditalic class.</p>
```

Результат выполненных действий показан на рисунке 46.2.

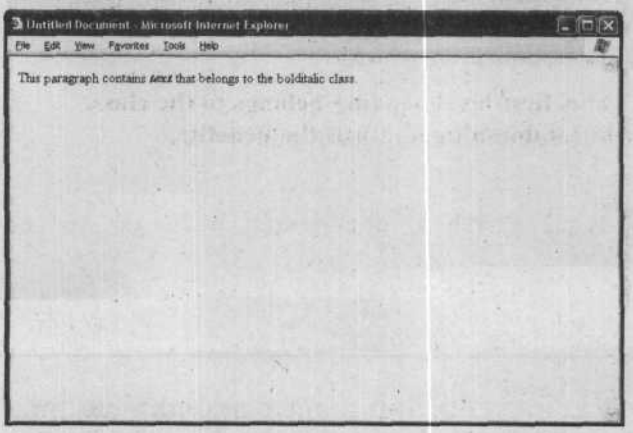


Рис. 46.2. Если в требуемом месте нет тега, поместите тег точно там, где он необходим, а затем внесите тег в свой стиль класса

Формирование привилегированного клуба

Если обычные стили классов подобны клубам, в которые может «вступить» любой тег, то можно также определить привилегированные клубы, в которые могут войти только теги определенного типа. Так можно создать стиль класса, который не воздействует на все абзацы в документе, а только на те абзацы, которые принадлежат классу. Заголовки первого уровня, заголовки второго уровня и все остальные не могут принадлежать этому классу, если только вы не добавили их в селектор стиля.

Такое правило стиля выглядит следующим образом:

```
p.bolditalic {
    font-weight: bold;
    font-style: italic;
}
```

На рисунке 46.3 показан результат действия этого стиля.

Листинг 46.2. Исходный код страницы, вид которой в браузере показан на рисунке 46.3

```
<p class="bolditalic">Only paragraphs can join the p.bolditalic
class.</p>

<p>This paragraph opts not to join, so it doesn't enjoy the bene-
fits of membership.</p>

<h1 class="bolditalic">This first-level heading belongs to the
class, but it doesn't get any of the benefits, because it isn't a
paragraph.</h1>
```

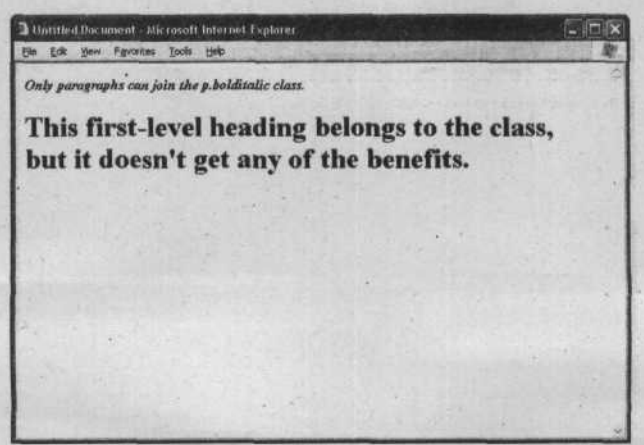


Рис. 46.3. Объединяя селектор тега с именем класса, можно создать стиль класса, которому могут принадлежать только теги этого определенного типа

Как можно заметить, ничто не препятствует тегу `<h1>` присоединиться к классу, но браузер полностью игнорирует этот тег, поскольку он не является абзацем. Конечно, браузер по-прежнему отображает заголовок первого уровня полужирным шрифтом, но это обусловлено тем, что данные элементы отображаются с полужирным начертанием по умолчанию. Полужирное начертание здесь никак не связано с таблицей стилей.

Совет

При заполнении атрибута класса не включайте селектор тега в имя класса, относящегося к тегу. Просто введите имя класса.

Совет

Любой тип селектора тега (простой, контекстно-зависимый, дочерний или братский) может иметь специальное обозначение класса. Просто добавьте точку и имя класса после селектора. Дополнительные сведения о типах селекторов смотрите в главе 45.

ГЛАВА 47.
Управление шрифтом

Обычно для отображения текста браузером на компьютерах с операционной системой Windows почти всегда используется по умолчанию шрифт **Times New Roman**. В HTML существует тег шрифта, позволяющий изменять гарнитуру для определенной части текста. Тег шрифта работает следующим образом:

```
<p>
   <font face="Arial">Этот абзац отображается шрифтом Arial.</font>
</p>
```

Значением атрибута `face` является имя шрифта, который вы хотите использовать.

Изменяя положение тегов `<font>`, можно применять шрифт к нескольким текстовым элементам подряд:

```
<font face="Arial">
   <h1>Этот заголовок отображается шрифтом Arial.</h1>
   <p>Как и этот абзац.</p>
   <p>Как и этот абзац.</p>
</font>
```

Но теперь, когда имеются каскадные таблицы стилей, нет необходимости использовать тег `<font>`. Атрибут `font-family` выполняет ту же функцию. Атрибут можно добавить в определение стиля HTML-тега:

```
<p>
   <font family: Arial;
</p>
```

или стиля класса:

```
.arialtext {
  <font family: Arial;
}
```

Каскадные таблицы стилей облегчают также управление шрифтом. Если правило стиля определено так, что все абзацы отображаются шрифтом **Arial**, как в первом примере каскадных таблиц стилей, можно забыть об этих раздражающих тегах `<font>`. При каждом добавлении на страницу абзаца он автоматически отображается со шрифтом **Arial**.

Совет

*Если имя шрифта состоит из нескольких слов, как **Times New Roman** или **Courier New**, удостоверьтесь в том, что имя шрифта помещено в кавычки в определении CSS-стиля.*

Теперь стало лучше. Скажем, вы хотите, чтобы весь текст на странице отображался шрифтом **Arial**. Просто поместите атрибут `font-family` в правило стиля для тега `<body>`:

```
body {
    <font family: Arial;
}
```

Устранение ошибок, связанных со шрифтами

Чтобы тег `<font>` и атрибут `font-family` работали корректно, на компьютере посетителя должен иметься указываемый вами шрифт. Не важно, имеется ли этот шрифт на вашем компьютере. Главное, чтобы шрифт присутствовал на компьютере посетителя.

Итак, как узнать, какие шрифты имеются на компьютере посетителя? Краткий ответ: узнать невозможно. Но пусть это не мешает вам сформулировать некоторые обоснованные предположения. Все типовые персональные компьютеры поставляются со стандартными шрифтами. Гарнитуры, перечисляемые в таблице 47.1, практически всегда имеются на большинстве компьютеров. Смотрите также рисунок 47.1.

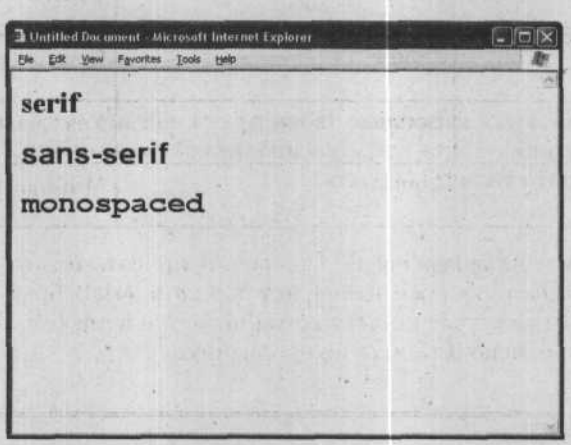


Рис. 47.1. Шрифт с засечками отличается наличием мелких украшений на концах букв, а шрифт без засечек не имеет таких украшений. В моноширинном шрифте все буквы имеют одинаковые интервалы между знаками для имитации стиля печатной машинки

Определение

*Шрифт **без засечек** (**sans serif**) такой, как **Arial**, не имеет небольших элементов украшения на концах букв. Шрифт с засечками (**serif**), например, **Times New Roman**, имеет такие засечки на концах букв. Моноширинный шрифт, например, **Courier New**, стилизован под шрифт печатающей машинки, где между любыми знаками и символами сохраняется одинаковый интервал.*

Чтобы дополнительно минимизировать риск отсутствия шрифта, можно задавать диапазон предпочитаемых шрифтов в теге `<font>` или CSS-атрибуте `font-family`. Просто укажите свои предпочтения в некотором порядке, разделяя названия шрифтов запятыми:

```
<font face="Arial, Helvetica, sans-serif">
```

или:

```
body {
  font-family: Arial, Helvetica, sans-serif;
}
```

Табл. 47.1. Наиболее распространенные шрифты, рекомендуемые для использования на Web-страницах

Шрифт Windows	Шрифт Macintosh	Категория
Times New Roman	**Times**	С засечками
Arial	**Helvetica**	Без засечек
Georgia	**Times**	С засечками
Verdana	**Geneva**	Без засечек
Courier New	**Courier**	Моноширинный

В этих случаях браузер сначала пытается визуализировать текст с помощью шрифта **Arial**. Если на компьютере посетителя нет шрифта **Arial**, браузер пробует найти **Helvetica**. Если и в этом случае попытка заканчивается неудачей, текст отображается шрифтом без засечек, используемым по умолчанию.

Совет

Продуманный список предпочитаемых шрифтов должен составляться таким образом: сначала указывайте шрифт Windows, за ним его эквивалент для операционной системы Mac, и в конце приводится обобщенная категория шрифта: ***serif*** *(с засечками),* ***sans serif*** *(без засечек) или* ***monospace*** *(моноширинный).*

Совет

*В конце списка предпочитаемых шрифтов всегда включайте категорию **serif**, **sans-serif** или monospaced в зависимости от типа шрифта, который желательно использовать для отображения текста браузером. Эти три общие категории шрифта не связаны с какой-либо конкретной гарнитурой. Точнее они соответствуют шрифтам с засечками, без засечек, моноширинному, используемым по умолчанию на компьютере посетителя. Вызов шрифта по его типу представляет собой безотказный вариант. Если все остальные шрифты, указываемые вами, будут отсутствовать на компьютере посетителя, вы гарантируете, что в браузере будет отображаться текст с помощью шрифта требуемой категории.*

Выбор наилучшей гарнитуры

Любой художник-дизайнер, занимающийся разработкой стиля книги, скажет вам, что гарнитура играет большую роль в создании профессионального вида книги. Если планируется сделать книгу в виде серьезного, авторитетного исследования либо беззаботного, легкого чтива, дизайнер пытается выбрать гарнитуру, которая визуально создает соответствующее настроение и ощущение. Вряд ли найдется много книг по философии Гегеля, которые выполнены шрифтом комиксов. Кроме того, выбрать надлежащую гарнитуру дизайнеру помогает размер текста. Некоторые гарнитуры хорошо подходят для малых размеров в силу своей удобочитаемости, тогда как другие гарнитуры по той же причине лучше использовать для больших размеров шрифтов.

Web-оформление не так сложно, как дизайн книг. Здесь гораздо меньше набор гарнитур, и количество устанавливаемых параметров не так велико, даже в таком надежном языке, как CSS (каскадные таблицы стилей), что сужает возможности дизайна, но одновременно облегчает задачу выбора правильной гарнитуры.

Ниже перечисляется несколько практических правил по выбору гарнитуры:

- Шрифты с засечками, такие, как **Times New Roman**, хороши для Web-сайтов с серьезным содержанием;
- Шрифты без засечек, такие, как **Arial**, лучше подойдут для Web-сайтов с легким, игровым содержанием;
- Шрифты с засечками и стандартными размерами делают большие фрагменты текста удобными для чтения;
- Широкие гарнитуры, такие, как **Verdana** или **Georgia**, лучше использовать для малых размеров шрифта. При больших размерах шрифта эти гарнитуры выглядят чрезмерно большими;
- Шрифты без засечек обычно чаще, чем шрифты с засечками, используют для заголовков.

ГЛАВА 48.
Управление размером шрифта

В главе 47 объяснялось, как с помощью тега `<font>` в HTML и атрибута `font-family` в каскадных таблицах стилей (CSS) можно задавать гарнитуру шрифта текста на Web-странице. Следуя аналогичным правилам, можно задавать размер шрифта в атрибуте `size` тега `<font>` или в атрибуте `font-size` каскадных таблиц стилей.

Использование тега <font>

Атрибут `size` тега `<font>` выглядит подобно следующему примеру:

```
<font face="Arial, Helvetica, sans-serif" size="4">
```

Значение, указываемое в атрибуте `size`, не является ни пикселами, ни пунктами, ни дюймами, ни футами, ни миллиметрами, ни милями. Вместо этого размер соответствует одному из семи предопределенных размеров текста в HTML. Обычный текст на Web-странице отображается с размером 3. Таким образом, данный текст с размером 4 на один размер больше обычного текста. Как показано на рисунке 48.1, минимальный размер текста равен 1, а максимальный размер – 7.

Другим возможным значением атрибута `size` является относительный размер, в котором размер шрифта выражается как некоторое число плюс или минус текущий размер. В языке HTML относительный размер описывается таким образом:

```
<font face="Arial, Helvetica, sans-serif" size="+1">
```

или

```
<font face="Arial, Helvetica, sans-serif" size="-3">
```

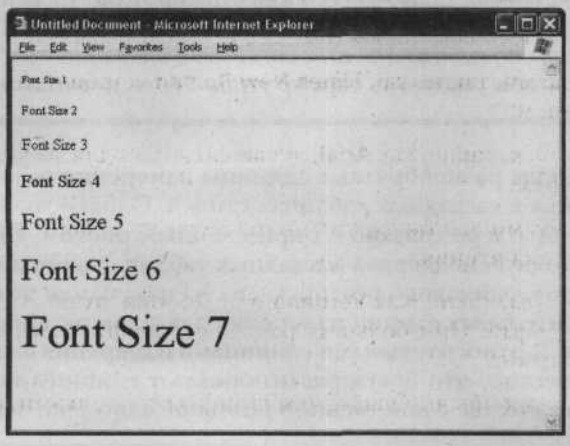


Рис. 48.1. При использовании тега `<font>` для выбора доступны только семь типов размеров, пронумерованных числами от 1 до 7

В первом примере шрифт увеличивается на один размер относительно размера предыдущего текста, тогда как во втором примере шрифт уменьшается на три размера. Минимальный относительный размер равен -6, а максимальный относительный размер равен +6.

Использование атрибута font-size

Если семь размеров шрифта кажутся вам нелепым ограничением, вы не одиноки в своих чувствах. Каскадные таблицы стилей спасают положение, предлагая практически неограниченное число размеров шрифта. Просто добавьте атрибут `font-size` в определение своего стиля и укажите значение и единицы измерения, которые хотите использовать:

```
p {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 18px;
}
```

Текст, размеченный тегами абзацев, использующими данное правило стиля, отображается в браузере так, как показано на рисунке 48.2.

Часто задаваемые вопросы

Можно ли получить более семи типов шрифта в HTML?

Сообразительные Web-дизайнеры пытаются выжать из HTML дополнительные размеры шрифта путем установки в теге `<font>` размера 7 в качестве начального значения и затем сразу увеличивая размер с помощью относительного тега шрифта. Если вы относитесь к таким умельцам, можете порадоваться своей хитрости, но, к сожалению, браузер не поддается на такие уловки. Максимальный размер, который можно получить с помощью HTML, равен 7. Браузером игнорируются относительные значения, которые добавляются, чтобы увеличить размер выше 7 (или уменьшить менее 1).

В таблице 48.1 сведены разнообразные единицы измерения размера шрифта, которые предоставляются в каскадных таблицах стилей. Однако знайте, что изменение размера шрифта в браузере связано с определенным риском. Поскольку браузеры предоставляют неполную поддержку каскадных таблиц стилей и поскольку не существует двух браузеров, одинаково реагирующих на изменение размеров, лучше придерживаться относительных единиц измерения для размера шрифта текста. Как показано в таблице 48.1, относительными единицами измерения являются `em`, `ex` и `px`; однако хорошо известно, что браузеры игнорируют единицы измерения `em` и `ex`. Вследствие этого в качестве единственной реальной альтернативы остаются старые добрые пикселы.

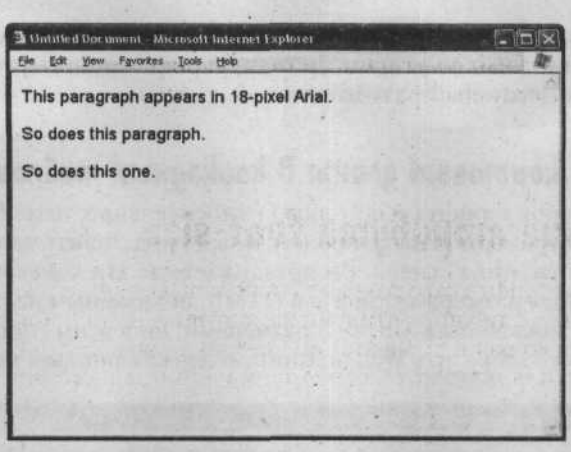


Рис. 48.2. С помощью каскадных таблиц стилей можно задавать любые размеры шрифта с разнообразными единицами измерения

Табл. 48.1. Единицы измерения длины в каскадных таблицах стилей

Единицы измерения	Описание	Категория
cm	Сантиметры	Абсолютные
em	Эм; 1 эм равен приблизительно ширине заглавной буквы M	Относительные
ex	X-высота; 1 ex равен приблизительно высоте строчной буквы x	Относительные
in	Дюймы	Абсолютные
mm	Миллиметры	Абсолютные
pc	Пайки (1 пайка содержит 12 пунктов и равна 4,23 мм)	Абсолютные
pt	Пункты	Абсолютные
px	Пикселы	Относительные

Относительные единицы измерения не являются относительными в том смысле, что их значения изменяются. Вместо этого фактический размер единиц измерения зависит от ряда факторов. Например, разрешающей способностью экрана определяется размер пиксела. Если на экране монитора отображается 800 x 600 пикселов, размер пикселов будет больше, чем на экране монитора, отображающего 1600 x 1200 пикселов, в предположении, что размеры обоих экранов одинаковы.

Относительные единицы измерения позволяют браузеру подстраиваться под конкретные настройки экрана посетителя. Если вместо относительных единиц используются абсолютные единицы измерения, такие, как пункты, пайки или дюймы, вид Web-страницы полностью находится во власти браузера, который определяет, насколько большим должен быть дюйм на экране. Нет ничего удивительного в том,

что в прежних браузерах поддержка параметров оборудования не была такой хорошей, как сейчас, и посетители могли видеть страницы с неразборчивым мелким текстом, который невозможно было увеличить.

Использование констант длины в каскадных таблицах стилей

Если задание размеров шрифта с помощью относительных или абсолютных единиц не привлекает вас из-за возможных проблем, можно подставить одну из семи констант длины в каскадные таблицы стилей. Вы догадываетесь: эти константы приблизительно эквивалентны семи размерам шрифта в HTML, показанным на рисунке 48.3. Таким образом, вы снова вовлечены в «игру» с размерами, но в этом способе вы не оставляете браузеру шансов отобразить Web-страницу с неразборчивым текстом.

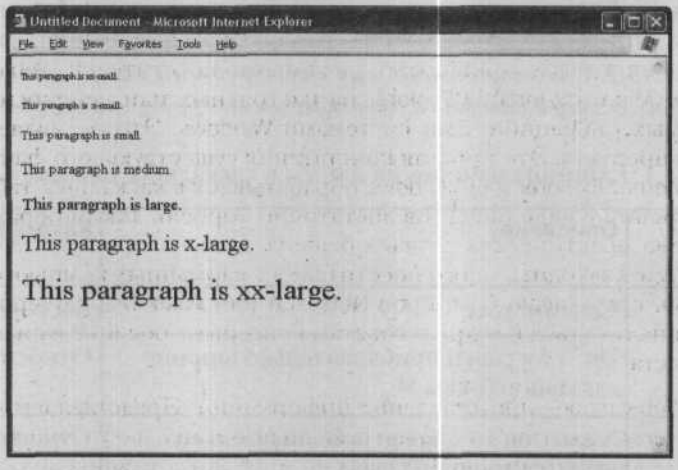


Рис. 48.3. В случае сомнений каскадные таблицы стилей предоставляют семь констант длины, которые приблизительно соответствуют семи размерам шрифта в HTML

Семь констант длины, используемые в каскадных таблицах стилей, это: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large` и `xx-large`. Одно из этих ключевых слов помещается в определение стиля вместо значения размера и единиц измерения, подобно следующему примеру:

```
p {
    font-size: x-large;
}
```

Можно также задавать относительную константу длины для настройки величины шрифта относительно его текущего размера, каким бы он ни был. Используйте ключевое слово `larger`, чтобы увеличить размер шрифта на один размер, или воспользуйтесь ключевым словом `smaller`, чтобы сделать шрифт на один размер меньше:

```
<p style="font-size: x-large;">
This type size is x-large, but <span style="font-size:
smaller;">this type size is one size smaller.</span>
</p>
```

Дневник: способ изменения размера шрифта

Разные дизайнеры подходят к проблеме задания размера шрифта в каскадных таблицах стилей по-разному. Некоторые говорят: «Никогда не изменяйте размер шрифта, и точка». Другие советуют придерживаться констант длины. Третьи утверждают, что изменение размера шрифта не вызывает проблем, если пользоваться только относительными единицами измерения.

Основываясь на подавляющем числе пользователей, бороздящих Интернет с помощью более или менее современных браузеров, по-видимому, следует считать, что задание размера шрифта с помощью относительных единиц измерения является достаточно надежной технологией для Web-сайтов общего назначения. Это заключение основывается на предположении, что большинство посетителей Web-сайтов пользуется браузером Microsoft Internet Explorer на настольных или переносных компьютерах, управляемых операционными системами Windows. Это не является рекламой программного продукта. Это простая констатация существующего факта. Последняя пара версий Internet Explorer для Windows обрабатывает в каскадных таблицах стилей единицы измерения в виде пикселов достаточно хорошо. Таким образом, проблема для большинства практических случаев решена. Однако если вы ожидаете, что на ваш Web-сайт будет заходить много посетителей с карманных компьютеров, компьютеров Macintosh, с помощью браузеров Netscape или Windows-браузеров старых версий, следует прислушаться к мудрым советам консерваторов и не варьировать размеры шрифта текста.

Главная цель Web-сайта – представление информации. Представляемое содержимое должно быть легко доступно и удобно для пользования, поэтому оно обязательно должно быть удобочитаемым. Любые аспекты эстетики и дизайна должны всегда подчиняться требованиям практичности, несмотря на возможные противоречия с внутренним эстетическим чувством Web-дизайнера.

Конечно, связь между эффективным дизайном и удобством пользования является нетривиальной. Эта тема здесь не обсуждается. Однако одно дело стремиться к эффективному и интуитивно понятному графическому макету, и другое дело – вдаваться в тонкости шрифтового оформления Web-страниц. Таким образом, если изменение размеров шрифта помогает сделать Web-сайт более удобным для большей части вашей аудитории, то этим стоит заняться.

ГЛАВА 49.
Управление интервалами

В HTML величина зазора между элементами в достаточно большой степени определяется браузером. В браузер встроены механизмы, позволяющие представлять, как выглядит текст, размеченный каждым тегом. Конечно, можно сделать принудительный обрыв строки с помощью тега <br>, но это грубый и неточный метод. Художники-дизайнеры понимают, что пустое, или отрицательное, пространство также важны, как занятое, или положительное, пространство. Пробелы являются крайне важным элементом дизайна, и они требуют управления с точностью до пиксела. Не требуется разрабатывать много Web-страниц, чтобы начать желать лучших параметров для управления зазорами, чем те, что предлагаются в HTML.

Хорошо, что есть каскадные таблицы стилей. Они предоставляют все виды управления зазорами: пространство вокруг элементов, пространство внутри элементов, интервалы между строками текста, зазоры между словами и интервалы между буквами. В главе 26 обсуждается вопрос задания полей страницы с помощью каскадных таблиц стилей. В этой главе рассматриваются другие виды зазоров, регулируемых с помощью каскадных таблиц стилей, в особенности те из них, которые относятся к тексту.

Управление пространством вокруг элементов

Атрибуты полей в каскадных таблицах стилей применяются не только к границам страницы. Поля можно применять к любому элементу, тем самым регулируя свободное пространство вокруг элемента, например, так, как показано на рисунке 49.1.

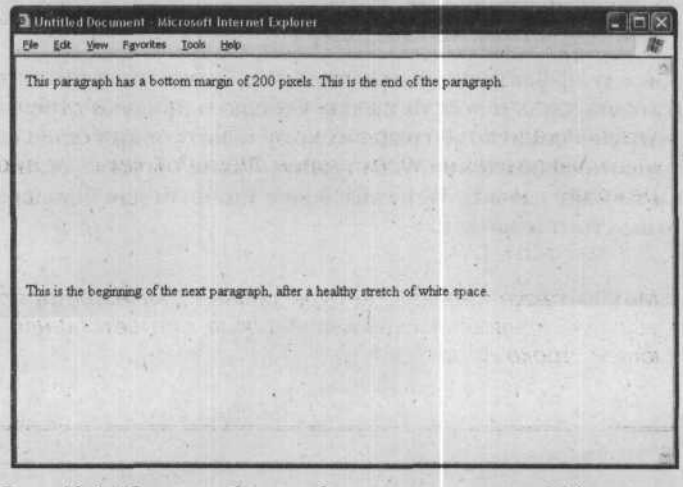


Рис. 49.1. У этого абзаца задано нижнее поле в 200 пикселов, которое создает приятное для глаз пустое пространство для отделения абзаца от следующего за ним текста

Листинг 49.1. Исходный код страницы, вид которой в браузере показан на рисунке 49.1

```
<p style="margin-bottom: 200 px;">This paragraph has a bottom margin of 200 pixels. This is the end of the paragraph.</p>

<p>This is the beginning of the next paragraph, after a healthy stretch of white space.</p>
```

Совет

При добавлении пустого пространства с помощью полей к нескольким элементам обычно рекомендуется выбирать верхнее или нижнее поле, а также левое или правое поле, и последовательно придерживаться сделанного выбора для каждого правила стилей элементов. Другими словами, не задавайте в одном элементе верхнее поле, а в другом элементе нижнее поле. Если вы будете придерживаться такого правила, у вас не возникнут проблемы, скажем, когда 200-пиксельное нижнее поле стыкуется с 200-пиксельным верхним полем, образуя зазор в 400 пикселов.

Существует четыре атрибута полей: `margin-top`, `margin-bottom`, `margin-left` и `margin-right`. Атрибуты могут иметь разные значения и единицы измерения (таблицу единиц измерения, действующих в каскадных таблицах стилей, см. в главе 48). Как видно из кода для Web-страницы, показанной на рисунке 49.1, можно не задавать все атрибуты полей. Указывайте лишь поля, которые необходимы для создания требуемых отступов.

Совет

Для быстроты можно использовать групповой атрибут `margin`. При указании единственного значения для этого атрибута создаются равные поля со всех четырех сторон элемента, как в следующем примере кода:

```
margin: 200px;
```

Можно также задавать четыре разные значения, представляющие верхнее, правое, нижнее и левое поля, соответственно, как в следующей строке кода:

```
margin: 100 50 200 25px;
```

Управление пространством внутри элементов

Атрибуты набивки определяют величину пробела между границами элемента и началом содержимого элемента. Чтобы получить более полное представление о том, как работает набивка, представьте, что каждый элемент на экране имеет свой собственный невидимый прямоугольник, как те, что показаны на рисунке 49.2. С помощью набивки добавляется пробел между краем прямоугольника и содержимым внутри его.

Как и в случае полей, существует четыре атрибута набивки: `padding-top`, `padding-bottom`, `padding-left` и `padding-right`. Каждый из атрибутов можно задавать отдельно, а можно не указывать в определении стиля те из них, которые вам не нужны:

```
p {
    padding-top: 20px;
    padding-left: 10px;
}
```

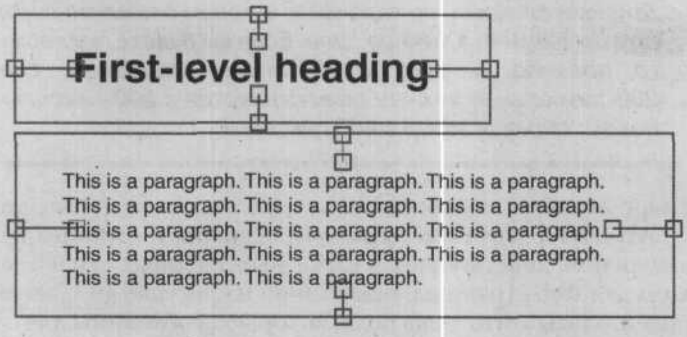


Рис. 49.2. Набивка определяет величину пробела между внешней границей элемента и содержимым внутри границ

Совет

Как и в случае полей, в каскадных таблицах стилей предоставляется групповой атрибут `padding`. Он работает аналогичным образом: указывайте одиночное значение, чтобы добавить равную набивку со всех четырех сторон, или задавайте четыре разные значения для набивки сверху, справа, снизу и слева:

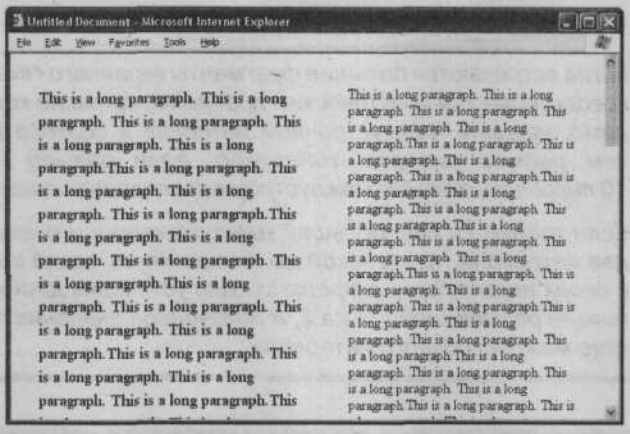
```
padding: 20px;
padding: 10 12 18 8px;
```

Управление междустрочными интервалами

Используйте атрибут `line-height` для управления величиной зазора между строками текста. Этот прием удобен для улучшения удобочитаемости длинных фрагментов текста, как, например, на рисунке 49.3.

Для атрибута `line-height` указывается значение и единицы измерения, как в следующем примере:

```
p {
    font-size: 18px;
    line-height: 27px;
}
```


Untitled Document - Microsoft Internet Explorer
File Edit View Favorites Tools Help
This is a long paragraph. This is a long paragraph. This is a long paragraph.

```
word-spacing: -3px;
```

удаляет три пиксела из зазора, что приводит к сжатию строк текста. Чтобы установить зазор между словами равным значению, используемому в браузере по умолчанию, введите следующий код в определение стиля:

```
word-spacing: normal;
```

Совет

Когда встречаются большие фрагменты экранного текста, следует руководствоваться практическим правилом, согласно которому необходимо задавать междустрочный интервал в полтора раза большим, чем размер шрифта. Например, если размер шрифта равен 10 пикселам, задайте междустрочный интервал равным 15 пикселам.

Если требуется еще больший междустрочный интервал, попробуйте два интервала, точно такой же интерлиньяж, какой вы использовали в своих институтских рефератах. Для установки двух интервалов умножьте размер шрифта на 2, и полученное значение задайте в качестве междустрочного интервала.

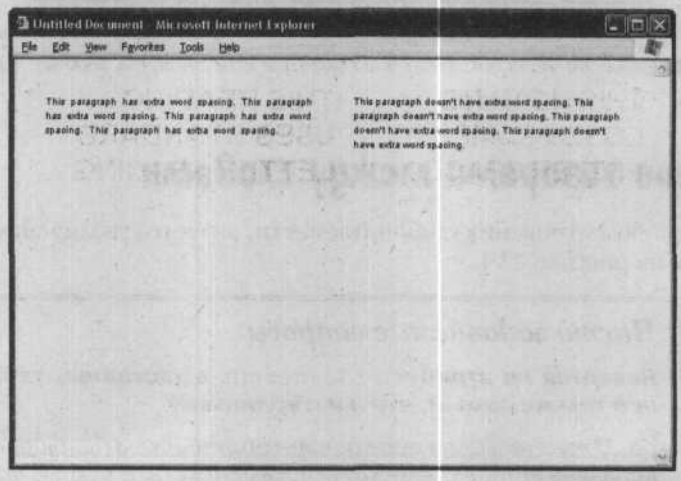


Рис. 49.4. Всего три пиксела дополнительного зазора между словами делают левый абзац более удобочитаемым

Управление зазором между буквами

Наконец, каскадные таблицы стилей позволяют регулировать величину зазора между буквами в HTML-элементе с помощью атрибута `letter-spacing`. Пример изменения зазоров между буквами показан на рисунке 49.5. Как и в случае атрибута `word-spacing`, значение атрибута `letter-spacing` добавляется к зазору между буквами, используемому браузером по умолчанию. Атрибут может принимать и отрицательные значения, при которых используемый по умолчанию зазор между буквами уменьшается на задаваемую величину.

Совет

*В общем случае для шрифта больших размеров требуются меньшие зазоры между словами, тогда как для шрифтов малого размера необходимы большие зазоры. Кроме того, широкие шрифты, например, **Verdana**, в большинстве случаев выигрывают от небольших дополнительных зазоров между словами, тогда как сжатые шрифты смотрятся лучше со слегка уменьшенными зазорами.*

В определении стиля атрибут `letter-spacing` выглядит подобно следующему примеру:

```
letter-spacing: 1px;
```

или, чтобы уменьшить зазор между буквами, можно воспользоваться следующим кодом:

```
letter-spacing: -1px;
```

или для установки зазоров между буквами, используемых браузером по умолчанию, вставьте следующую строку:

```
letter-spacing: normal;
```

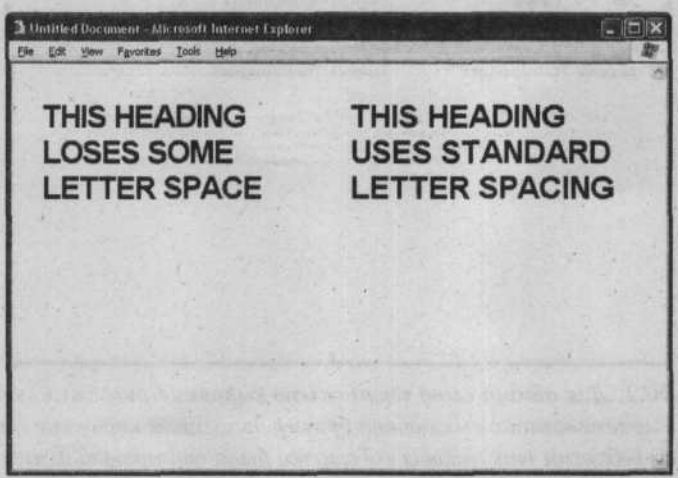


Рис. 49.5. Применяйте атрибут `letter-spacing`, чтобы увеличить (или уменьшить) в элементе зазор между буквами, используемый браузером по умолчанию

Совет

Увеличенные зазоры между буквами, как правило, улучшают удобочитаемость мелкого текста, тогда как для крупного шрифта лучше подходят уменьшенные зазоры между буквами. Кроме того, гарнитуры с широкими буквами лучше смотрятся с увеличенными зазорами, тогда как для гарнитур с узкими буквами четкость текста выиграет от уменьшения зазоров между буквами.

ГЛАВА 50.
Управление выравниванием текста

Выключенные строки текста имеют одинаковую длину. Для создания такого эффекта в более короткие строки браузером добавляются дополнительные пробелы, как показано в примере абзаца на рисунке 50.1.

Определение

Выключенные *строки текста имеют одинаковую длину.*

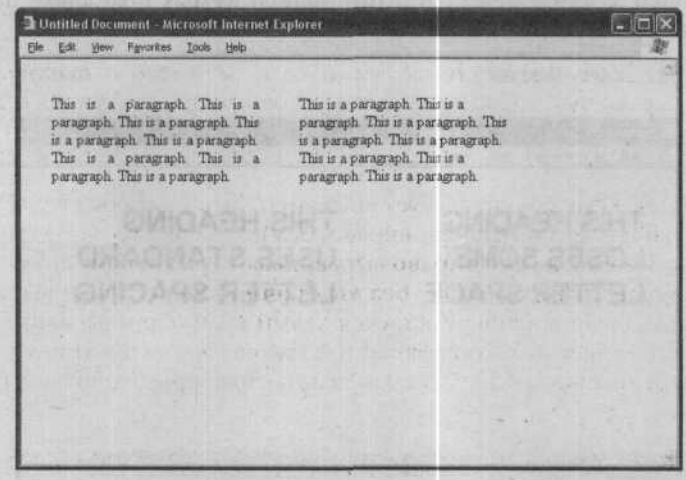


Рис. 50.1. Для абзаца слева установлено выравнивание с выключкой. При выравнивании с выключкой браузер дополняет короткие строки с пробелами так, чтобы все строки были одинаковой длины

Чтобы задать в HTML выключенный текст, в теге выравниваемого абзаца для атрибута `align` следует указать значение `justify`:

```
<p align="justify">Этот абзац содержит текст с выключкой.</p>
```

Однако каскадные таблицы стилей делают этот метод устаревшим. Вместо этого включите в свое определение стиля CSS-атрибут `text-align`:

```
<p style="text-align: justify;"> Этот абзац содержит текст с выключкой.</p>
```

Совет в копилку опыта. Встроенный в браузер механизм расчетов, используемый для выравнивания текста, не является очень сложным, в результате блоки текста с выключкой часто выглядят на Web-страницах напыщенными в худшем смысле этого

слова и по своему виду уступают даже самой отсталой провинциальной газете. Часто, когда все сделано правильно, профессионально и точно, выключка придает стильный вид книгам и другим печатным материалам. Однако текст без выключки, как правило, легче читается на экране монитора, а удобочитаемость, как известно, по важности превалирует над всеми остальными свойствами экранного шрифта.

Определение

По визуальному восприятию противоположным для текста с выключкой является текст ***без выключки****, который в большинстве случаев лучше смотрится на экранах мониторов, чем блоки текста с одинаковой длиной (с выключкой).*

Определение

Текст ***без выключки вправо*** *(ragged right) выравнивается по левому краю, а правый край остается невыровненным. Текст* ***без выключки влево*** *(ragged left) выравнивается по правому краю, при этом невыровненным остается левый край. В тексте* ***с выравниванием по центру*** *(ragged center) строки центрируются, а края строк остаются невыровненными.*

Говоря о тексте без выключки, следует упомянуть, что существует три вида выравнивания без выключки: без выключки вправо, без выключки влево и выравнивание по центру. Абзацы **без выключки вправо** выравниваются по левому краю, а правый край остается невыровненным; в абзацах **без выключки влево** строки выравниваются по правому краю, а невыровненным остается левый край. При выравнивании строк по центру оба края абзацев выглядят рваными, а строки текста центрируются так, как показано в примере на рисунке 50.2. По умолчанию браузеры отображают текст без выключки вправо.

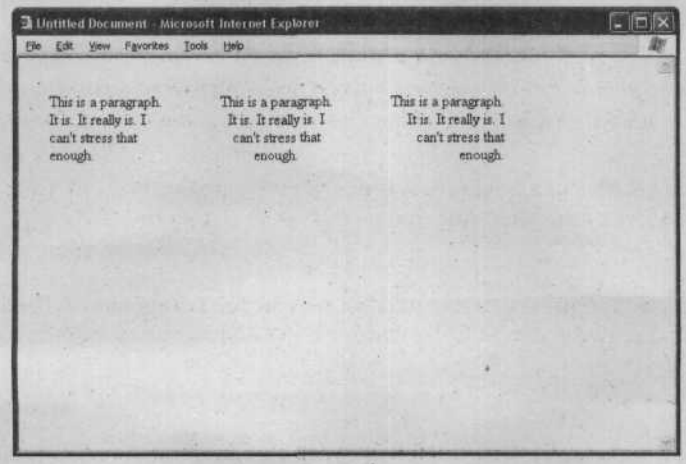


Рис. 50.2. Слева показан абзац без выключки вправо, по центру располагается абзац текста с выравниванием строк по центру, и справа находится абзац без выключки влево

При задании выравнивания текста без выключки фактически необходимо действовать прямо противоположно логике, а именно, правильное значение для атрибута `align` в HTML или значение атрибута `text-align` в каскадных таблицах стилей должно быть противоположным тому, что следует получить на самом деле. Для текста без выключки вправо правильным значением будет `left` (влево), а для текста без выключки влево правильное значение – `right` (вправо):

```
<p align="left">Этот абзац выровнен справа.</p>


<p style="text-align: right;">Этот абзац выровнен слева.</p>
```

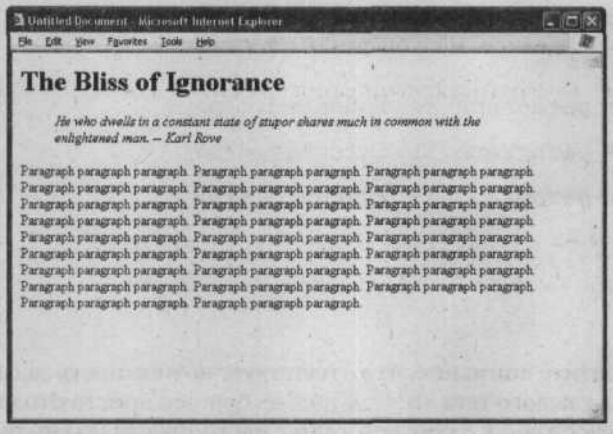
Если у вас возникает путаница в связи с таким противоречием между задаваемым типом выравнивания и присваиваемым значением, попробуйте представить это с точки зрения стороны, по которой выравнивается текст, возможно именно так размышляли разработчики HTML и каскадных таблиц стилей. В абзацах без выключки вправо текст выравнивается по левой стороне, поэтому с этой точки зрения запись `text-align: left;` выглядит вполне логичной.

Эти рассуждения не касаются выравнивания по центру, для которого атрибуты принимают значение `center`:

```
<p align="center">Этот абзац выровнен по центру.</p>


<p style="text-align: center;">Этот абзац выровнен по центру.</p>
```

ГЛАВА 51.
Управление отступами

Стандартным решением при создании отступа с помощью средств HTML является использование тега `<blockquote>`, применение которого противоречит требованию соблюдения стандартов. Строго говоря, назначение тега `<blockquote>` – это смещение фрагмента экранного текста подобно цитате в начале школьного сочинения. Пример такой цитаты показан на рисунке 51.1. Однако желательно найти более практичный, ориентированный на результат метод, чем тот, который используют Web-дизайнеры, решая свои задачи в условиях дефицита времени. Озабоченные быстротой разработки, они прежде применяли тег `<blockquote>` для быстрого создания отступов, пример которых показан на рисунке 51.2.


Untitled Document - Microsoft Internet Explorer
File Edit View Favorites Tools Help
The Bliss of Ignorance
He who dwells in a constant state of stupor shares much in common with the enlightened man. – Karl Rove
Paragraph paragraph paragraph. Paragraph paragraph paragraph. Paragraph paragraph paragraph.

В обоих случаях HTML-код выглядит одинаково. Просто разместите теги <blockquote> вокруг содержимого, которое необходимо сдвинуть так, как показано в следующем примере кода:

```
<blockquote>
   <p>This paragraph is indented.</p>
   <p>This paragraph is indented.</p>
   <p>This paragraph is indented.</p>
</blockquote>
```

Можно также применять вложение тегов <blockquote>, чтобы увеличить величину отступа:

```
<blockquote>
  <blockquote>
    <blockquote>
      <p>This paragraph is indented.</p>
      <p>This paragraph is indented.</p>
      <p>This paragraph is indented.</p>
    </blockquote>
  </blockquote>
</blockquote>
```

Тем не менее, обратите внимание, что отсутствует возможность задания точной величины отступа. Для каждого тега <blockquote> браузер просто отодвигает текст вправо на некоторое расстояние в соответствии с настройками по умолчанию.

В целом, лучшим решением задачи создания отступа является использование каскадных таблиц стилей. С помощью атрибута margin-left обеспечивается изящное решение проблемы, и такой способ предоставляет дополнительную гибкость в настройке точного значения отступа, пример которого показан на рисунке 51.3.

Листинг 51.1. Исходный код страницы, вид которой в браузере показан на рисунке 51.3

```
<head>
  <style type="text/css">

    p {
      margin-left: 50px;
    }

  </style>
</head>
```

```
<body>
  <h1>Blah blah headline</h1>

  <p>Blah blah blah....</p>

  <p>Blah blah blah....</p>
</body>
```

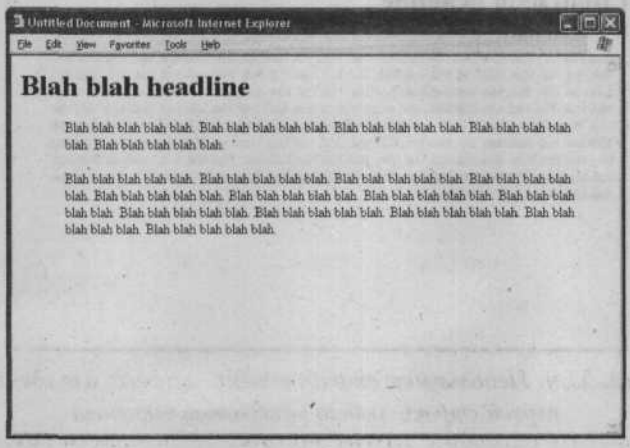


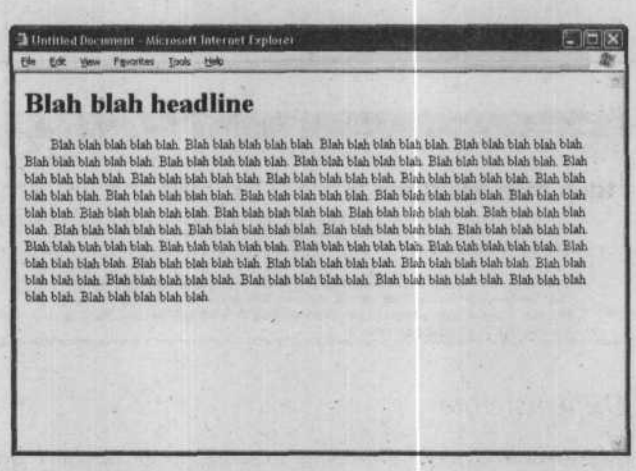
Рис. 51.3. Отступы лучше создавать с помощью средств каскадных таблиц стилей. Такой метод позволяет задавать точное значение отступа

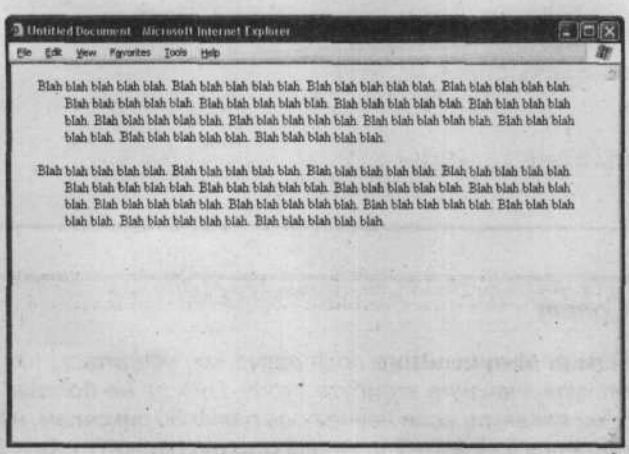
Каскадные таблицы стилей справляются лучше с еще одной задачей. Они позволяют задавать дополнительный отступ для первой строки любого текстового элемента, будь то абзац, заголовок или что-либо еще. Для этого применяется атрибут `text-indent`, действие которого показано на рисунке 51.4.

Листинг 51.2. Исходный код страницы, вид которой в браузере показан на рисунке 51.4

```
<head>
  <style type="text/css">

    p {
      text-indent: 30px;
    }

  </style>
</head>
```

```
<body>
  <h1>Blah blah headline</h1>

  <p>Blah blah blah....</p>
</body>
```


Untitled Document - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Blah blah headline
Blah blah blah blah blah. Blah blah blah blah blah. Blah blah blah blah blah.


Untitled Document - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Blah blah blah blah blah. Blah blah blah blah blah. Blah blah blah blah blah.

```
<body>
  <p>Blah blah blah....</p>

  <p>Blah blah blah....</p>
</body>
```

Совет

Если величина левого поля равна «x», убедитесь, что для обратного отступа значение атрибута `text-indent` *не больше, чем «-x». Другими словами, если левое поле равно 50 пикселам, не задавайте -80 пикселов в качестве значения отступа первой строки, иначе она выйдет за левую границу страницы.*

ГЛАВА 52.
Создание списков

В HTML существует два типа списков: нумерованные (упорядоченные) списки и ненумерованные (маркированные) списки. Разметка списка начинается с тега <ol> или <ul> для нумерованного и ненумерованного списка, соответственно. Далее следуют теги <li> для каждого элемента списка. Примеров обоих типов списков см. на рисунке 52.1.

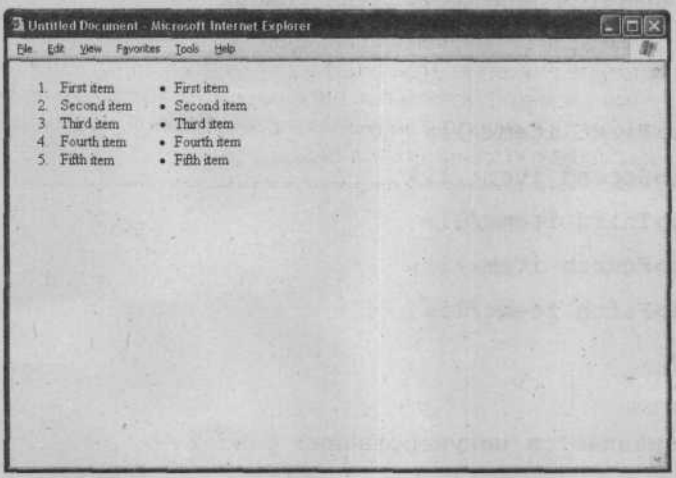


Рис. 52.1. В HTML имеются нумерованные (упорядоченные) и ненумерованные (маркированные) списки. Разметка списков одинакова за исключением тега списка

Листинг 52.1. Исходный код страницы, вид которой в браузере показан на рисунке 52.1

```
<table>
  <tr>
    <td>

<!-Здесь начинается нумерованный список ->

      <ol>
        <li>First item</li>
        <li>Second item</li>
        <li>Third item</li>
        <li>Fourth item</li>
        <li>Fifth item</li>
```

```
      </ol>

<!-Здесь оканчивается нумерованный список ->

    </td>
    <td>

<!-Здесь начинается ненумерованный список ->

      <ul>
        <li>First item</li>
        <li>Second item</li>
        <li>Third item</li>
        <li>Fourth item</li>
        <li>Fifth item</li>
      </ul>

<!-Здесь оканчивается ненумерованный список ->

    </td>
  </tr>
</table>
```

Как можно заметить, HTML-код для списков полностью одинаков, за исключением тега списка. Следовательно, чтобы изменить нумерованный на ненумерованный список, достаточно поменять тег `<ul>` на тег `<ol>`.

Обратите также внимание, что в нумерованном списке не надо набирать номера элементов. Браузер самостоятельно отслеживает нумерацию и отображает номера элементов списка автоматически.

В отличие от элементов HTML, описанных в других главах данного раздела, для списков имеется ряд полезных атрибутов, управляющих внешним видом списков. Эти атрибуты обобщены в таблице 52.1.

В нумерованных списках, отличных от стандартных десятичных нумерованных списков, атрибут `start` указывает n-е значение в последовательности. Таким образом, алфавитный список со значением 4 для атрибута `start` начинается с буквы «D», поскольку «D» – четвертая буква алфавита. Подобным образом, список с римскими номерами начинается с «X», если значение атрибута `start` равно 10.

Табл. 52.1. Атрибуты списков в HTML

Атрибут	Тег, к которому применяется	Управляемый элемент	Возможные значения	Примеры
type	ol	Начальный символ	A (заглавный буквенный), a (строчной буквенный), I (заглавный римский числовой), i (строчный римский числовой), 1 (десятичный)	<ol type="A"> <ol type="i">
type	ul	Форма буллита (маркера)	circle (пустой кружок), disc (сплошной кружок), square (квадрат)	<ul type="circle"> <ul type="square">
start	ol	Начальное число или первая буква в списке	Любое числовое значение	<ol start="4">

Вложенные списки в HTML

Вложенные списки – это списки, которые находятся внутри других списков. В HTML-разметке вложенный список занимает место элемента списка, как в следующем примере:

```
<ol>
  <li>First item of the main list</li>
  <li>Second item of the main list</li>
  <li>Third item of the main list</li>

<!– Здесь начинается вложенный список –>

  <ol>
    <li>First item of the nested list</li>
    <li>Second item of the nested list</li>
    <li>Third item of the nested list</li>
  </ol>

<!– Здесь окончился вложенный список –>

  <li>Fourth item of the main list</li>
```

```
  <li>Fifth item of the main list</li>
</ol>
```

При необходимости можно вложить вложенный список в список, который находится внутри другого списка:

```
<ul>
  <li>First item of the main list</li>
  <li>Second item of the main list</li>
  <li>Third item of the main list</li>

<!- Здесь начинается вложенный список ->

  <ul>
    <li>First item of the nested list</li>
    <li>Second item of the nested list</li>

<!- Здесь начинается вложенный список, расположенный внутри другого вложенного списка ->

    <ul>
      <li>First item of the very nested list</li>
      <li>Second item of the very nested list</li>
    </ul>

<!- Здесь окончился вложенный список, расположенный внутри другого вложенного списка ->

    <li>Third item of the nested list</li>
  </ul>
<!- Здесь окончился вложенный список ->
  <li>Fourth item of the main list</li>
  <li>Fifth item of the main list</li>
</ul>
```

Определение

Вложенные списки *(nested lists) — это списки, которые находятся внутри других списков.*

При вложении ненумерованных списков браузер обычно изменяет циклически тип маркера для каждого последующего вложенного списка. Таким образом, если список начинается с маркера в виде сплошного кружка (`disc`), во вложенном списке используется маркер в виде пустого кружка (`circle`), во вложенном списке второго уровня используется маркер в виде квадрата (`square`), а во вложенном списке третьего уровня используется опять маркер в виде сплошного кружка (`disc`) и так далее. Пример вложенных списков см. на рисунке 52.2.

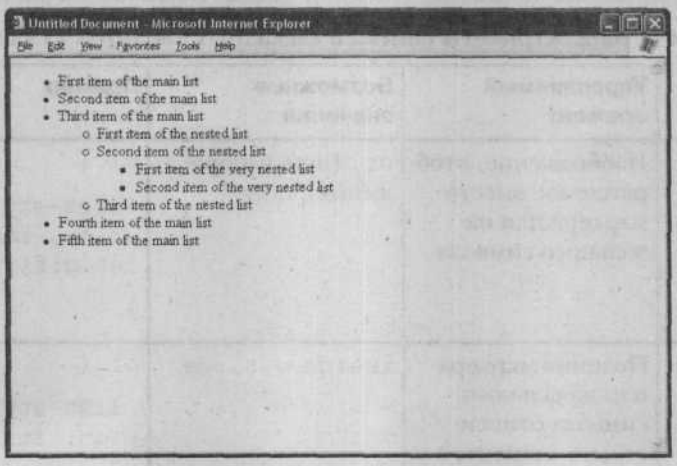


Рис. 52.2 В ненумерованных списках браузер обычно циклически изменяет тип маркера для каждого последующего вложенного списка

Совет

Нумерованные списки не подвергаются подобной обработке. Чтобы создать эффект структуры с нумерованными списками, необходимо в явном виде задать атрибуты шрифта для каждого вложенного списка — или задействовать таблицу стилей. Подробности см. в разделе «Инструментарий» в конце данной главы.

Настройка свойств списка с помощью каскадных таблиц стилей

Каскадные таблицы стилей вводят несколько дополнительных возможностей форматирования списков, которые приводятся в таблице 52.2.

Замена маркеров изображением

Возможно, наиболее интересным из этих атрибутов является атрибут `list-style-image`, который позволяет задавать файл изображения вместо стандартного маркера в виде сплошного кружка, пустого кружка или квадрата. Пример списка, где маркер заменен изображением, см. на рисунке 52.3.

Изменение положения начального символа

Атрибутом list-style-position определяется, где будут отображаться маркеры или начальные символы относительно элементов списка. При задании для этого атрибута значения outside начальные символы отображаются вне основного блока элементов списка, а при установке для атрибута list-style-position значения inside начальные символы находятся внутри основного блока элементов списка. Примеры расположения начальных символов списка показаны на рисунке 52.4.

Табл. 52.2. Атрибуты списка в каскадных таблицах стилей

Атрибут	Управляемый элемент	Возможные значения	Пример

list-style-image	Изображение, отображаемое вместо маркера или начального символа	url (путь изображения), none	ul {
			list-style-image: url(../images/bullet.gif);
			}

list-style-position	Позиция маркера или начального символа относительно элементов списка	inside, outside	ol {
			list-style-position: inside;
			}

list-style-type	Тип отображаемого маркера или начального символа	disc, circle, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha, none	ol {
			list-style-type: lower-roman;
			}

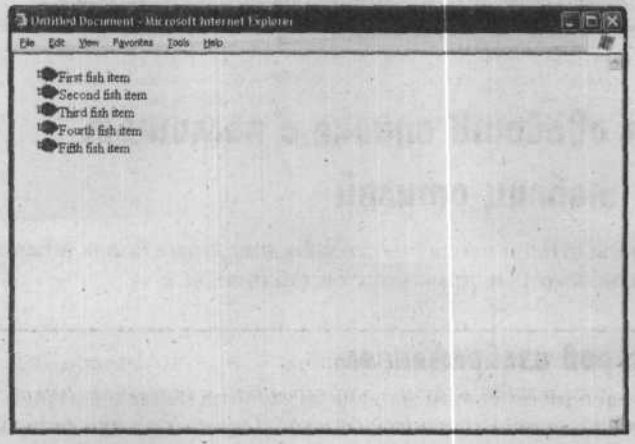


Рис. 52.3. Используйте атрибут list-style-image для замены стандартных маркеров изображением из файла

Листинг 52.2. Исходный код страницы, вид которой в браузере показан на рисунке 52.3

```
<head>
  <style type="text/css">

    ul {
      list-style-image: url(images/fish.gif);
    }

  </style>
</head>

<body>
  <ul>
    <li>First fish item</li>
    <li>Second fish item</li>
    <li>Third fish item</li>
    <li>Fourth fish item</li>
    <li>Fifth fish item</li>
  </ul>
</body>
```

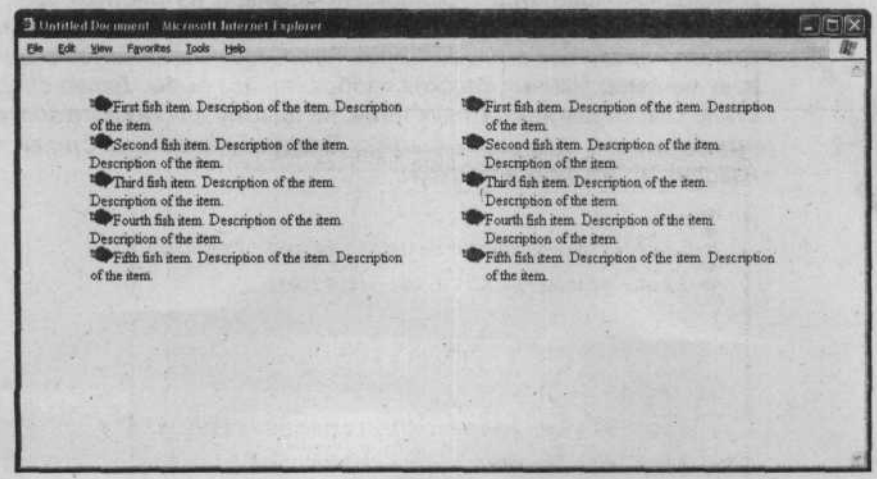


Рис. 52.4. Используйте атрибут `list-style-position` *для задания положения маркеров или начальных символов относительно элементов списка. Для списка слева значение этого атрибута равно* `inside`*, а для списка справа –* `outside`

Листинг 52.3. Исходный код страницы, вид которой в браузере показан на рисунке 52.4

```
<style type="text/css">

/* Сначала изменим маркер всех тегов <ul> независимо от класса. */

  ul {
    list-style-image: url(images/fish.gif);
  }

/* Теперь определим классы для значений атрибута list-style-position. */

  ul.in {
    list-style-position: inside;
  }

  ul.out {
    list-style-position: outside;
  }

</style>
```

Совет

В исходном коде Web-страницы, показанной на рисунке 52.4, используется удобный прием с применением каскадных таблиц стилей. Сначала правилом стиля для всех тегов `<ul>` *заменяются маркеры всех ненумерованных списков изображением рыбы. Далее следуют стили класса для конкретных тегов, по одному для каждого значения атрибута* `list-style-position`*. Следующая таблица стилей производит тот же самый эффект:*

```
ul.in {
  list-style-image: url(images/fish.gif);
  list-style-position: inside;
}

ul.out {
  list-style-image: url(images/fish.gif);
  list-style-position: outside;
}
```

Преимущество кода таблицы стилей, приведенного в листинге 52.3, состоит в том, что атрибут `list-style-image` *не повторяется два раза. Вместо этого он приводится однажды, в глобальном переопределении тега. А два класса просто уточняют данный общий стиль.*

Переопределение типа списка, используемого браузером по умолчанию

Атрибут `list-style-type` удобен для переопределения HTML-тегов `<ol>` и `<ul>`, чтобы не надо было хранить параметр отдельных типов атрибутов для каждого списка на Web-странице:

```
ul {
  list-style-type: square;
}
```

На Web-странице с данным правилом стиля автоматически все ненумерованные списки форматируются с маркерами в виде квадратиков.

Совет

Чтобы создать «голый» список без маркеров и начальных символов любого типа, задайте для атрибута `list-style-type` значение `none` в определении стиля, относящегося к этому списку.

Инструментарий

Создание для нумерованных списков стиля автоматической структуры

При вложении нумерованных списков браузер не изменяет циклически типы списков для создания эффекта структуры, как в случае ненумерованных списков.

Простая таблица стилей устраняет этот недостаток (см. рис. 52.5). Поместите данную таблицу стилей между тегами `<style>` в разделе заголовка HTML-страницы или скопируйте таблицу стилей точно в том виде, в котором она представлена, во внешний CSS-файл.

Часто задаваемые вопросы

Сколько уровней может добавляться в структуру?

Эта таблица стилей работает со структурами, включающими до пяти уровней. Если в структуре больше пяти уровней, браузер возвращается к обычной нумерации списков. С данным эффектом возврата к обычной нумерации можно бороться, добавляя правила стилей для требуемого количества уровней структуры, следуя при этом образцу, который составляют существующие правила стилей.

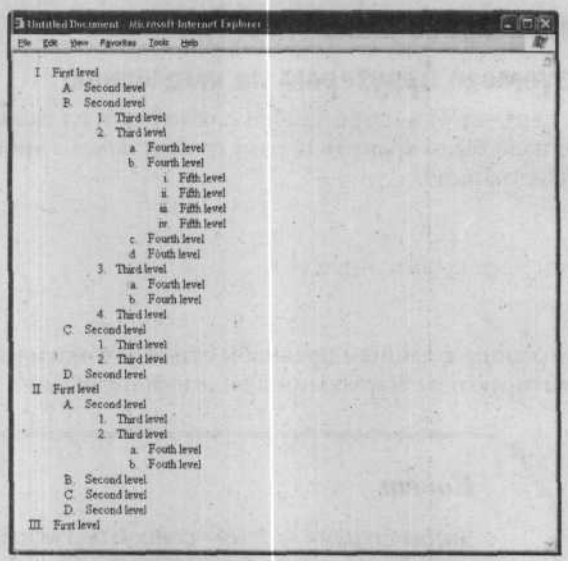


Рис. 52.5. Данная таблица стиля структуры служит для автоматического отображения вложенных нумерованных списков в формате структуры

```
/* Пронумеруем первый уровень структуры с помощью заг-
лавных римских числительных (I, II, III, и т.д.). */

ol {
  list-style-type: upper-roman;
}

/* Второй уровень структуры является вложенным
списком или тегом <ol>, расположенным где-нибудь
внутри другого тега <ol>. Эта задача решается с по-
мощью контекстно-зависимого селектора. */

ol ol {

/* Пронумеруем второй уровень структуры с помощью
заглавных букв (A, B, C, и т. д.). */

  list-style-type: upper-alpha;
}
/* Третий уровень структуры является вложенным спис-
ком внутри вложенного списка или тега <ol>, распо-
ложенного где-нибудь внутри другого тега <ol>, ко-
торый находится внутри еще одного тега <ol>. Необ-
ходим еще один контекстно-зависимый селектор. */
```

```
ol ol ol {

/* Пронумеруем третий уровень структуры с помощью
десятичных чисел (1, 2, 3, и т. д.). */

  list-style-type: decimal;
}

/* Четвертый уровень структуры является вложенным
списком внутри вложенного списка внутри вложенного
списка, или, иными словами глубина структуры со-
ставляет четыре тега <ol>. */
ol ol ol ol {

/* Пронумеруем четвертый уровень структуры с по-
мощью строчных букв (a, b, c, и т. д.). */

  list-style-type: lower-alpha;
}

/* Пятый уровень структуры является вложенным спис-
ком внутри вложенного списка внутри вложенного
списка внутри вложенного списка, или иными словами
глубина структуры составляет пять тегов <ol>. */

ol ol ol ol ol {

/* Пронумеруем пятый уровень структуры с помощью
строчных римских числительных (i, ii, iii, и
т. д.). */

  list-style-type: lower-roman;
}
```

Совет

Если возникают проблемы, воспользуйтесь каскадными таблицами стилей, чтобы отрегулировать зазоры для элементов <li> и поля для тега <ol> с целью создания более привлекательного вида Web-страницы.

ГЛАВА 53.

Выделение текста с помощью цвета

В главе 5 вы видели, как задавать цвет по умолчанию для текста Web-страницы вместе с цветом фона и цветами ссылок, используемыми по умолчанию. В этой главе объясняется, как изменять цвет определенных текстовых элементов или фрагментов текста внутри элемента, не воздействуя на цвет текста, используемый по умолчанию.

Если вы принадлежите к числу твердолобых Web-дизайнеров, настаивающих на использовании тега <font>, вы будете рады узнать, что свойство цвета для этого тега позволяет выполнить стоящую перед нами задачу:

```
<font face="Arial, Helvetica, sans-serif" color="#FF0000">
  <h3>Примечание</h3>
  <p>Вы <strong>должны</strong> предоставить ваш текущий адрес электронной почты.</p>
</font>
```

Также как гарнитура тега <font> применяется ко всем текстовым элементам, вложенным внутри него, так и цвет шрифта применяется к этим же элементам. В предыдущем примере оба элемента, выделенных тегами <h3> и тегом абзаца, окрашиваются в красный цвет (#FF0000), независимо от цвета текста, используемого по умолчанию на этой Web-странице.

Того же самого эффекта можно достичь с помощью атрибута color каскадных таблиц стилей:

```
<h3 style="color="#FF0000;"> Примечание</h3>
<p style="color="#FF0000;"> Вы <strong>должны</strong> предоставить ваш текущий адрес электронной почты.</p>
```

Совет

Крайне осторожно изменяйте цвет текста там, где это приходится делать. Одним из основных идентификаторов гиперссылок является цвет, отличный от цвета остального текста. При изменении цвета текста, не являющегося гиперссылкой, определенный процент посетителей вашего Web-сайта будет щелкать на этом тексте, полагая, что это гиперссылка. Если вы еще и подчеркнете текст, цвет которого изменили, то рискуете получить рассерженные электронные сообщения о том, что ваш Web-сайт не работает.

Прибегайте к изменению цвета текста как к последнему средству после того, как испробованы остальные возможности, другие методы шрифтового выделения, такие, как полужирный шрифт или курсив.

Если требуется изменить цвет текстовой строки внутри элемента, вложите теги <font> внутрь элемента:

```
<p>Розы - <font color="#FF0000">красные</font>, а фиалки - <font
color="#0000FF">синие</font>.</p>
```

Часто задаваемые вопросы

Что в действительности окрашивается, когда в текстовый элемент добавляется цвет фона?

При задании для текстового элемента значения атрибута `background-color` в действительности окрашивается его блок или прямоугольная область, в которой находится текстовый элемент. Обычно этот блок является прозрачным.

или, пользуясь каскадными таблицами стилей, разместите теги вокруг текста, который хотите окрасить в другой цвет:

```
<p> Розы - <span style="color: #FF0000;"> красные</span>, а фиалки
- <span style="color: #0000FF">синие</span>.</p>
```

Каскадные таблицы стилей также позволяют окрашивать область фона текстового элемента, создавая эффект маркера, пример которого показан на рисунке 53.1.

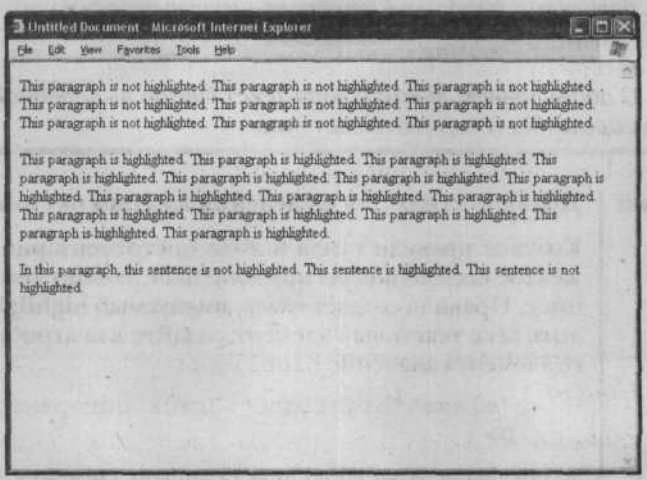


Рис. 53.1. С помощью каскадных таблиц стилей может создаваться эффект маркера для текста

Листинг 53.1. Исходный код страницы, вид которой в браузере показан на рисунке 53.1

```
<p>This paragraph is not highlighted....</p>

<p style="background-color="#FFFF00;">This paragraph is highlight-
ed....</p>

<p>In this paragraph, this sentence is not highlighted. <span
style="background-color: #FFFF00;">This sentence is highlight-
ed.</span> This sentence is not highlighted.</p>
```

Изменение цвета фона таким путем обычно имеет больший смысл, чем изменение цвета самого текста. Скорее всего, посетители вашего Web-сайта не будут полагать, что такой текст является гиперссылкой. Почему? Потому что в этом случае действует сила метафоры.

Большинство людей знает, как выглядит печатная страница, когда кто-нибудь поработал над ней маркером. При изменении цвета фона текстового элемента получаемый эффект выглядит на экране очень похоже. Поэтому люди будут склонны полагать, что фоновый цвет выполняет такую же функцию – привлекает внимание к важному фрагменту.

Когда изменяется цвет самого текста, люди, как правило, связывают это с другой метафорой: гиперссылкой. Изменение цвета текста не влечет за собой ничего, кроме проблем, поскольку вводит в заблуждение аудиторию вашего Web-узла. Посетители щелкают и щелкают на цветном тексте, и ничего не происходит, поскольку этот текст не является гиперссылкой. Данный текст и гиперссылка лишь похожи по внешнему виду.

Совет

Кто говорит, что выделение должно быть желтым? Попробуйте использовать другие цвета: зеленовато-голубой (#00FFFF), ярко розовый (#FF0099), серый (#CCCCCC) или бледно-зеленый (#99FF00).

Чтобы привлечь еще большее внимание к выделенному тексту, добавьте в определение стиля следующую строку кода:

```
font-weight: bold;
```

С добавлением данного кода текст внутри выделенной области приобретет полужирное начертание.

Инструментарий

Правило стиля для выделения желтым цветом

Краткое правило стиля в этом инструментарии предоставляет желтое выделение для применения к любому тексту на Web-странице. Правило создает класс, именуемый **highlight**. Чтобы выделить весь текстовый элемент, задайте для атрибута `class` в теге элемента значение `highlight`:

```
<p class="highlight">Здесь содержится текст.
</p>
```

Чтобы выделить строку текста внутри элемента, разметьте строку с помощью тегов `<span>` и задайте для атрибута `class` в теге `<span>` значение `highlight`:

```
<p>Здесь содержится текст. <span class="high-
light">Здесь содержится выделенный текст.</span>
Здесь содержится текст</p>
```

Поместите это правило между тегами `<style>` в разделе заголовка своей Web-страницы или скопируйте правило как есть, и сохраните во внешнем CSS-файле.

```
.highlight {
  background-color: #FFFF00;
}
```

ГЛАВА 54.
Реабилитация горизонтальных линий

Одним из наиболее полезных, но наименее часто используемых элементов во всем HTML является скромная горизонтальная линия. Ее цель проста, но почетна: создание визуальной разделяющей линии между разделами Web-страницы.

Наилучшее свойство этого элемента – это то, что линия является встроенной графикой. Браузер генерирует ее самостоятельно. Посетителям Web-сайта не надо загружать ее, что уменьшает общее время загрузки графики и улучшает производительность Web-сайта.

Одна из проблем, присущих горизонтальной линии, состоит в том, что готовая линия слишком длинна, неизящна и вызывающа, как можно заметить на рисунке 54.1. Большинство Web-дизайнеров предпочли бы разделить разделы на своих Web-страницах с помощью элегантного пустого пространства, не прибегая к этой безобразной линии. Однако если потратить некоторое время на настройку ее атрибутов, можно создать ряд действительно классных эффектов, примеры которых приведены на рисунке 54.2.

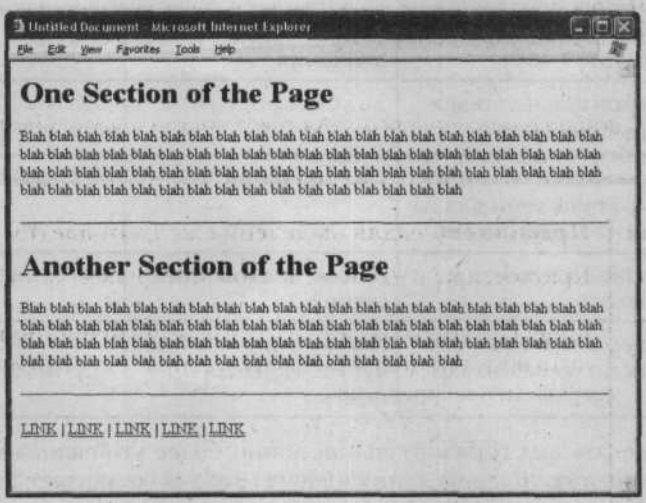


Рис. 54.1. Неизменяемая горизонтальная линия, созданная средствами HTML, имеет несколько неэстетичный вид

В таблице 54.1 приведены атрибуты HTML для тега <hr>, с помощью которого браузером генерируются горизонтальные линии. У пользователя имеется возможность изменять большую часть этих атрибутов, чтобы сделать вид горизонтальных линий более привлекательным.

В первую очередь следует избавиться от эффекта теней, используемого по умолчанию, из-за которого большей частью горизонтальная линия выглядит нелепо:

```
<hr noshade>
```

У атрибута noshade отсутствуют значения. Этот атрибут или присутствует в теге <hr>, или его там нет.

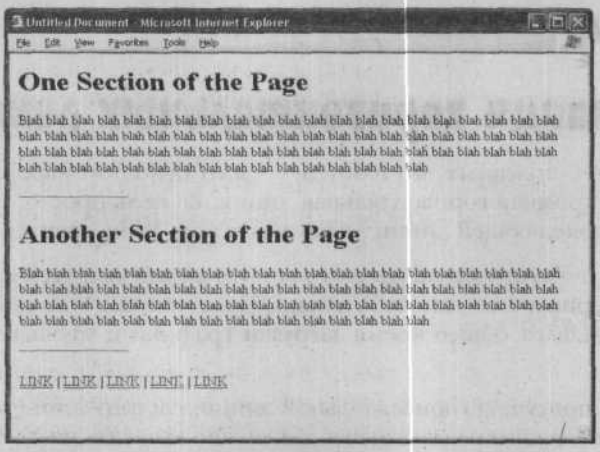


Рис. 54.2. Внеся небольшие изменения в атрибуты линии, можно существенно улучшить ее вид

Табл. 54.1. HTML-атрибуты для горизонтальных линий

Атрибут	Управляемый элемент	Возможные значения	Примеры
`align`	Горизонтальное положение элемента	`left`, `right`, `center`	`<hr align="left">` `<hr align="center">`
`noshade`	Отключение объемного эффекта тени для горизонтальной линии	Нет значений	`<hr noshade>`
`size`	Толщина элемента в пикселах	Любое числовое значение	`<hr size="10">`
`width`	Размер элемента в пикселах по горизонтали	Любое числовое значение, включая проценты	`<hr width="50">` `<hr width="33%">`

Далее, чтобы сделать вид горизонтальной линии более утонченным, уменьшите ее толщину. По умолчанию толщина линии в Internet Explorer составляет 2 пиксела и 3 пиксела в Netscape. Задавая значение атрибута `size` равным 1, вы не только стандартизуете вид элемента в браузерах, но и повышаете эффективность линии. Более тонкая линия лучше выглядит; вряд ли требуется, чтобы горизонтальная линия привлекала к себе слишком большое внимание. Таким образом:

```
<hr noshade size="1">
```

К тому же стандартная горизонтальная линия слишком длинна. По умолчанию она простирается на всю ширину страницы или ширину элемента контейнера, в котором находится, как, например, в ячейке таблицы или в разделе каскадных таблиц стилей `div`. Укороченные горизонтальные линии лучше подходят для большинства случаев. Они вводят дополнительное пустое пространство в макет страницы, что помогает создать более «воздушный» дизайн.

Некоторые дизайнеры предпочитают задать точные значения в пикселах для ширины горизонтальной линии, подобно следующему примеру:

```
<hr noshade size="1" width="200">
```

Точная величина зависит от ширины области, которую занимает горизонтальная линия. Если работа ведется с макетом фиксированной ширины, тогда, вероятно, такой способ задания размера будет представлять наилучший вариант. Однако если макет отличается плавающими размерами, которые зависят от размера окна браузера, горизонтальная линия фиксированной ширины может не подойти. Она будет выглядеть нормально в развернутом окне браузера, но может смотреться слишком большой в неразвернутом полностью окне браузера.

Совет

Цвет горизонтальных линий можно изменять с помощью каскадных таблиц стилей, но браузеры для этого используют разные атрибуты каскадных таблиц стилей. В Internet Explorer поддерживается атрибут `color`*, тогда как в Netscape применяется атрибут* `background-color`*. Если задать атрибут* `color` *в Internet Explorer, то при просмотре в Netscape с цветом линии ничего не происходит, и наоборот, если задать атрибут* `background-color` *в Netscape, при просмотре в Internet Explorer цвет линейки остается прежним.*

Возможным решением этой проблемы является задание обоих атрибутов в определении стиля:

```
hr {
  color: #FF0000; /* для Internet Explorer */
  background-color: #FF0000; /* для Netscape */
}
```

Совет

Еще один классный эффект можно получить, задавая для высоты горизонтальной линии умеренное значение, например, 4 или 5 пикселов, и убирая атрибут `noshade`*. В результате получается пустой прямоугольник.*

Чтобы устранить эту проблему, задайте плавающие размеры и для горизонтальной линии, указав ее ширину в процентах:

```
<hr noshade size="1" width="20%">
```

При значении в 20% ширина линии масштабируется до 20% ширины области, которую она занимает. Таким образом, если горизонтальная линия находится в ячейке таблицы шириной 600 пикселов, то ее ширина будет равна 120 пикселам или 20% от 600.

По умолчанию браузер центрирует горизонтальные линии – это незаметно, поскольку данный элемент автоматически простирается на всю ширину. Центрирование становится заметным при укорачивании ширины линии. Можете оставить ее центрирование, но многие дизайнеры предпочитают располагать линию слева или справа:

```
<hr noshade size="1" width="20%" align="left">
```

Точно такие атрибуты заданы для горизонтальных линий, показанных на рисунке 54.2.

Создание знаков конца

Здесь описывается еще один трюк, в котором используются горизонтальные линии: манипулируя атрибутами `width` и `size`, можно создать прямоугольник в конце раздела текста подобно **знаку конца** во многих журналах или специальному знаку или маркеру в конце статьи. Пример такого знака см. на рисунке 54.3.

Определение

Знак конца *(end sign)) — это маркер или типографский знак, который отображается в конце рассказа или статьи. Он также служит пророчеством о том, что грядет Армагеддон.*

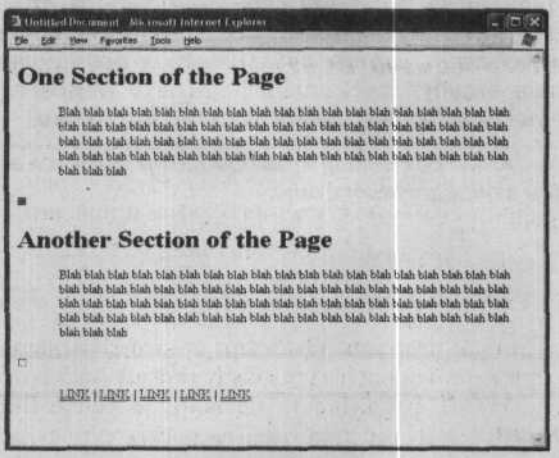


Рис. 54.3. Создайте сплошной или пустой прямоугольник с помощью горизонтальных линий. (Отступы в абзацах получены с помощью атрибута `margin-left` каскадных таблиц стилей.)

Чтобы создать квадрат, задайте одинаковые значения для атрибутов `width` и `size` горизонтальной линии:

```
<hr size="10" width="10">
```

Укажите атрибут `noshade` для получения сплошного квадрата или оставьте атрибут `noshade`, чтобы получить незакрашенный квадрат. Обратите внимание, что на рисунке 54.3 пустой квадрат выглядит как флажок из-за объемного эффекта тени, так что не забывайте об этом. По этой причине, вероятно, лучше использовать в качестве знаков конца окрашенные сплошным цветом квадратики.

Совет

Сплошные прямоугольники, получаемые из горизонтальной линии, выглядят в Netscape с закругленными углами, тогда как пустые прямоугольники отображаются с обычными углами. Учитывайте это.

ГЛАВА 55.
Добавление строки даты и метки времени

Строка даты – это фрагмент текста, сообщающий текущую дату и обычно отображаемый вверху страницы. Точно также, **метка времени** – это фрагмент текста, сообщающий текущее время и чаще всего отображаемый внизу страницы. Оба элемента удобны для придания ощущения частого обновления Web-страницы. Ничто так не радует посетителей, как посещение часто обновляемого, со свежими данными Web-сайта. Даже если вы обновляете содержимое Web-сайта от случая к случаю, вид текущей даты и текущего времени помогает создать постоянное ощущение новизны.

Определение

***Строка даты** (dateline) – фрагмент текста, сообщающий текущую дату. **Метка времени** (timestamp) – фрагмент текста, сообщающий текущее время.*

Вы можете самостоятельно вставлять строку даты и метку времени с помощью ряда простых функций JavaScript. Полные сведения об этом приводятся в разделе «Инструментарий».

Инструментарий

Генерирование строки даты

Данная функция JavaScript состоит из двух частей. Основной сценарий вставляется между тегами `<script>` в разделе заголовка HTML-файла или в отдельном JS-файле. Внутри тела Web-страницы, там, где требуется вставить строку даты, необходим дополнительный тег `<script>`.

Совет

Браузер воспроизводит строку даты в виде местного времени посетителя или в том формате, в котором дата и время отображаются на компьютере посетителя.

```
function doDateline() {
/* Вначале извлечем текущую дату. Это достигается
с помощью объекта Date языка JavaScript. С помощью
этой строки кода извлекается текущая дата и поме-
щается в переменную now. */
   var now = new Date();
/* Следующие строки кода предназначены для извле-
чения из переменной now дня недели, месяца, кален-
дарной даты и года. Все эти значения являются чис-
ловыми. */
```

```
var day = now.getDay();
var month = now.getMonth();
var date = now.getDate();
var year = now.getYear();
/* Web-браузер Internet Explorer возвращает правильный год, тогда как Netscape и Opera уменьшают год ровно на 1900 лет. Следующий блок кода с операторами if/then позволяет добиться вывода правильного года во всех браузерах. */
if (year <2000) {
  year += 1900;
}
/* Инициализируем переменную dateline. Эта переменная будет содержать в конечном итоге текст строки даты. */
var dateline = "";
/* Теперь сформируем текст строки даты. День недели в настоящее время существует в числовой форме, со значениями от 0 до 6, представляющими дни от Sunday (воскресенье) до Saturday (субботы). Следующие блоки кода if/then добавляют корректное название дня в переменную dateline на основе числового значения переменной day.

Обратите внимание на запятую и конечный пробел после названия дня недели. Пробел необходим, чтобы следующий элемент, название месяца, отделялся от запятой.

Если требуется сформировать простую строку даты, такую, как 12/22/2005, опустите этот блок кода. */
if (day == 0) {
  dateline = "Sunday, ";
}
if (day == 1) {
  dateline = "Monday, ";
}
if (day == 2) {
  dateline = "Tuesday, ";
}
if (day == 3) {
  dateline = "Wednesday, ";
```

```
    }
    if (day == 4) {
      dateline = "Thursday, ";
    }
    if (day == 5) {
      dateline = "Friday, ";
    }
    if (day == 6) {
      dateline = "Saturday, ";
    }
/* Месяц также существует в числовой форме, где 0
соответствует January (январь), 1 — February (фев-
раль), 2 — March (март) и так далее до числа 11
соответствующего December (декабрь). Эти блоки
кода if/then добавляют правильное название месяца
в переменную dateline. Обратите внимание на конеч-
ный пробел, необходимый для надлежащего формати-
рования.

Если необходимо использовать европейский стиль для
строки даты, поменяйте местами этот блок кода с
последующим блоком кода.

Если требуется получить простую строку даты, заме-
ните этот блок кода следующей строкой:
  dateline += (month + 1) + "/";
*/
    if (month == 0) {
      dateline = "January ";
    }
    if (month == 1) {
      dateline = "February ";
    }
    if (month == 2) {
      dateline = "March ";
    }
    if (month == 3) {
      dateline = "April ";
    }
    if (month == 4) {
      dateline = "May ";
```

```
    }
    if (month == 5) {
      dateline = "June ";
    }
    if (month == 6) {
      dateline = "July ";
    }
    if (month == 7) {
      dateline = "August ";
    }
    if (month == 8) {
      dateline = "September ";
    }
    if (month == 9) {
      dateline = "October ";
    }
    if (month == 10) {
      dateline = "November ";
    }
    if (month == 11) {
      dateline = "December ";
    }
/* Следующая строка кода добавляет числовую дату в
переменную dateline, а также добавляет запятую с
пробелом.

Для получения даты в европейском формате поменяйте
этот блок кода с предыдущим, удалите запятую и про-
бел, а также добавьте запятую после имени каждого
месяца в предыдущем блоке кода.

Если требуется получить простую строку даты, заме-
ните эту строку кода следующей строкой:
  dateline += date + "/";
*/
    dateline += date + ", ";
/* С помощью следующей строки кода добавляется год
в числовой форме в переменную dateline. */
    dateline += year;
```

```
/* Переменная dateline готова для отображения. Эта
строка кода предназначена для записи dateline на
Web-странице. */
    document.write(dateline);
}
```

Этим завершается основной сценарий. Теперь в том месте, где должна отображаться строка даты, добавьте следующий код:

```
<script        language="JavaScript">doDateline();
</script>
```

В зависимости от ваших предпочтений можно разметить данный тег `<script>` как абзац, заголовок или текстовый элемент любого другого типа:

```
<p><script     language="JavaScript">doDateline();
</script></p>
```

Поместите основной сценарий вместе с вызовом `doDateline` и вы получите вид строки даты, похожий на тот, что показана на рисунке 55.1. Простая версия строки даты показана на рисунке 55.2.

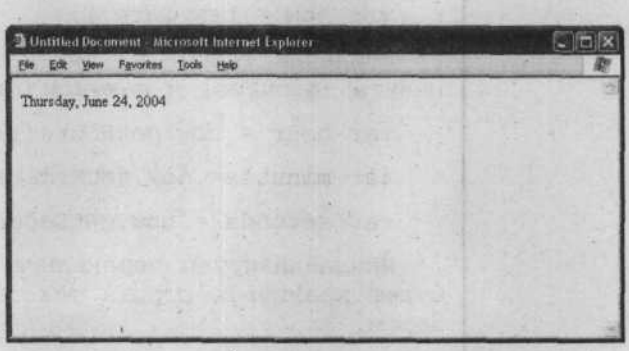


Рис. 55.1. Строку даты можно добавить на любую Web-страницу

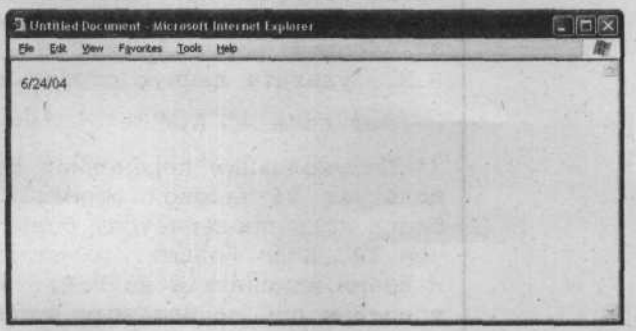


Рис. 55.2. Если предпочитаете, можно создать простую строку даты

Инструментарий

Генерирование метки времени

Данная функция JavaScript во многом сходна с предыдущей функцией. Основной сценарий располагается между тегами <script> в разделе заголовка HTML-файла или во внешнем файле JavaScript. Дополнительный тег <script> внутри тела Web-страницы служит для отображения метки времени.

Совет

Как и в случае строки даты, данный сценарий метки времени генерирует время согласно внутренним часам в компьютере посетителя.

Далее приводится код основной части сценария:

```
function doTimestamp() {

/* Как и в предыдущем сценарии, начнем с получения
текущей даты и поместим ее в переменную с названием now. */

   var now = new Date();

/* Извлечем теперь из переменной now часы (hour),
минуты (minutes) и секунды (seconds). */

   var hour = now.getHours();

   var minutes= now.getMinutes();

   var seconds = now.getSeconds();

/* Инициализируем переменную timestamp, в которой
будет храниться строка текста, отображаемая браузером. */

   var timestamp = "";

/* Зададим для переменной time значение " A.M." с
предшествующим пробелом для надлежащего форматирования. Если требуется отображать метку времени в
24-часовом формате, как 16:12:25 вместо 4:12:25
P.M., удалите данную строку кода. */

   var time =" A.M.";

/* По умолчанию переменная hours в JavaScript использует 24-часовой формат. С помощью следующего
блока кода проверяется, больше ли переменная hour,
чем 12. Если больше, то из значения вычитается 12
и время изменяется на P.M., опять с предшествующим
пробелом для надлежащего форматирования.
```

```
Если необходимо отображать время в 24-часовом формате, замените этот блок кода следующим кодом:
   if (hour < 10) {
     hour = "0" + hour;
   }
Данный блок кода if/then добавляет ноль слева от часов там, где это надо, чтобы шестой час отображался как "06", а не как "6". */
   if (hour > 12) {
     hour -= 12;
     time = " P.M.";
   }
/* Подобным образом последующие блоки кода if/then форматируют минуты и секунды так, чтобы они отображались с нулями слева, где необходимо. */
   if (minutes < 10) {
     minutes = "0" + minutes;
   }

   if (seconds < 10) {
     seconds = "0" + seconds;
   }

/* Теперь сформируем строку метки времени. */
   timestamp = hour + ":" + minutes + ":" + seconds + time;
/* Запишем метку времени на страницу. */
   document.write(timestamp);
}
```

Добавим следующий код в том месте, где требуется отображать на Web-странице метку времени:

```
<script      language="JavaScript">doTimestamp();
</script>
```

Возможно, вы пожелаете сдобрить метку времени дополнительным форматированием:

```
<p>
  <em>This page generated at
```

```
    <script  language="JavaScript">doTimestamp();
</script></em>
</p>
```

Результат дополнительного форматирования показан на рисунке 55.3.

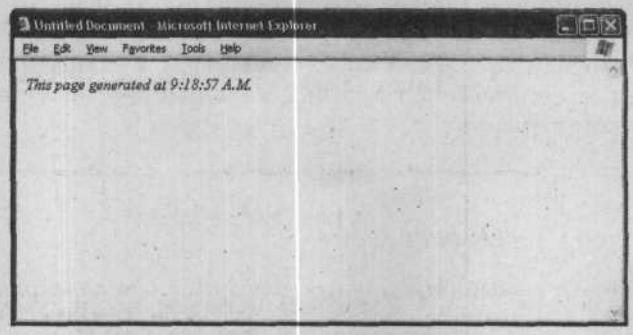


Рис. 55.3. Добавление метки времени

ГЛАВА 56.
Разработка таблиц данных

Таблицы данных – это элементы таблиц, которые содержат строки и столбцы данных. Что касается мнения членов консорциума W3C, то таблицы данных являются единственным типом таблиц, которые стоят обсуждения, поскольку спецификация HTML определяет теги `<table>` именно для этих целей. В консорциуме W3C не любят, когда заимствуют теги `<table>` для «неправедных» целей, например, для создания макета страницы.

Определение

Таблица данных *(data table) – HTML-таблица, в которой содержатся строки и столбцы данных, а не макет страницы.*

В таблице 56.1 показаны стандартные HTML-теги таблицы. Многие из этих тегов не имеют смысла в контексте макетных таблиц, поэтому, возможно, они вам не известны. Но когда вы создаете таблицу данных, помните о них, поскольку надлежащая разметка улучшает доступность результатов вашего труда для других пользователей.

Табл. 56.1. Стандартные HTML-теги таблицы в порядке их появления в коде

Тег	Значение
`table`	Таблица
`caption`	Заголовок таблицы
`col`	Столбец в таблице
`colgroup`	Группа связанных столбцов в таблице
`thead` .	Заголовок раздела таблицы
`tfoot`	Нижний раздел таблицы
`tbody`	Основной раздел таблицы; может приводиться несколько раз в таблице для обозначения разных разделов в теле таблицы
`tr`	Строка в таблице
`th`	Ячейка таблицы, которая содержит заголовок строки или столбца
`td`	Ячейка таблицы, которая содержит обычный элемент данных

На рисунке 56.1 показана таблица данных, в которой используются все десять тегов из таблицы 56.1.

Просмотрите исходный код для показанной Web-страницы, чтобы разобраться, как работает HTML-разметка.

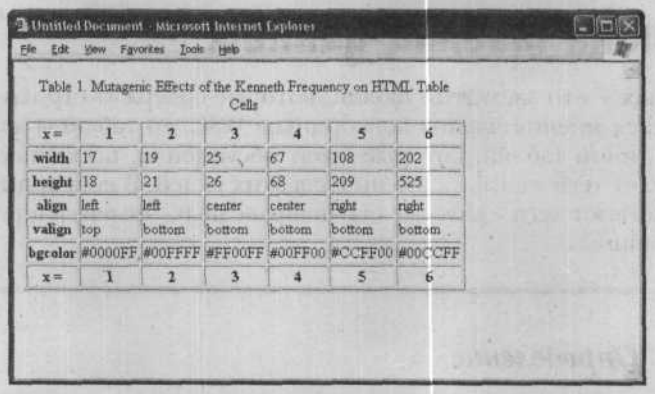


Рис. 56.1. В этой таблице данных используются все десять тегов из таблицы 56.1

Листинг 56.1. Исходный код страницы, вид которой в браузере показан на рисунке 56.1

```
<table border="1">

<!- Первая часть содержимого представляет собой заголовок таблицы -
>

  <caption>
     Table 1. Mutagenic Effects of the Kenneth Frequency on HTML
Table Cells
  </caption>

<!- Далее идет описание структуры столбца таблицы. Теги <col> и
<colgroup> в этой конкретной таблице показывают структуру одного
столбца слева, за которой следует группа из шести столбцов. Обра-
тите внимание, что для тега <col> отсутствует его закрывающая вер-
сия, тогда как для тега <colgroup> закрывающий тег есть. ->

  <col>
  <colgroup span="6"></colgroup>

<!- Далее идет головная часть таблицы, в которой описывается верх-
няя строка таблицы данных. Для ячеек таблицы используются теги <th>
```

```
вместо тегов <td>, чтобы показать, что они являются ячейками головной части таблицы. -->

  <thead>
    <tr>
      <th>x =</th>
      <th>1</th>
      <th>2</th>
      <th>3</th>
      <th>4</th>
      <th>5</th>
      <th>6</th>
    </tr>
  </thead>

<!- Далее идет нижняя часть таблицы, в которой описывается последняя строка таблицы данных. Она должна располагаться до любого из основных разделов таблицы. Нижний раздел часто полностью совпадает с головной частью, как в этом примере, но в нижней части может иметься уникальное содержимое, такое, как строка итогов. -->

  <tfoot>
    <tr>
      <th>x =</th>
      <th>1</th>
      <th>2</th>
      <th>3</th>
      <th>4</th>
      <th>5</th>
      <th>6</th>
    </tr>
  </tfoot>

<!- Далее идет тело таблицы. Содержимое этой конкретной таблицы разбито на три разные основные части. Первая из них начинается здесь.
-->
```

```
  <tbody>
    <tr>
      <th>width</th>
      <td>17</td>
      <td>19</td>
      <td>25</td>
      <td>67</td>
      <td>108</td>
      <td>202</td>
    </tr>
    <tr>
      <th>height</th>
      <td>18</td>
      <td>21</td>
      <td>26</td>
      <td>68</td>
      <td>209</td>
      <td>525</td>
    </tr>
  </tbody>

<!- Здесь размечается второй основной раздел. ->

  <tbody>
    <tr>
      <th>align</th>
      <td>left</td>
      <td>left</td>
      <td>center</td>
      <td>center</td>
      <td>right</td>
      <td>right</td>
    </tr>
    <tr>
```

```
      <th>valign</th>
      <td>top</td>
      <td>bottom</td>
      <td>bottom</td>
      <td>bottom</td>
      <td>bottom</td>
      <td>bottom</td>
    </tr>
  </tbody>

<!- Здесь размечается третий основной раздел. ->

  <tbody>
    <tr>
      <th>bgcolor</th>
      <td>#0000FF</td>
      <td>#00FFFF</td>
      <td>#FF00FF</td>
      <td>#00FF00</td>
      <td>#0000FF</td>
      <td>#00CCFF</td>
    </tr>
  </tbody>

<!-Добавим закрывающий тег таблицы, чтобы завершить разметку. ->

</table>
```

Использование большинства тегов группировки

Возможно, вы удивляетесь, что хорошего в HTML-тегах группировки: <col>, <colgroup>, <thead>, <tfoot> и <tbody>. Все таки, эти теги не добавляют никакого содержимого в таблицу. Теги <col> и <colgroup> просто описывают структуру столбцов в таблице, а теги <thead>, <tfoot> и <tbody> размечают разделы таблицы. Зачем надо в первую очередь кодировать в таблице эти сведения?

Совет

При использовании тегов <col> и <colgroup> с тегами <thead>, <tfoot> и <tbody> в одной и той же таблице браузер Netscape работает неустойчиво и не всегда применяет атрибуты col и colgroup. Простой способ решения этой проблемы состоит в отказе от использования col и colgroup вместе с thead, tfoot и tbody. Выберите одну группу тегов или другую — в зависимости от того, которая из них больше подходит для конкретной таблицы данных.

Теги группировки не только улучшают доступность таблицы, описывая подробно ее структуру, они предоставляют ряд удобных способов форматирования. Возьмите в качестве примера таблицу, показанную на рисунке 56.1. Предположим, что вы хотите выровнять вправо содержимое каждой ячейки таблицы в первом столбце. Нет необходимости писать строку кода:

```
<td align="right">
```

в каждой ячейке таблицы. Просто укажите атрибут align один раз в теге <col>:

```
<col align="right">
```

Лучше задавать выравнивание в виде определения CSS-стиля:

```
<col style="text-align: right;">
```

Эта сокращенная запись работает также и с тегом <colgroup>:

```
<colgroup span="6" style="text-align: center;"></colgroup>
```

Добавьте атрибут width, чтобы задать размер столбцов по горизонтали:

```
<col style="text-align: right; width: 75px;">
```

```
<colgroup span="6" style="text-align: center; width: 100px;"></col-
group>
```

HTML-атрибуты и определения CSS-стилей можно также применить к разделам thead, tfoot и tbody:

```
<htead style="font-family: Arial, Helvetica, sans serif; font-
size: 10px; font-style: italic;">
  <!– Содержимое располагается здесь –>
</thead>
```

Работа с атрибутами colspan и rowspan

Атрибуты colspan и rowspan представляют беду для макетных таблиц, поскольку являются источником всевозможных проблем, связанных со странными разрывами. Тем не менее, эти атрибуты удобно использовать для таблиц данных, поскольку именно для них предназначались cospan и rowspan разработчиками HTML.

С технической точки зрения атрибут `colspan` определяет, сколько столбцов занимает или охватывает конкретная ячейка таблицы, тогда как атрибут `rowspan` определяет количество охваченных строк. Лучший способ понять, как работают атрибуты `colspan` и `rowspan` – рассмотреть пример, показанный на рисунке 56.2.

Определение

Атрибут `colspan` определяет количество столбцов, охватываемых ячейкой таблицы, тогда как атрибут `rowspan` определяет число охватываемых строк.

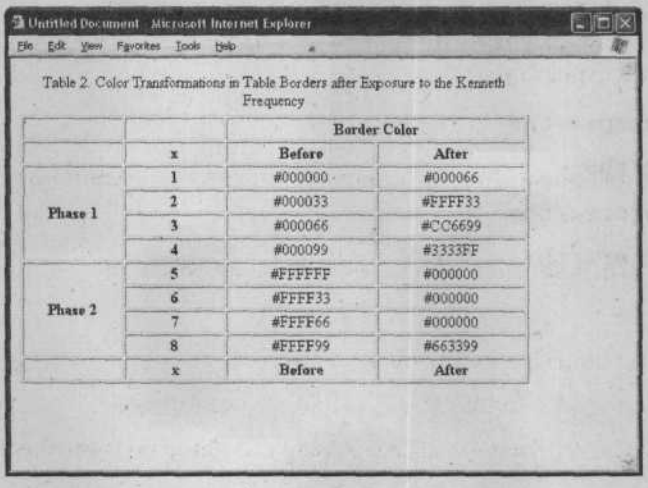


Рис. 56.2. Для разметки этой таблицы применяются атрибуты `colspan` и `rowspan`

Листинг 56.2. Исходный код страницы, вид которой в браузере показан на рисунке 56.2

```
<table border="1">

  <caption>Table 2. Color Transformations in Table Borders after
Exposure to the Kenneth Frequency</caption>

  <colgroup span= "2" style="width: 100px;"></colgroup>
     <colgroup span="2" style="width: 150px; text-align:
center;"></colgroup>

<!- Здесь головной элемент состоит из двух строк таблицы. Несколько ячеек содержат неразрывные пробелы ( ). Используем их для заполнения пустых ячеек таблицы.
```

```
Последняя ячейка в первой строке содержит атрибут colspan. Как можно видеть на рисунке 56.2, ячейка "Border Color" охватывает два столбца: столбец "Before" и столбец "After".
-->

  <thead>
    <tr>
      <th> </th>
      <th> </th>
      <th colspan="2">Border Color</th>
    </tr>
    <tr>
      <th> </th>
      <th>x</th>
      <th>Before</th>
      <th>After</th>
    </tr>
  </thead>

  <tfoot>
    <tr>
      <th> </th>
      <th>x</th>
      <th>Before</th>
      <th>After</th>
    </tr>
  </tfoot>

<!- В этой таблице имеются две основные части, заключенные в теги <tbody>. В первой ячейке первой строки таблицы содержится атрибут rowspan. Посмотрев на рисунок, вы увидите, что ячейка "Phase 1" охватывает строки 1, 2, 3 и 4. ->

  <tbody>
```

```
    <tr>
      <th rowspan="4">Phase 1</th>
      <th>1</th>
      <td>#000000</td>
      <td>#000066</td>
    </tr>
    <tr>
      <th>2</th>
      <td>#000033</td>
      <td>#FFFF33</td>
    </tr>
    <tr>
      <th>3</th>
      <td>#000066</td>
      <td>#CC6699</td>
    </tr>
    <tr>
      <th>4</th>
      <td>#000099</td>
      <td>#3333FF</td>
    </tr>
  </tbody>

<!- Здесь размечается вторая основная часть, заключенная в теги
<tbody>. Опять атрибут rowspan в ячейке "Phase 2" обуславливает ох-
ват этой ячейкой четырех строк. ->

  <tbody>
    <tr>
      <th rowspan="4">Phase 2</th>
      <th>5</th>
      <td>#FFFFFF</td>
      <td>#000000</td>
    </tr>
    <tr>
```

```
      <th>6</th>
      <td>#FFFF33</td>
      <td>#000000</td>
    </tr>
    <tr>
      <th>7</th>
      <td>#FFFF66</td>
      <td>#000000</td>
    </tr>
    <tr>
      <th>8</th>
      <td>#FFFF99</td>
      <td>#663399</td>
    </tr>
  </tbody>

</table>
```

Как можно видеть из листинга 56.2, атрибуты `colspan` и `rowspan` находятся в теге ячейки таблицы `<th>` или `<td>` в зависимости от того, является охватывающей ячейкой ячейка головной части таблицы или обычная ячейка. Значения этих атрибутов определяют число охватываемых столбцов или строк.

Совет

Одиночная ячейка таблицы может содержать оба атрибута `colspan` *и* `rowspan`.

Вы заметите, что в коде строки, включающей ячейку, которая охватывает несколько столбцов или строк, по-видимому, пропущена часть разметки. Например, в коде первой строки данных таблицы, показанной на рисунке 56.2, вы обнаружите три ячейки таблицы:

```
<tr>
  <th> </th>
  <th> </th>
```

```
  <th colspan="2">Border Color</th>
</tr>
```

В следующей строке имеется четыре ячейки таблицы:

```
<tr>
  <th> </th>
  <th>x</th>
  <th>Before</th>
  <th>After</th>
</tr>
```

Отсутствующий код в первой строке – это вполне нормально. В действительности именно так и должно быть. Ячейка таблицы с атрибутом `colspan` считается за 2 ячейки, в этом случае в обеих строках содержится по четыре ячейки таблицы.

Еще одно замечание: теги `<colgroup>` вверху кода таблицы имеют следующую структуру:

```
<colgroup span="2"></colgroup>
<colgroup span="2"></colgroup>
```

Диапазоны тегов `<colgroup>` в сумме дают четыре столбца, а не три, хотя, как видно на рисунке, самая первая строка таблицы состоит из трех ячеек. Здесь опять ячейку с атрибутом `colspan`, равным 2, необходимо считать как две ячейки, а не одну. Таким образом, все совпадает. Теги `<colgroup>` описывают четыре столбца, и, действительно, в этом разделе таблицы содержится четыре столбца.

Выбор значений атрибутов cellpadding и cellspacing

Обычно при создании макетной таблицы в теге `<table>` для атрибутов `cellpadding` и `cellspacing` задаются значения равные 0. Однако когда создается таблица данных, атрибуты `cellpadding` и `cellspacing` служат полезными инструментами форматирования. Атрибут `cellpadding` определяет величину внутреннего пустого пространства между границами ячейки таблицы и ее содержимым, тогда как атрибут `cellspacing` определяет величину внешнего зазора между отдельными ячейками таблицы.

При задании в теге `<table>` атрибутов `cellpadding` и `cellspacing` их значения применяются ко всем ячейкам в таблице:

```
<table border="1" cellpadding="5" cellspacing="8">
```

На рисунке 56.3 показана таблица, для которой установлены именно эти значения атрибутов. Сравните ее с таблицей, показанной на рисунке 56.2, чтобы увидеть разницу во внешнем виде, возникающую из-за небольших значений атрибутов `cellpadding` и `cellspacing`.

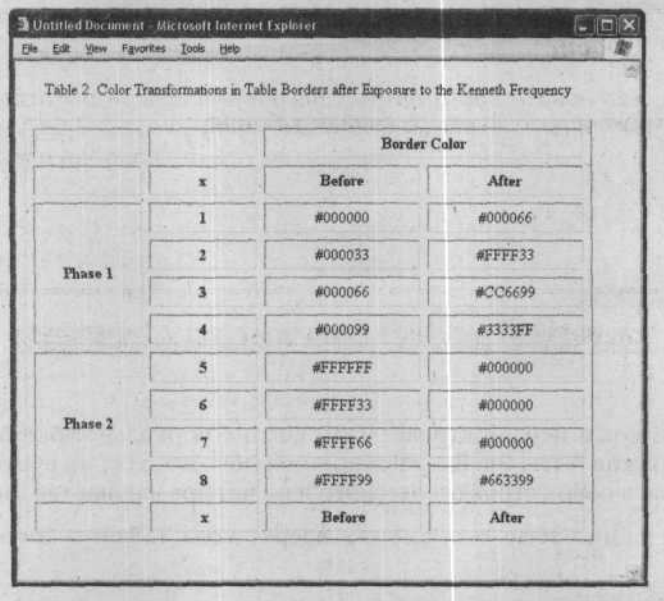


Рис. 56.3. В этой таблице данных отрегулированы зазоры внутри и вне ячеек с помощью атрибутов `cellpadding` *и* `cellspacing`

Используйте атрибуты `cellpadding` и `cellspacing`, чтобы добавить немного пустого пространства в таблицу, плотно заполненную данными. Таблицы обычно легче воспринимаются, когда в них есть немного дополнительного свободного пространства. Однако здесь важно не перестараться. Слишком много пустого пространства также плохо, как и его нехватка. В таких ситуациях следует руководствоваться зрительным восприятием. Если таблица не достаточно удобна для чтения, попробуйте несколько изменить значения атрибутов `cellpadding` и `cellspacing` в ту или другую сторону.

Определение

Значением атрибута `cellpadding` *определяется величина внутреннего пустого пространства между границами ячейки таблицы и ее содержимым. Значением атрибута* `cellspacing` *определяется величина зазора между отдельными ячейками таблицы.*

Если в таблице данных используются фоновые цвета, возможно, атрибут `cellpadding` будет лучшим выбором для регулировки зазоров. Атрибут `cellspacing` часто является причиной видимых зазоров между ячейками, нарушая непрерывное поле фонового цвета, как, например, на рисунке 56.4. Как видно из рисунка 56.5, атрибут `cellpadding` не оказывает такого эффекта до тех пор, пока значение атрибута `cellspacing` равно 0.

Совет

Установите необходимое значение набивки для отдельной ячейки таблицы с помощью CSS-атрибутов `padding-top`, `padding-bottom`, `padding-left` *и* `padding-right` *подобно следующему примеру:*

```
<td style="padding-top: 3px; padding-bottom: 4px;
    padding-left: 2px; padding-right: 5px;">
```

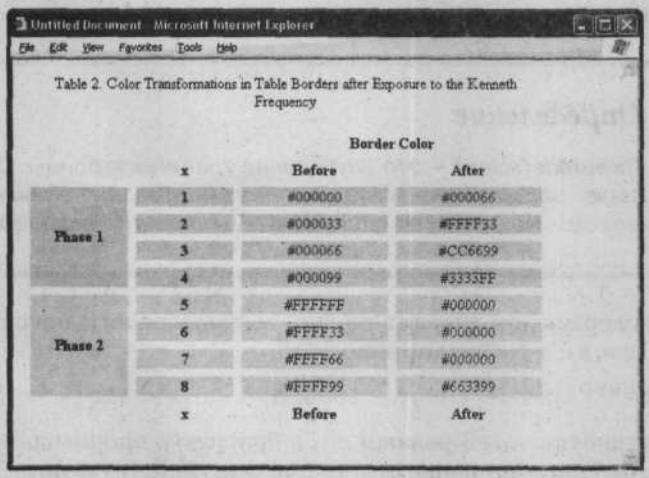


Рис. 56.4. Осторожно подходите к изменению значения атрибута `cellspacing`*, когда в ячейках таблицы используется цветной фон. При задании значения атрибута* `cellspacing` *часто происходит разрыв сплошного цветного фона в таблице*

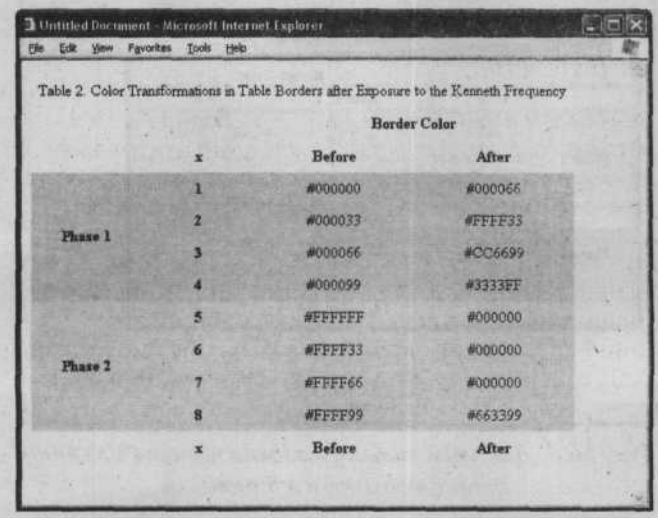


Рис. 56.5. Чтобы сохранить цветной фон таблицы сплошным, пользуйтесь атрибутом `cellpadding`*, а значение атрибута* `cellspacing` *установите равным 0*

Изменение границы

Вряд ли кто-нибудь использует границы в макетных таблицах, однако все пользуются границами в таблицах данных. Табличный атрибут `border` управляет толщиной границы, а атрибут `bordercolor` определяет ее цвет:

```
<table border="5" bordercolor="#000000">
```

На рисунке 56.6 показана таблица с приведенными выше свойствами. Внешняя граница таблицы отображается с толщиной 5 пикселов. Внутренние границы, или так называемые **линейки**, сохраняют свое нормальное значение толщины, но заимствуют цвет границы таблицы.

Определение

Линейки *(rules) — это внутренние границы таблицы. Они сохраняют свою нормальную толщину независимо от значения атрибута* `border`*, но окрашиваются в цвет внешней границы таблицы.*

Способ рисования браузером линеек в таблице можно изменить посредством задания атрибута `rules`, как в следующем примере:

```
<table border="5" bordercolor="#000000" rules="cols">
```

При установке атрибута `rules` равным `cols` браузером прорисовываются линейки только вокруг столбцов, как, например, на рисунке 56.7. В таблице 56.2 приведены возможные значения атрибута `rules`.

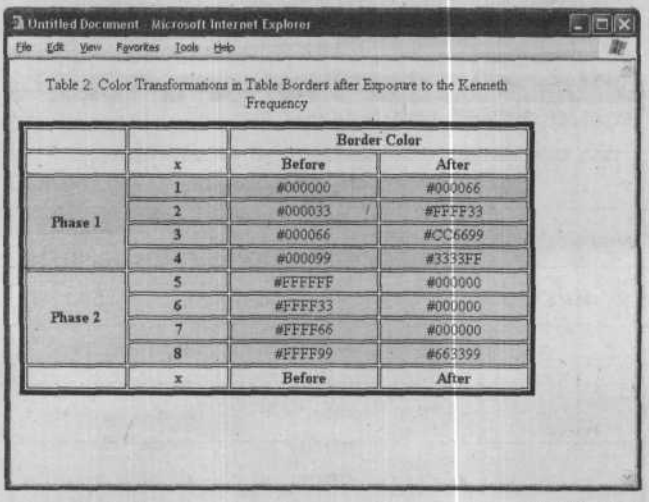


Рис. 56.6. Для этой таблицы задана черная (#000000) граница размером в 5 пикселов

Совет

Границы таблицы отображаются в Netscape с объемным эффектом теней. В браузере Internet Explorer границы таблицы отображаются сплошными.

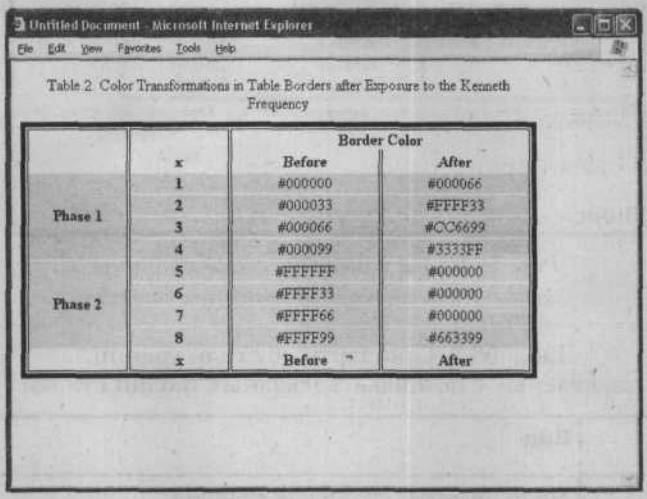


Рис. 56.7. Задайте атрибут `rules` в теге `<table>`, чтобы определить, как будут воспроизводиться линейки браузером. Здесь значение атрибута `rules` равно `cols`

Табл. 56.2. Значения атрибута `rules`

Значение	Линейки рисуются вокруг
`all`	Всех ячеек таблицы
`cols`	Только столбцов
`rows`	Только строк
`groups`	Только групп: `col`, `colgroup`, `thead`, `tfoot` и `tbody`
`none`	Ничего; линейки отсутствуют

С помощью каскадных таблиц стилей можно изменить стиль границы:

```
<table border="5" bordercolor="#000000" rules="groups" style="bor-
der-style: dotted;">
```

На рисунке 56.8 показано, как выглядит в браузере стиль точечной (`dotted`) границы. Другие стили границы см. в таблице 56.3.

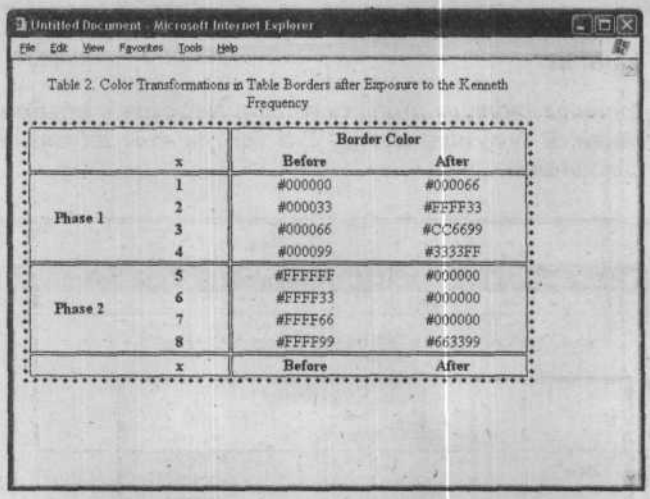


Рис. 56.8. Для изменения стиля границы используйте каскадные таблицы стилей

Табл. 56.3. Стандартные стили границ, задаваемые с помощью каскадных таблиц стилей

Стиль	Вид
`solid`	Непрерывная, сплошная граница
`dashed`	Штриховые линии
`dotted`	Точечные линии
`double`	Пара сплошных границ
`none`	Отсутствие границ

В каскадных таблицах стилей определяется множество других стилей границ, но не все браузеры поддерживают их.

Для границ разных сторон таблицы можно задать разные значения, как на рисунке 56.9:

```
<table border="5" bordercolor="#000000" rules="groups" style="bor-
der-top-style: dashed; border-bottom-style: dotted; border-left-
style: double; border-right-style: none;">
```

В действительности с помощью каскадных таблиц стилей можно установить толщину и цвет границы независимо для всех четырех сторон. В таблице 56.4 приводится список стандартных CSS-атрибутов границы, которые можно взять на вооружение.

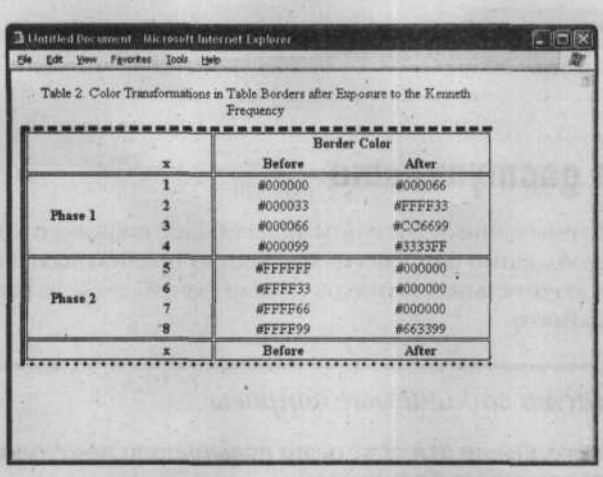


Рис. 56.9. С помощью каскадных таблиц стилей для верхней, правой, нижней и левой границ можно задать разные стили

Табл. 56.4. Стандартные CSS-атрибуты границы

Атрибут	Управляемый элемент	Пример
border-color	Цвет всех четырех сторон границы	border-color: #FF0000;
border-style	Стиль всех четырех сторон границы	border-style: dotted;
border-width	Вес или толщина всех четырех сторон границы	border-width: 10px;
border-bottom-color	Цвет нижней границы	border-bottom-color: #00FF00;
border-bottom-style	Стиль нижней границы	border-bottom-style: none
border-bottom-width	Вес или толщина нижней границы	border-bottom-width: 8px;
border-left-color	Цвет левой границы	border-left-color: #0000FF;
border-left-style	Стиль левой границы	border-left-style: solid;
border-left-width	Вес или толщина левой границы	border-left-width: 5px;
border-right-color	Цвет правой границы	border-right-color: #FFFF00;
border-right-style	Стиль правой границы	border-right-style: dashed;
border-right-width	Вес или толщина правой границы	border-right-width: 4px;
border-top-color	Цвет верхней границы	border-top-color: #FFFFFF;
border-top-style	Стиль верхней границы	border-top-style: double;
border-top-width	Вес или толщина верхней границы	border-top-width: 6px;

Каскадными таблицами стилей определяются также другие атрибуты, но не все браузеры поддерживают их.

Повышение доступности

Чтобы сделать таблицы данных доступными для более широкого круга людей (включая инвалидов), необходимо расставить множество правильных тегов. Для добавления некоторых заключительных штрихов в разметку таблицы потребуется еще немало дополнительных тегов.

Часто задаваемые вопросы

Пригодны ли эти советы по повышению доступности таблиц для современных браузеров?

На самом деле нет. Сохраните эти советы в категории «Совместимость снизу вверх». Самые последние версии основных браузеров не используют эту дополнительную разметку, и текущие инструменты доступности также не используют ее. Применяйте эти методы, чтобы сделать таблицы пригодными для следующего поколения технологии Web-просмотра.

Если текст в заголовках таблицы излишне многословен:

```
<th>Intensity of the Kenneth Frequency</th>
<th>Fluctuation in Beta Transmission</th>
<th>Margin of Error</th>
```

можно добавить свои собственные сокращения в атрибут `abbr` в теге `<th>`. Устройства чтения с экрана заменяют этими сокращениями полную фразу, что сокращает монотонное речевое воспроизведение таблицы данных.

```
<th abbr="Intensity">Intensity of the Kenneth Frequency</th>
<th abbr="Fluctuation">Fluctuation in Beta Transmission</th>
<th abbr="Error">Margin of Error</th>
```

Можно также точно указать, какие ячейки таблицы под какими ячейками заголовка располагаются. Чтобы сделать это, задайте для каждого тега `<th>` в группе заголовка таблицы уникальные идентификаторы (ID). Затем для каждой ячейки таблицы укажите эти идентификаторы в атрибуте `headers`, подобно следующему примеру:

```
<table border="1" rules="groups">

  <caption>Table 3: Intensity of the Kenneth Frequency and
Fluctuation in Beta Transmission</caption>

  <col width="300" align="center">
  <col width="300" align="center">
```

```
<col width="160" align="center">

<thead>
  <tr>
    <th id="intensity" abbr="Intensity">Intensity of the
Kenneth Frequency</th>
    <th id="fluctuation" abbr="Fluctuation">Fluctuation in Beta
Transmission</th>
    <th id="error" abbr="Error">Margin of Error</th>
  </tr>
</thead>

<!- В этой таблице отсутствует раздел нижнего заголовка, что вполне нормально. Нет необходимости указывать все возможные группы в каждой таблице. Достаточно включить те из них, которые вам необходимы. ->

<tbody>
  <tr>
    <td headers="intensity">1</td>
    <td headers="fluctuation">90</td>
    <td headers="error">6%</td>
  </tr>
  <tr>
    <td headers="intensity">2</td>
    <td headers="fluctuation">128</td>
    <td headers="error">35%</td>
  </tr>
  <tr>
    <td headers="intensity">3</td>
    <td headers="fluctuation">2726</td>
    <td headers="error">32%</td>
  </tr>
  <tr>
    <td headers="intensity">4</td>
    <td headers="fluctuation">263443</td>
    <td headers="error">32%</td>
```

```
    </tr>
    <tr>
      <td headers="intensity">5</td>
      <td headers="fluctuation">2</td>
      <td headers="error">98%</td>
    </tr>
  </tbody>

</table>
```

Совет

Идентификаторы (ID) всех тегов на странице должны быть уникальными. Не используйте тот же самый идентификатор для другого тега заголовка или любого другого HTML-элемента на странице.

Таблица, код которой приведен выше, показана на рисунке 56.10.

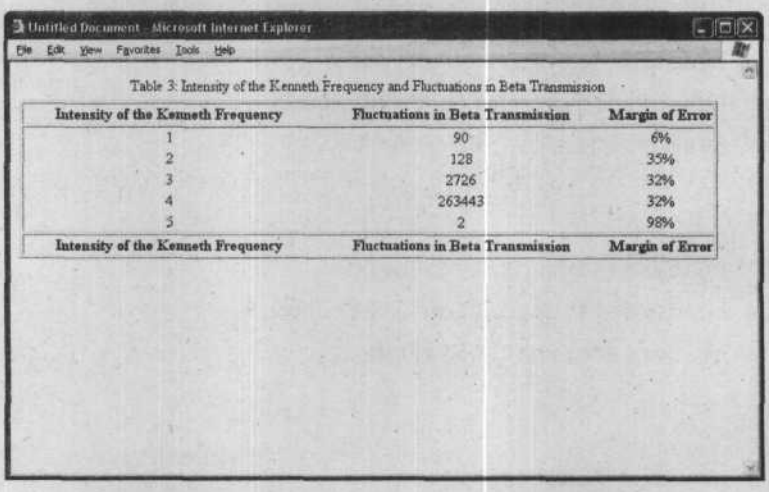


Рис. 56.10. Вид таблицы данных в браузере не изменяется при задании идентификаторов (ID) и атрибутов `headers`*, но эта дополнительная разметка будет иметь значение для будущих устройств чтения с экрана*

Другой способ улучшения доступности таблицы данных состоит в группировке ячеек по категориям с помощью атрибута `axis`. Имена категорий могут быть любыми – просто используйте имена согласованно, как в следующем примере, где для всех ячеек «**intensity**» значение атрибута `axis` равно `intensities`, для всех ячеек «**fluctuation**» значение атрибута `axis` равно `fluctuations`, и для всех ячеек «**error**» значение атрибута `axis` равно `errormargins`.

Совет

Если ячейка принадлежит двум или большему числу заголовков, разделяйте идентификаторы (ID) заголовков пробелами, как в следующем примере:

<td headers="первыйзаголовок второйзаголовок третийзаголовок четвертыйзаголовок">

```
<table border="1" rules="rows">

  <caption>Table 4: Intensities of the Kenneth and Bradley
Frequencies and Fluctuations in Beta Transmission</caption>

  <col width="150">
  <colgroup span="3" width="150" align="center">

  <thead>
    <tr>
      <th> </th>
      <th id="intensity" abbr="Intensity"
axis="intensities">Intensity</th>
      <th id="fluctuation" abbr="Fluctuation" axis="fluctua-
tions">Fluctuations in Beta Transmission</th>
      <th id="error" abbr="Error" axis="errormargins">Margin of
Error</th>
    </tr>
  </thead>

  <tbody>
    <tr>
      <th id="kenneth" rowspan="5" abbr="Kenneth">Kenneth
Frequency</th>
      <td headers="intensity kenneth" axis="intensities">1</td>
      <td headers="fluctuation kenneth"
axis="fluctuations">90</td>
      <td headers="error kenneth" axis="errormargins">6%</td>
    </tr>
    <tr>
      <td headers="intensity kenneth" axis="intensities">2</td>
```

```
      <td headers="fluctuation kenneth" axis="fluctua-
tions">128</td>
      <td headers="error kenneth" axis="errormargins">35%</td>
    </tr>
    <tr>
      <td headers="intensity kenneth" axis="intensities">3</td>
      <td headers="fluctuation kenneth" axis="fluctua-
tions">2726</td>
      <td headers="error kenneth" axis="errormargins">32%</td>
    </tr>
    <tr>
      <td headers="intensity kenneth" axis="intensities">4</td>
      <td headers="fluctuation kenneth" axis="fluctua-
tions">263443</td>
      <td headers="error kenneth" axis="errormargins">32%</td>
    </tr>
    <tr>
      <td headers="intensity kenneth" axis="intensities">5</td>
      <td headers="fluctuation kenneth"
axis="fluctuations">2</td>
      <td headers="error kenneth" axis="errormargins">98%</td>
    </tr>
  </tbody>

  <tbody>
    <tr>
      <th id="bradley" rowspan="5" abbr="Bradley">Bradley
Frequency</th>
      <td headers="intensity bradley" axis="intensities">1</td>
      <td headers="fluctuation bradley"
axis="fluctuations">4</td>
      <td headers="error bradley" axis="errormargins">73%</td>
    </tr>
    <tr>
      <td headers="intensity bradley" axis="intensities">2</td>
      <td headers="fluctuation bradley"
axis="fluctuations">45</td>
      <td headers="error bradley" axis="errormargins">35%</td>
```

```
    </tr>
    <tr>
      <td headers="intensity bradley" axis="intensities">3</td>
      <td headers="fluctuation bradley" axis="fluctua-
tions">356</td>
      <td headers="error bradley" axis="errormargins">12%</td>
    </tr>
    <tr>
      <td headers="intensity bradley" axis="intensities">4</td>
      <td headers="fluctuation bradley"
axis="fluctuations">32</td>
      <td headers="error bradley" axis="errormargins">11%</td>
    </tr>
    <tr>
      <td headers="intensity bradley" axis="intensities">5</td>
      <td headers="fluctuation bradley"
axis="fluctuations">12</td>
      <td headers="error bradley" axis="errormargins">2%</td>
    </tr>
  </tbody>

</table>
```

Вид этой таблицы показан на рисунке 56.11.

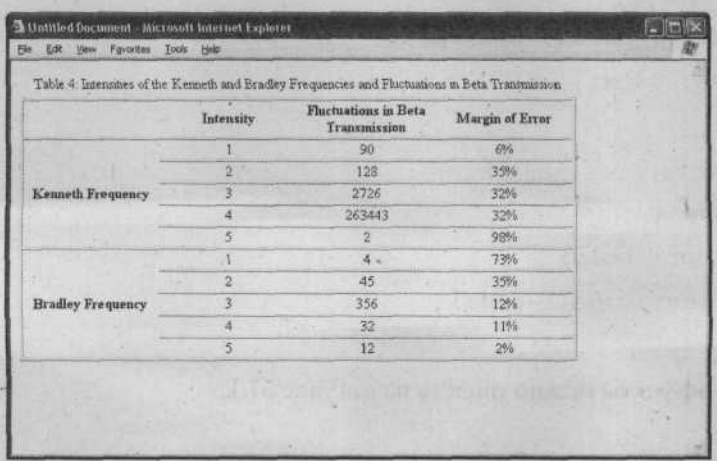


Рис. 56.11. Группирование ячеек по категориям атрибута `axis` *не влияет на внешний вид таблицы данных в браузере, но для будущих устройств чтения с экрана имена атрибута* `axis` *позволят лучше понимать компоновку таблицы*

ГЛАВА 57.
Использование псевдоэлементов

В каскадных таблицах стилей **псевдоэлементы** — это специальные дополнения к селекторам HTML-тегов. Они идентифицируют определенные разделы текста внутри HTML-элемента. В настоящее время существуют два псевдоэлемента: `first-line` и `first-letter`. Их можно применять к любому текстовому тегу или любым тегам, содержащим текст, таким, как `<table>` или `<div>`.

Любая старая таблица стилей могла бы идентифицировать общий стиль абзаца следующим образом:

```
p {
   font-family: Arial, Helvetica, sans-serif;
   font-size: 14px;
}
```

Определение

В каскадных таблицах стилей ***псевдоэлемент*** *(pseudo-element) — это специальное дополнение к селектору HTML-тега, которое идентифицирует определенную часть текста внутри элемента.*

Только в очень изощренных таблицах стилей добавляются псевдоэлементы, которые указывают браузеру, что надо делать с первой строкой абзаца:

```
p {
   font-family: Arial, Helvetica, sans-serif;
   font-size: 14px;
}

p:first-line {
   font-weight: bold;
   font-variant: small-caps;
}
```

Полученные эффекты можно увидеть на рисунке 57.1.

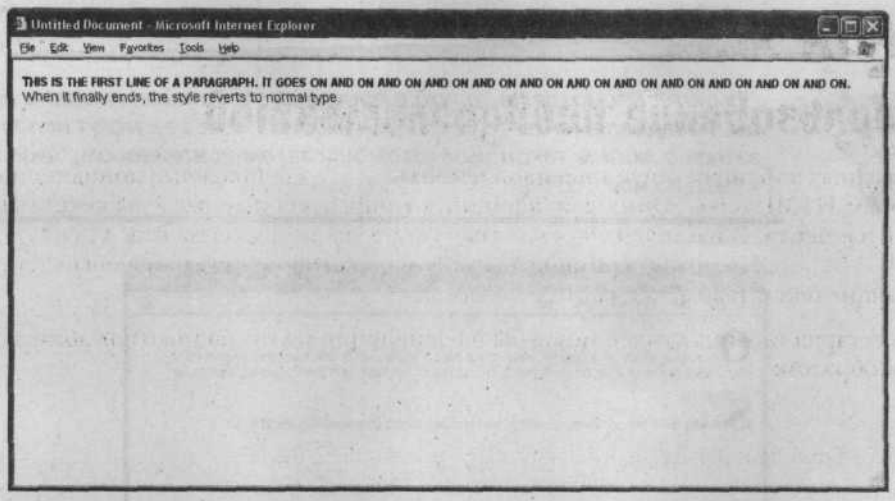


Рис. 57.1. Псевдоэлемент `first-line` идентифицирует специальный стиль для первой строки текстового элемента, как в этом абзаце

Часто задаваемые вопросы

Как узнать, где кончается первая строка?

Ширина окна браузера определяет, где заканчивается первая строка. Поэтому дизайнер теряет некоторую степень контроля над тем, какой текст обрабатывается псевдоэлементом `first-line`. Если вы хотите сохранить полный контроль над своим текстом, воздержитесь от использования псевдоэлемента `first-line` и вместо него заключите в теги `<span>` ту строку текста, которую хотите изменить. Затем примените специальное определение стиля к атрибуту `style` тега `<span>`.

Псевдоэлемент `first-letter` функционирует во многом так же, только он воздействует на первую букву текста, а не на всю первую строку:

```
p {
   font-family: Arial, Helvetica, sans-serif;
   font-size: 14px;
}

p:first-letter {
   font-family: "Times New Roman", Times, serif;
   font-weight: bold;
   font-size: 48px;
}
```

На рисунке 57.2 показан результат действия этих правил стиля.

Совет

Помните: несмотря на то, что в примерах этой главы показаны абзацы, псевдоэлементы `first-line` и `first-letter` могут использоваться с любым текстовым тегом или тегом-контейнером, таким как `<table>` или `<div>`.

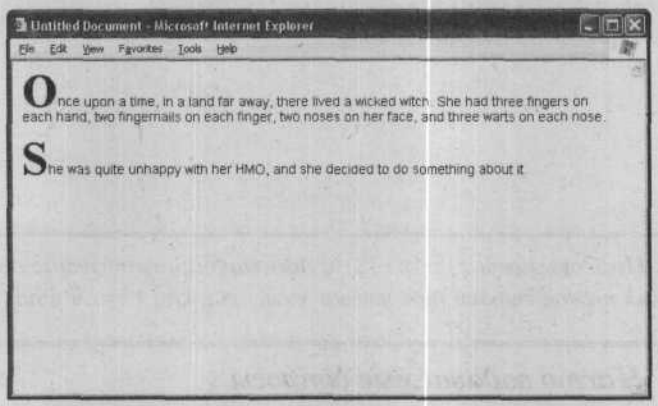


Рис. 57.2. Псевдоэлемент `first-letter` определяет специальный стиль для первой буквы текстового элемента

Инструментарий

Таблица стилей для буквиц

В издательском деле буквица – это заглавная буква увеличенного размера, которая обычно располагается в начале главы. Однако по сравнению с увеличенной заглавной буквой, показанной на рисунке 57.2, буквица отличается одной особенностью: буквица выходит вниз за базовую линию текста, и первые три или порядка этого строк текста в абзаце обтекают ее, как на рисунке 57.3.

Определение

Буквица *(drop cap) — это заглавная буква увеличенного размера, обычно находящаяся в самом начале главы. Она выходит вниз за базовую линию текста, и соседние строки текста обтекают ее.*

Определение

Нижние выносные элементы *(descenders) — это части букв, которые выходят вниз под базовую линию строки текста, как, например, ножка строчной буквы* ***p****.*

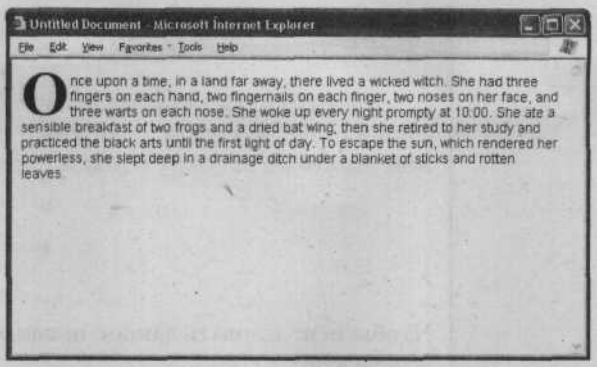


Рис. 57.3. Буквица – это заглавная буква увеличенного размера, которая выходит вниз за базовую линию текста. Следующая пара строк текста обтекает буквицу

В этом «Инструментарии» приводится правило стиля для буквиц. Просто скопируйте и вставьте его в существующую таблицу стилей. Не бойтесь изменять размер шрифта. Возможно, потребуется подогнать стиль для конкретной страницы в зависимости от гарнитуры и размера шрифта основного текста. По высоте буквица должна занимать от трех до четырех строк текста. Заглавные буквы с **нижними выносными элементами** или частями, расположенными ниже базовой линии, подобно заглавной букве **Q** в гарнитуре с засечками (serif), занимают четыре строки текста, тогда как обычные заглавные буквы занимают три строки.

В правило стиля можно добавлять также другие CSS-атрибуты, такие, как фоновый цвет и дополнительное пустое пространство.

```
p.dropcap:first-letter {
    font-family: Georgia, "Times New Roman",
Times, serif;
    font-weight: bold;
    font-size: 63px;

/* Данный размер шрифта выбран в предположении, что
размер основного текста равен 14 пикселам и что бук-
вица занимает от трех до четырех строк текста. */

  float: left;

/* Атрибут float является секретным оружием, кото-
рое заставляет работать буквицу. Когда элемент пла-
вает, браузер окружает его соседними элементами.
Элемент может свободно перемещаться влево или впра-
во. */
```

```
  padding-right: 2px;


/* Добавление небольшого пустого пространства
справа от буквицы помогает избежать соприкоснове-
ний буквицы с окружающими строками текста. */


}
```

Чтобы использовать данное правило стиля, задайте для любого абзаца класс dropcap:

```
  <p class="dropcap">Once upon a time, in a land
far away....</p>
```

Поскольку область применения класса dropcap в этом «Инструментарии» ограничивается абзацами, буквицу нельзя добавить в заголовок, таблицу или тег любого другого типа. Если необходимо, чтобы класс dropcap работал с любым подходящим элементом, включая абзацы, просто удалите тег абзаца из селектора, как в следующей строке кода:

```
  .dropcap:first-line {
```

Часто задаваемые вопросы

Эта буквица выглядит нелепо в браузере** Netscape**. В чем проблема?

Стили буквицы, включая тот, что приведен в этом «Инструментарии», как правило, выглядят лучше в Internet Explorer, чем в Netscape, поскольку браузером Netscape добавляется чрезмерная величина пустого пространства над верхом буквицы. Чтобы улучшить вид данного стиля в Netscape, добавьте в определение стиля фоновый цвет. В результате буквица будет помещена внутрь цветного прямоугольника, и пустое место над буквой будет меньше бросаться в глаза.

Для улучшения внешнего вида можно воспользоваться очевидным искусственным приемом, состоящим в добавлении в определение стиля отрицательных значений верхнего и нижнего полей:

```
margin-top: -12px;
margin-bottom: -12px;
```

В этом случае буквица сдвигается вверх, но одновременно добавляются дополнительные 12 пикселов пустого пространства перед началом абзаца.

Отрицательные поля, по крайней мере, не влияют на внешний вид буквицы в Internet Explorer, поэтому данное решение проблемы пригодно для обоих браузеров.

ГЛАВА 58.
Определение стилей идентификаторов

Селектор идентификатора (ID) в каскадных таблицах стилей во многом подобен классу, относящемуся к определенному тегу, в том отношении, что он является привилегированным клубом. В действительности селектор идентификатора настолько привилегирован, что применяется к одному и только к одному элементу на странице. Никакие два элемента не должны принадлежать одному и тому же идентификатору, даже если элементы относятся к одному и тому же типу: абзацам, изображениям, ссылкам или тому, что имеется в распоряжении.

Для определения селектора идентификатора применяется специальный синтаксис – а именно, знак порядкового номера (#), за которым следует имя идентификатора:

```
#barney {
  font-weight: bold;
  color: #FF00FF;
}
```

Определение

Селектор идентификатора *(ID selector) — это селектор, который применяется к одному и только к одному элементу на странице.*

Затем задайте атрибут `id` тега, к которому требуется применить стиль:

```
<p id="barney">Этот абзац имеет идентификатор ID of "barney"</p>
```

Идентификаторы (ID) удобно использовать в качестве заменителей правил стиля, которые находятся в атрибуте `style` HTML-тега. Вспомните, что помещение CSS-кода в атрибут `style` лучше всего действует для одноразовых стилей. Что такое идентификатор (ID), если не одноразовый стиль? Поэтому без всяких сомнений код, подобный следующему:

```
<p style="font-weight: bold; color: #FF0000;">Alert!</p>
```

можно заменять чем-нибудь, похожим на следующий блок кода:

```
<style type="text/css">

  #alert {
    font-weight: bold;
    color: #FF0000;
  }
```

```
</style>
<body>
  <p id="alert">Warning!</p>

</body>
```

Как можно видеть на рисунке 58.1, оба метода приводят к одному и тому же результату. Если вы эстет в области кодирования, метод идентификаторов может быть для вас чуть более ясным, более привлекательным и по смыслу более близким к каскадным таблицам стилей. Возможно, вы предпочтете использовать идентификатор вместо вставки правила стиля в атрибут неуклюжего старого HTML-тега, хотя требуемого результата позволяет достичь любой из методов.

Часто задаваемые вопросы

Что произойдет, если применить один и тот же идентификатор (ID) к нескольким элементам?

Вы не попадете в тюрьму, ваш компьютер не взорвется, и вы не разрушите Интернет. Браузер обрабатывает идентификаторы (ID) как классы, поэтому, если вы используете один и тот же идентификатор (ID) для нескольких элементов, все элементы отобразятся со стилем идентификатора (ID).

Однако вы создадите для себя всевозможные проблемы, когда настанет время добавить на страницу функции JavaScript, использующие метод `getElementByID`. *Если ваши идентификаторы (ID) не являются уникальными, функции не будут работать надлежащим образом.*

Поэтому, если вам требуется стиль класса, создайте стиль класса. Оставьте идентификаторы для специальных классов.

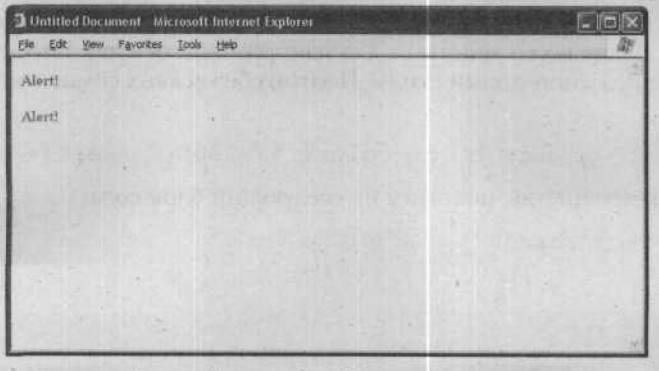


Рис. 58.1. Стиль идентификатора (ID) (внизу) работает точно так же, как атрибут `style` *HTML-тега (внизу), и определяет одноразовое правило стиля. Получаемые результаты идентичны*

ЧАСТЬ 5.
Ссылки

- Применение стилей к гиперссылкам
- Дискуссионная тема: в защиту подчеркивания ссылок
- Создание ролловерных гиперссылок
- Выбор надлежащего пути
- Изменение цвета одиночной гиперссылки
- Выбор подходящего текста для гиперссылки
- Создание экранных подсказок для гиперссылок
- Открытие нового окна браузера
- Использование именованных анкеров
- Создание ссылок электронной почты

ГЛАВА 59.
Применение стилей к гиперссылкам

Язык HTML предоставляет основной набор атрибутов для управления внешним видом гиперссылок на Web-странице. Эти атрибуты включаются в тег <body>:

```
<body link="#0000FF" vlink="#FF00FF" alink="FF0000">
```

Все атрибуты ссылки сведены в таблицу 59.1. Как можно видеть, HTML привязывает их к трем состояниям ссылки: непосещенное состояние, посещенное состояние и активное состояние. Для разделения трех состояний используется цвет. **Непосещенная ссылка** – это ссылка на страницу, о посещении которой в браузере нет сведений. По умолчанию цвет непосещенных ссылок – синий. **Посещенная** ссылка – это ссылка на страницу, сведения о посещении которой имеются в браузере. По умолчанию цвет посещенных ссылок – фиолетовый или ярко-красный. Наконец, активная ссылка – это ссылка, на которой щелкают мышью в текущий момент. В браузере Internet Explorer цвет активной ссылки отображается также в том случае, когда посетитель выделяет ссылку.

Определение

***Непосещенная ссылка** – это ссылка на страницу, о посещении которой в браузере нет сведений. **Посещенная** (visited) ссылка – это ссылка на страницу, сведения о посещении которой имеются в браузере. **Активная** (active) ссылка – это ссылка, на которой в текущий момент щелкают мышью или которая выделена.*

Табл. 59.1. HTML-атрибуты ссылки

Атрибут	Управляемое свойство
link	Цвет непосещенной ссылки
vlink	Цвет посещенной ссылки
alink	Цвет ссылки, на которой щелкают кнопкой мыши или которая выделена

Чтобы задать пользовательские цвета для состояний ссылки на странице, укажите их непосредственно с помощью атрибутов link, vlink и alink.

Трудно поверить, что Интернет приобрел такую популярность с таким ограниченным набором конструктивных возможностей для наиболее важного элемента на любой Web-странице. Имеется три цвета ссылки для трех ее состояний и это все. Однако возможности дизайна сильно расширяются при использовании для форматирования ссылок каскадных таблиц стилей (CSS) вместо атрибутов HTML.

На самом элементарном уровне каскадные таблицы стилей воспроизводят функции атрибутов link, vlink и alink с помощью псевдоклассов link, visited и active. В каскадных таблицах стилей **псевдокласс** – это специальное дополнение тега-анкера, которое сообщает браузеру, к какому состоянию ссылки применяется правило стиля.

В главе 57 рассматривались псевдоэлементы `first-line` и `first-letter`. Псевдоклассы сходны с ними, но в настоящее время они применяются только к тегу-анкеру.

Часто задаваемые вопросы

Как выделить гиперссылку?

Несомненно, все знают, как щелкать на гиперссылке, но не всякий знает, как выделить гиперссылку, в основном, потому, что большинство людей не представляют, что в браузерах Internet Explorer и Netscape можно перемещаться по Web-сайтам без использования мыши.

*Вот как это работает: вы циклически перемещаетесь по интерактивным элементам на странице, нажимая клавишу **Tab**, а после перехода на требуемый элемент нажимаете клавишу **Enter**, чтобы активизировать его. А так выделяется гиперссылка: нажимая клавишу **Tab**, переходите на гиперссылку, устанавливая на нее фокус, если выражаться на интернет-жаргоне. Браузер Internet Explorer изменяет внешний вид ссылки, окрашивая ее в цвет активной ссылки. Если еще раз нажать клавишу **Tab**, фокус переходит с гиперссылки и при этом возвращается ее цвет по умолчанию, соответствующий цвету либо посещенной, либо непосещенной ссылки.*

Хотя ссылки можно выделять и в Netscape, этот браузер не меняет цвет выделенной ссылки на цвет ее активного состояния.

Определение

***Псевдокласс** (pseudo-class) — это специальное дополнение для селектора тега-анкера, сообщающее браузеру, к какому состоянию ссылки применяется правило стиля.*

Следующий блок CSS-кода оказывает такой же эффект, как и HTML-атрибуты, приведенные в теге `<body>` в начале этой главы:

```
a:link {
   color: #0000FF;
}

a:visited {
   color: #FF00FF;
}

a:active {
```

```
  color: #FF0000;
}
```

Эти правила стилей представляют начальный продукт, который можно усовершенствовать. Как насчет выделения непосещенных и активных ссылок полужирным шрифтом и оставления посещенных ссылок с нормальным начертанием?

```
a:link {
  color: #0000FF;
  font-weight: bold;
}

a:visited {
  color: #FF00FF;
  font-weight: normal;
}

a:active {
  color: #FF0000;
  font-weight: bold;
}
```

Эти стили показаны на рисунке 59.1. Обратите внимание, как полужирное начертание помогает выделить на Web-странице непосещенные ссылки, при этом посещенные ссылки кажутся менее важными.

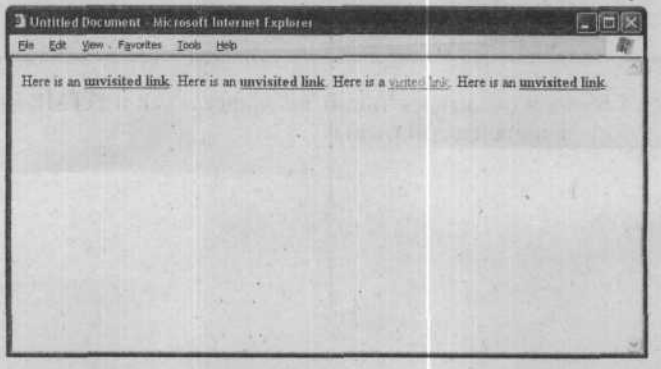


Рис. 59.1. Используйте каскадные таблицы стилей для изменения других свойств ссылок, помимо окрашивания разных состояний ссылки в разные цвета

На некоторых Web-сайтах вокруг гиперссылок добавляется едва различимый фоновый цвет, создающий эффект блока, как на рисунке 59.2.

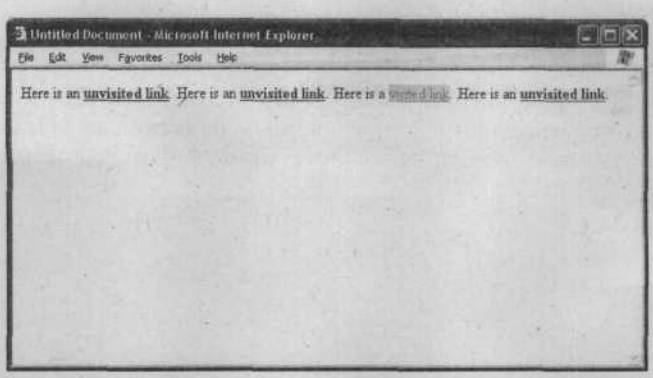


Рис. 59.2. Добавьте атрибут `background-color` *в правило стиля для ссылки или состояния ссылки, чтобы создать эффект блока*

Листинг 59.1. Исходный код страницы, вид которой в браузере показан на рисунке 59.2

```
a {
  font-weight: bold;
}

/* В этих правилах стилей каждому состоянию ссылки соответствует
свой фоновый цвет. */

a:link {
  color: #0000FF;
  background-color: #CCFFFF;
}

a:visited {
  color: #FF00FF;
  font-weight: normal;
  background-color: #CCCCFF;
}

a:active {
  color: #FF0000;
  background-color: #FFFFCC;
}
```

Можно также выключить используемое по умолчанию подчеркивание, как на рисунке 59.3.

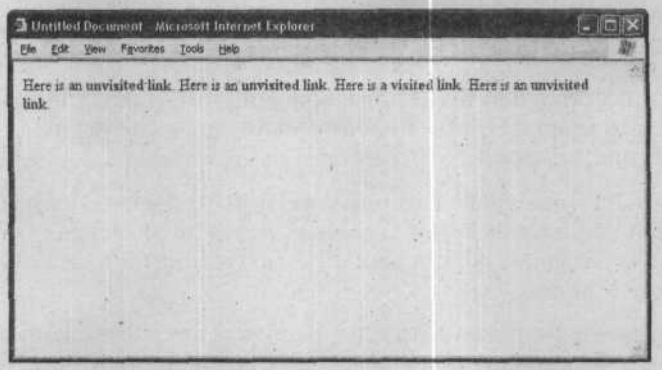


Рис. 59.3. Подчеркивание гиперссылок отключено

Листинг 59.2. Исходный код страницы, вид которой в браузере показан на рисунке 59.3

```
a {
  font-weight: bold;
  text-decoration: none;
}

/* Для выделения гиперссылки необходима какая-нибудь визуальная
подсказка, отличная от цвета, поэтому в отсутствие подчеркивания в
этих правилах стилей используется полужирное начертание для всех
трех состояний ссылок. */

a:link {
  color: #0000FF;
  background-color: #CCFFFF;
}

a:visited {
  color: #FF00FF;
  background-color: #CCCCFF;
}

a:active {
  color: #FF0000;
  background-color: #FFFFCC;
}
```

Дневник: В защиту подчеркивания ссылок

Каскадные таблицы стилей великолепны, когда они работают. Они позволяют делать всякие интересные вещи с HTML-элементами. Можно даже удалить святая святых – подчеркивание гиперссылок.

На самом первом из когда-либо созданных Web-сайтов была гиперссылка, и для этой гиперссылки использовалось подчеркивание, так что существует давняя традиция применения этого элемента оформления. Но подчеркивание желательно сохранить не только по этой причине.

Очевидно одно: большинство гиперссылок использует подчеркивание. В действительности, именно по этой причине удаление подчеркивания так притягательно. Дизайнеры любят отказываться от подчеркивания, чтобы сделать вид своих Web-сайтов уникальным, что составляет важную часть работы дизайнера. Интернет был бы менее интересным, если бы на всех Web-сайтах использовалось одни и те же правила оформления.

Существует лишь одна проблема. Те, кто бороздит Интернет, настолько привыкли к вездесущему подчеркиванию, что они щелкают на любом элементе, содержащем подчеркивание, даже если этот элемент не является гиперссылкой. Именно поэтому следует проявлять осторожность и не подчеркивать элементы, которые не являются гиперссылками. И по этой же причине следует подчеркивать элементы, которые являются гиперссылками.

Интернет – это, прежде всего, скорость. Люди, которым нравится Интернет, ценят возможность быстрого поиска информации. Вместе с тем, занимающиеся Web-серфингом скользят не только по Web-сайтом, но и по информации. Они в действительности не читают текст, пока не найдут нужные им сведения.

Как дизайнеру, вам следует помнить об этих фактах. Вы хотите, чтобы посетители могли перемещаться по вашему Web-сайту быстро и непринужденно, поэтому вам необходимо что-нибудь для привлечения их внимания к гиперссылкам.

Подчеркивание работает идеально. Все знают, что оно указывает на гиперссылку. Подчеркивание легко различимо и посетителям не требуется задумываться, что оно обозначает. Зачем портить отличную вещь?

Кроме того, неподчеркнутые гиперссылки заставляют посетителей задумываться о том, какие соглашения по стилям используются на вашем Web-сайте. Используется ли для этого полужирное начертание? Курсив? Слова в цветных прямоугольниках? Конечно, это очевидно для вас, но вы – дизайнер. Знания и опыт путешествий по Интернету обычно не одно и то же. Другими словами, посетители не могут применить свои знания о том, как работает Интернет, для просмотра вашего Web-сайта, это означает, что они не сумеют перемещаться по вашему ресурсу на высоких скоростях, к которым они привыкли.

Во что бы то ни стало, разрабатывайте состояния ссылок с помощью каскадных таблиц стилей. Добавьте полужирное начертание. Добавьте курсив. Измените гарнитуру. Измените размер шрифта. Но прежде чем отказаться от подчеркивания, хорошенько подумайте.

ГЛАВА 60.
Создание ролловерных гиперссылок

В главе 59 рассматривались три псевдокласса каскадных таблиц стилей – `link`, `visited` и `active` – и вы обратили внимание, что они соответствуют трем состояниям ссылок в HTML. В каскадных таблицах стилей добавляется четвертое состояние ссылки: **выделенное состояние** (hover state), которое описывает ссылку, когда посетитель наводит на нее указатель мыши. Еще одно состояние ссылки означает наличие еще одного псевдокласса каскадных таблиц стилей, а еще один псевдокласс означает еще одну возможность для создания особого стиля.

Определение

***Выделенное состояние** (hover state) описывает ссылку, когда посетитель наводит на нее указатель мыши. Используйте псевдокласс* **hover**, *чтобы создать ролловерные стили для гиперссылок в каскадных таблицах стилей.*

Следующая таблица стилей включает правило для выделенного состояния:

```
a {
  font-weight: bold;
  text-decoration: none;
}
a:hover {
  text-decoration: underline;
}
```

Как видно на рисунке 60.1, ссылки отображаются с полужирным начертанием, без подчеркивания, используемого по умолчанию, благодаря правилу общего стиля, которое управляет всеми тегами-анкерами. Но когда посетитель наводить указатель на ссылку, вступает в действие правило специального стиля, которое применяется только к ссылкам, на которые наведен указатель, и появляется подчеркивание.

Обратите внимание, что нет необходимости включать полужирное начертание в правило стиля для выделенного состояния. Все, что относится в общем стиле к тегам-анкерам, автоматически применяется к псевдоклассам. Таким образом, когда пишутся правила стилей для состояний ссылок, просто включите определения для дополнительного форматирования, которое хотите применить, включая несоответствия или изменения для основных правил (подобно переходу от нормального начертания в обычных ссылках к полужирному начертанию в ролловерных ссылках). Определения стилей, которые остаются одинаковыми для разных состояний ссылок, не требуется повторять.

Стиль `hover` может включать любую каскадную таблицу стилей, которая применяется к тексту. Следовательно, в ролловерном состоянии ссылка может увеличиваться в размере или по толщине начертания, может изменяться ее стиль, цвет; элементы украшения и так далее. Просто внимательно относитесь к большим изменениям стандартной ссылки. Если выделенная ссылка не помещается в то же пространство, что и стандартная ссылка, браузер перерисовывает всю Web-страницу при наведении

указателя на ссылку, как на рисунке 60.2, что может смутить посетителей или даже вынудить их отказаться от дальнейшего просмотра Web-сайта.

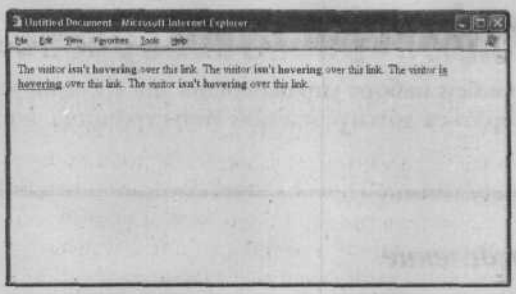


Рис. 60.1. С помощью псевдокласса `hover` *можно создать ссылки, которые изменяют свой внешний вид при наведении на них указателя мыши*

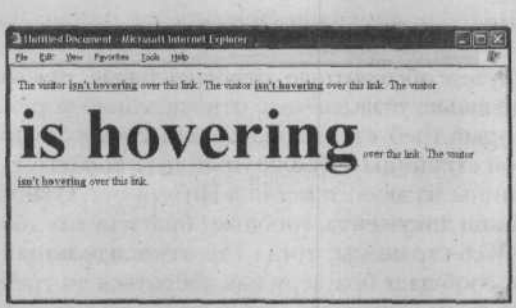


Рис. 60.2. Сравните этот рисунок с рисунком 60.1, и вы увидите, что браузер перерисовывает страницу, чтобы вместить ссылку с этим чрезмерно большим ролловерным стилем

Совет

Некоторые дизайнеры любят создавать ролловеры ссылок, которые по умолчанию отображаются подчеркнутыми, а при наведении указателя на ссылки подчеркивание исчезает. Хотя данная стратегия не лишена смысла, поскольку подчеркивание, используемое по умолчанию, помогает посетителям уяснить, какие элементы на странице являются гиперссылками, однако не чувствуется, что ролловер переходит из низкоэнергетического состояния в высокоэнергетическое состояние. С психологической точки зрения именно этого ожидает посетитель. Когда подчеркивание пропадает, ссылка выглядит как закрывающаяся — она переходит из состояния с высокой энергией в состояние с низкой энергией. Некоторые посетители могут сделать вывод, что ссылки так и не являются активными.

Если вы не хотите удалить подчеркивание, используемое по умолчанию, то и не удаляйте подчеркивание в ролловерном состоянии. Найдите другой способ выделить ролловер, например, с помощью перехода от нормального к полужирному начертанию или посредством добавления фонового цвета. Каким бы не был выбранный стиль, ролловерная ссылка должна выглядеть как активизирующаяся, а не как закрывающаяся.

ГЛАВА 61.
Выбор надлежащего пути

Путь гиперссылки подобен набору управляющих инструкций для браузера. Он сообщает браузеру, как добраться до загружаемой Web-страницы, когда посетитель щелкает на ссылке.

Определение

***Путь** (path) гиперссылки сообщает браузеру, как добраться до страницы, которая загружается при щелчке пользователем на ссылке.*

Существует три типа путей: абсолютные, относительные, указываемые относительно документа, и относительные, указываемые относительно корня. Выбор типа пути зависит от способа, который требуется использовать для сообщения браузеру о том, как добраться до связанной страницы. **Абсолютный** путь сообщает браузеру, как добраться до требуемой страницы из любого места в Интернете. **Относительный путь, указываемый относительно документа,** сообщает браузеру как добраться до требуемого документа с текущей Web-страницы, тогда как **относительный путь, указываемый относительно корня,** сообщает браузеру, как добраться до требуемой страницы с текущего Web-сайта. У каждого типа пути есть свои достоинства и недостатки, которые рассматриваются в данной главе.

Определение

***Абсолютный** (absolute) путь сообщает браузеру, как добраться до требуемой страницы из любого места в Интернете. **Относительный путь, указываемый относительно документа** (document-relative), сообщает браузеру, как добраться до требуемой страницы с текущей Web-страницы. **Относительный путь, указываемый относительно корня** (root-relative), сообщает браузеру, как добраться до требуемой страницы с текущего Web-сайта.*

Использование абсолютных путей

Абсолютный путь предоставляет полный путь URL (Uniform Resource Locator – унифицированный указатель информационного ресурса) требуемой Web-страницы, включая префикс «**http://www**», как в следующем примере:

```
<a href="http://www.pearson.com/">Абсолютный путь</a>
```

Другими словами, абсолютный путь полностью совпадает с путем URL, вводимым в браузере в поле адреса. Чтобы перейти на требуемую страницу, не надо вводить другой URL, когда используется другой компьютер или когда вы находитесь на другом Web-сайте. Именно это свойство делает путь абсолютным – он всегда одинаков, независимо от того, в каком месте Интернета просматривает ресурсы браузер.

Совет

Используйте абсолютные пути для ***внешних ссылок*** *(external links) или для ссылок на страницы, которые не принадлежат вашему Web-сайту. На практике для этих целей следует обязательно использовать абсолютные пути. Относительные пути, указываемые относительно документа (document-relative) или корня (root-relative), не работают с внешними страницами.*

Поскольку абсолютный путь всегда одинаков, его не надо обновлять в исходном коде, если изменяется положение страницы в иерархии вашего Web-сайта. Страницу с абсолютной ссылкой можно перемещать в любое место на своем Web-сайте и ссылка всегда будет соединять с требуемой страницей до тех пор, пока путь URL в атрибуте `href` будет точным.

Определение

Внутренняя ссылка *(internal link) – это ссылка на страницу, находящуюся в пределах вашего Web-сайта, тогда как* ***внешняя ссылка*** *(external link) – это ссылка на страницу за пределами вашего Web-сайта.*

Но поскольку необходимо приводить полный путь URL при каждой записи абсолютного пути, эти пути могут быть громоздкими для **внутренних ссылок** (internal links) или для ссылок, которые указывают на страницы собственного Web-сайта. Для внутренних ссылок, как правило, лучше подходят относительные ссылки, указываемые относительно документа или корня. Кроме того, чтобы абсолютные пути работали, для вашего компьютера должно быть установлено активное подключение к Интернету. Если открыть окно браузера в автономном режиме для тестирования собственного Web-сайта, то при щелчке на ссылке с абсолютным путем браузер откроет подключение к Интернету.

Часто задаваемые вопросы

Можно ли использовать для внешних ссылок относительные пути, указываемые относительно документа?

Нет. Относительные пути, указываемые относительно документа, предназначены исключительно для внутренних ссылок.

Использование относительных путей, указываемых относительно документа

В относительном пути, указываемом относительно документа, вы сообщаете браузеру как добраться до требуемой страницы с текущей Web-страницы. Это делается путем пошагового перемещения браузера по иерархии сайта.

Следует тогда отметить, что для написания действующего относительного пути, указываемого относительно документа, необходимо знать иерархию своего сайта. Возьмем следующую структуру сайта:

- Домашняя страница
    - Фильмы
        - Приключения
        - Комедии
        - Драмы
        - Романтические
        - Детективы
        - Ужасы
        - Научная фантастика
    - Музыка
        - Популярная
        - Джаз
        - Классика
    - Игры
        - Компьютерные
        - Консольные

Допустим, что текущей страницей является домашняя страница. Относительный путь, указываемый относительно документа, в применении к странице «Фильмы» относительно домашней страницы будет выглядеть следующим образом:

```
<a href="фильмы/фильмы.htm">С домашней страницы на страницу "Фильмы"</a>
```

тогда как для страницы «Приключения» в категории «Фильмы» относительный путь, указываемый относительно документа, будет следующим:

```
<a href="фильмы/приключения/приключения.htm">С домашней страницы
на страницу "Приключения"</a>
```

Однако если текущая страница изменяется, изменяется и относительный путь. Для перехода на страницу «Приключения» со страницы «Фильмы», а не с домашней страницы, путь будет выглядеть следующим образом:

```
<a href="приключения/приключения.htm">Со страницы "Фильмы" на
страницу "Приключения"</a>
```

Как можно видеть, нет необходимости указывать префикс «**http://www**» или имя домена для своего сайта. Все это подразумевается в относительном пути, указываемом относительно документа. Все, что необходимо сделать, это направить браузер к надлежащему каталогу и файлу относительно текущей страницы.

Все объяснения, приведенные выше, касались переходов вглубь структуры сайта, а что если необходимо возвращаться в обратном направлении по иерархии Web-страниц. Как записать относительный путь, указываемый относительно документа, для страницы «Фильмы» относительно страницы «Фантастика»? Потребуется некоторая техническая терминология:

```
<a href="..фильмы.htm">Со страницы "Научная фантастика" на страни-
цу "Фильмы"</a>
```

Две точки (..) являются компьютерным кодом действия «Перейти на один уровень вверх в структуре сайта». Таким образом, если браузер находится на странице «Научная фантастика» и видит две точки в относительном пути, указываемом относительно документа, он понимает, что необходимо вернуться на один уровень назад по иерархии, чтобы найти файл **фильмы.htm**. Подобным образом, если необходимо перейти со страницы «Научная фантастика» назад на несколько уровней в направлении к домашней странице, потребуется несколько наборов двоеточий:

```
<a href="../../index.htm">Со страницы "Научная фантастика" на до-
машнюю страницу</a>
```

Данный относительный путь предписывает браузеру перейти на два уровня вверх в структуре сайта, чтобы найти файл **index.htm**.

В одном и том же относительном пути, указываемом относительно документа, можно комбинировать перемещения вверх и вниз по структуре сайта. Скажем, требуется перейти со страницы «Ужасы» в категории «Фильмы» на страницу «Классика» в категории «Музыка». Относительный путь, указываемый относительно документа, мог бы выглядеть следующим образом:

```
<a href="../Музыка/Классика/классика.htm">Со страницы "Научная фан-
тастика" на страницу "Классика"</a>
```

Если описывать действия на обычном языке, то браузер переходит в иерархии на один уровень вверх, чтобы найти папку музыки. Он ищет внутри папки музыки папку классики, и внутри папки классики ищет файл **классика.htm**.

Совет

При перемещении страниц в структуре сайта инструменты управления сайтами такие, как Macromedia Dreamweaver, Adobe GoLive и Microsoft FrontPage автоматически обновляют относительные пути, указываемые относительно документа.

Полезное свойство относительных путей, указываемых относительно документа, состоит в том, что они работают в автономном режиме также хорошо, как и в интерактивном режиме. Другими словами, можно тестировать сайт автономно в браузере без

необходимости подключаться к Интернету, при этом все ссылки с относительными путями, указываемыми относительно документа, действуют в полном объеме. Основной недостаток относительных путей, указываемых относительно документов – при перемещении страницы, содержащей ссылку на другой уровень сайта, ссылка больше не будет работать. Помните, относительный путь, указываемый относительно документа, направляет браузер вверх и вниз по иерархии сайта. Если страница не находится в прежнем положении в иерархии, шаги перемещения по структуре сайта не будут совпадать с прежней последовательностью шагов. Эта проблема отсутствует для абсолютных путей, поскольку абсолютные пути не зависят от текущего положения браузера.

Использование относительных путей, указываемых относительно корня

Относительные пути, указываемые относительно корня (root-relative paths), сходны с относительными путями, указываемыми относительно документа (document-relative paths), в том, что они используются только для внутренних ссылок и не применяются для внешних ссылок. Однако относительные пути, указываемые относительно корня, больше похожи на абсолютные пути, хотя в них не указывается ни префикс «**http://www**», ни имя домена для сайта:

```
<a href="/index.htm">Относительный путь, указываемый относительно корня</a>
```

Относительные пути, указываемые относительно корня, начинаются с левой косой черты (/), и они описывают положение целевой страницы относительно **корня** (root) или самого верхнего уровня вашего сайта. Поэтому относительный путь, указываемый относительно корня, всегда одинаков, как и абсолютный путь. Путь не изменяется в зависимости от текущего положения браузера, т.к. не изменяется сам корень сайта.

Совет

***Корень** (root) сайта – это самый верхний уровень сайта.*

Взяв в качестве примера иерархию сайта из предыдущего раздела данной главы, получим для страницы «Классика» следующий относительный путь, указываемый относительно корня:

```
<a href="/музыка/классика/классика.htm">Из корня к странице "Классика"</a>
```

Абсолютный путь для этой страницы выглядит точно также, только добавляются префикс «**http://www**» и имя домена для сайта.

При поверхностном взгляде относительные пути, указываемые относительно корня, кажутся более удобными для внутренних ссылок, чем относительные пути, указываемые относительно документов. Страницу с относительным путем, указываемым относительно корня, можно перемещать на другой уровень в структуре сайта, и ссылка будет по-прежнему работать, поскольку путь всегда начинается от корня сайта. Однако относительные пути, указываемые относительно корня, имеют один роковой недостаток – они работают только на Web-сервере. Ссылки с относительными путями, указываемыми относительно корня, невозможно тестировать не только автономно, но и с установленным подключением к Интернету (если только на вашем персональном компьютере не установлен Web-сервер). Для тестирования ссылок требуется загрузить страницы на Web-сервер. Это связано не только с дополнительными усилиями, но также ставит разработчика в неудобное положение. Обычно вы загружаете свои страницы на сервер после того, как они протестированы и отлажены. Если требуется загружать страницы и для тестирования, посетители вашего сайта могут наткнуться на страницы, которые еще не готовы и находятся в процессе тестирования и отладки.

Часто задаваемые вопросы

Итак, я не могу тестировать в автономном режиме относительные ссылки, указываемые относительно корня?

В действительности вы можете, но для этого требуются специальное программное обеспечение и оборудование. Профессиональные разработчики Web-сайтов обходят проблему с относительными путями, указываемыми относительно корня, одним из двух способов: устанавливая Web-сервер на своем персональном компьютере или загружая страницы на вспомогательный сервер (staging server), являющийся личным Web-сервером, предназначенным для отладки и тестирования.

ГЛАВА 62.
Изменение цвета одиночной гиперссылки

В обычном коде HTML не существует способа для изменения цвета одиночной гиперссылки на странице. В теге <body> доступны атрибуты link, vlink и alink, которые определяют цвета в равной степени для всех ссылок, и у вас, например, нет возможности сделать стандартные непосещенные ссылки синими, а одну конкретную непосещенную ссылку красной.

Это означает, что у вас нет такой возможности, если вы не используете каскадные таблицы стилей, которые предоставляют пару разных способов для изменения цвета одиночной ссылки.

Совет

Будьте осторожны, изменяя цвета отдельных ссылок слишком часто. Посетители вашего сайта полагаются на цвета ссылок, которые позволяют им судить о том, какие страницы вашего сайта уже просмотрены ими. Любое отклонение от образца цветовой схемы стандартных ссылок связано с риском запутать аудиторию своего сайта.

Однако применение этих методов полностью правомерно для изменения цвета ссылок, запускающих сценарии, с целью обособления их от обычных гиперссылок. Поступая таким образом, вы акцентируете внимание на различии между ссылками, запускающими сценарии, и ссылками, предназначенными для загрузки страницы, что помогает сделать сайт более удобным для использования.

Изменение цвета, используемого по умолчанию для всех состояний ссылок

Если требуется изменить цвет, в который ссылка окрашена по умолчанию, и если необходимо, чтобы этот же цвет применялся для ссылки независимо от ее состояния, просто добавьте правило стиля в атрибут style тега-анкера ссылки:

```
<a href="../special.htm" style="color: #FF0000;">Смотрите экстренный выпуск</a>
```

В этом примере гиперссылка всегда отображается красной, независимо от состояния ссылки и используемых по умолчанию цветов ссылки.

В альтернативном варианте, если то же самое форматирование применяется к нескольким ссылкам на странице, можно создать стиль класса для управления специальным цветом гиперссылки:

```
<style type="text/css">
  a.special {
```

```
    color: #FF0000;
  }

</style>

<body>

  <a class="special" href="../special1.htm">Смотрите экстренный выпуск</a>
  <a class="special" href="../special2.htm">Смотрите дополнительный экстренный выпуск</a>

</body>
```

Теперь любые теги-анкеры, принадлежащие классу `special`, имеют одинаковые свойства цвета независимо от состояния ссылки и стандартных цветов ссылок.

Изменение стандартного цвета для отдельных состояний ссылок

Если требуется менять специальный цвет ссылки в зависимости от состояния ссылки, атрибут `style` тега-анкера не поможет сделать это, поскольку необходим доступ к псевдоклассам тега-анкера. Нет проблем – просто создайте специальный стиль класса подобный следующему:

```
<style type="text/css">

  a.special:link {
    color: #FF0000;
  }

  a.special:visited {
    color: #CC0000;
  }

  a.special:active {
    color: #990000;
```

```
  }

  a.special:hover {
    color: #660000;
  }

</style>

<body>

  <a class="special" href="../special1.htm">Смотрите экстренный выпуск</a>
  <a class="special" href="../special2.htm">Смотрите дополнительный экстренный выпуск</a>

</body>
```

В этом примере представлены четыре правила стилей для четырех CSS-состояний ссылки, каждое со своим, отличным от других, цветом. Но все правила стилей принадлежат одному классу – `special` – таким образом, когда тег-анкер присоединяется к классу, применяются все четыре правила стилей. Состояние ссылки определяет, какой цвет будет использоваться независимо от стандартного цвета для этого состояния.

Совет

Нет необходимости приводить правила стилей для всех четырех состояний ссылки. Достаточно определить состояния, которые применяются к ссылке. Если для специальной ссылки необходимо использовать отдельные цвета для ***посещенного*** *(visited) и* ***непосещенного*** *(unvisited) состояний, а для* ***активного*** *(active) и* ***выделенного*** *(hover) состояний отдельные цвета не требуются, создайте стили только для состояний* `link` *и* `visited`*. Стандартные ссылки на странице могут иметь активное и выделенное состояния, но они не будут относиться к специальному классу.*

ГЛАВА 63.
Выбор подходящего текста для гиперссылки

Создание гиперссылки не представляет трудностей. В пару тегов-анкеров заключается фрагмент текста, на котором должны будут щелкать кнопкой мыши посетители.

Разработка гиперссылки — это полностью другое дело. Действительно, какой текст должен заключаться в теги-анкеры?

Первое практическое правило для любой текстовой гиперссылки заключается в следующем: ссылка должна сообщать посетителю, что следует ожидать от щелчка на ней. Это означает, что текст ссылки должен подсказывать, куда ведет ссылка и какие сведения посетитель получит, проследовав по предлагаемому пути.

Иногда выбор очень прост. Если существует список ссылок, которые функционируют как часть системы навигации на сайте, очевидно, что ссылки должны быть полными именами категорий навигации, как на рисунке 63.1. Данное текстовое описание удовлетворяет требованиям первого практического правила для гиперссылок, поскольку оно полностью сообщает посетителю, куда ведут ссылки.

Листинг 63.1. Исходный код страницы, вид которой в браузере показан на рисунке 63.1

```
<p>
  <a href="../index.htm">HOME</a> |
  <a href="../products.htm">PRODUCTS</a> |
  <a href="../services.htm">SERVICES</a> |
  <a href="../aboutus.htm">ABOUT US</a> |
</p>
```

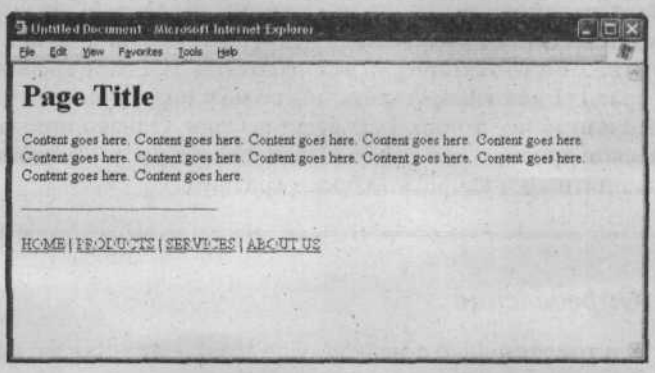


Рис. 63.1. Когда существует список ссылок, которые функционируют как часть системы навигации на сайте, заключите в теги-анкеры каждый из вариантов навигации

Выбор текста для ссылок также не составит труда, если имеется список ссылок на боковой панели или в специальном разделе на странице, как на рисунке 63.2. Здесь не возникает вопросов, куда введут ссылки. Текст ссылок сам сообщает об их назначении.

Листинг 63.2. Исходный код страницы, вид которой в браузере показан на рисунке 63.2

```
<p><a href="http://external_site01">External Site 1</a></p>
<p><a href="http://external_site02">External Site 2</a></p>
<p><a href="http://external_site03">External Site 3</a></p>
<p><a href="http://external_site04">External Site 4</a></p>
```

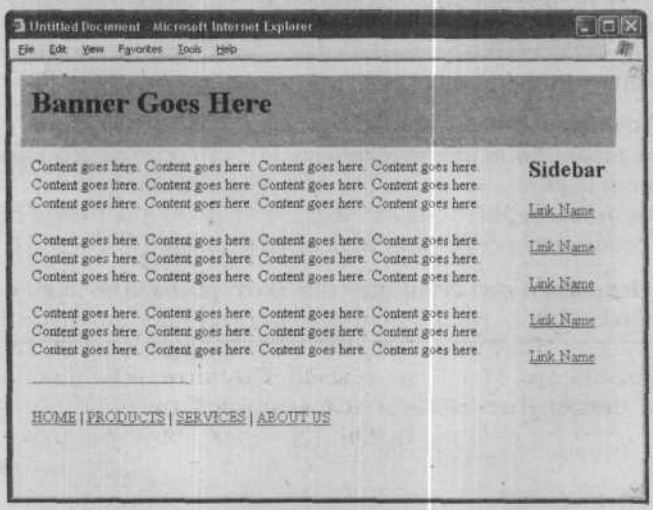


Рис. 63.2. Для ссылок на боковых панелях не возникает вопросов о том, какой текст заключать в теги-анкеры

Но что происходит, если встречаются **внутристрочные ссылки** или ссылки, которые отображаются внутри основного текста на странице. Вряд ли будет хорошим решением создание ссылки из целого абзаца, хотя некоторые дизайнеры иногда делают это, как на странице 63.3. Чисто технически эта стратегия отвечает требованиям первого практического правила для гиперссылок, поскольку в конечном итоге посетителям сообщается более или менее точно, куда ведут ссылки. Однако при этом нарушается второе практическое правило для гиперссылок, состоящее в следующем: текст ссылки должен быть понятным и как можно более кратким.

Определение

Внутристрочные ссылки *(inline links) — это ссылки, расположенные в основном тексте на странице.*

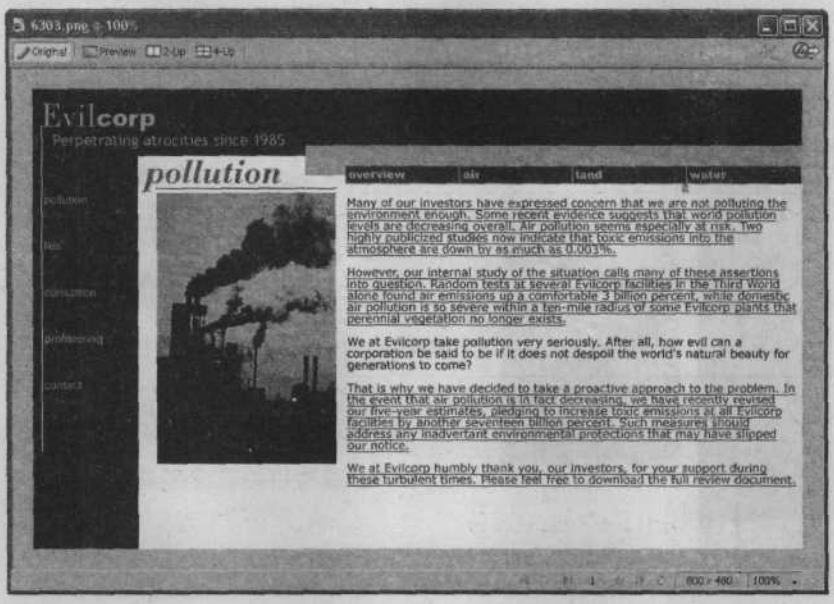


Рис. 63.3. Здесь целые абзацы служат гиперссылками. Возникает ли у вас желание щелкнуть на них. Конечно же нет. У вас нет времени для этого

Почему понятным и кратким? Потому что неясность и многословность противоречат сути Интернета, иными словами, они «тормозят» посетителей. Посетители вашего сайта вынуждены будут внимательно вчитываться в текст, как правило, несколько раз, чтобы понять, куда ведут ссылки. Помните, ваши посетители, возможно, величайшие почитатели печатного слова после Стивена Кинга (Stephen King), но они не читают текст в Интернете. Они скользят. Они сканируют. Они надеются найти нужную информацию и попросту не смогут это сделать, если будут вникать во все подряд.

Итак, как сделать ссылку понятной и краткой? Ключ к решению этой задачи – формирование текста ссылки из контекста. Это означает, что если посетители вашего сайта не читают какой-либо соседний текст (что вероятнее всего), они по-прежнему должны иметь возможность понять, куда ведут ссылки.

Возьмем рисунок 63.4. Текст ссылки стал более понятным, и он сообщает о том, какие сведения следует ожидать на другом конце ссылки, и при этом не требуется читать весь абзац. Но текст ссылки все еще слишком многословен. Слишком много текста бросается в глаза со страницы. По-прежнему необходимо читать текст большого объема.

Дизайн Web-страницы на рисунке 63.5 стал еще лучше. Ссылки достаточно кратки и они сообщают, куда по ним можно попасть. Даже если не читать соседний текст, ясно, что первая ссылка ведет к информации по каким-то исследованиям. Очевидно, что вторая ссылка указывает на сведения о собственном исследовании. Понятно, что по третьей ссылке выполняется переход к сведениям о пятилетних оценках, и не возникает сомнений, что последняя ссылка направляет к краткому обзору документа.

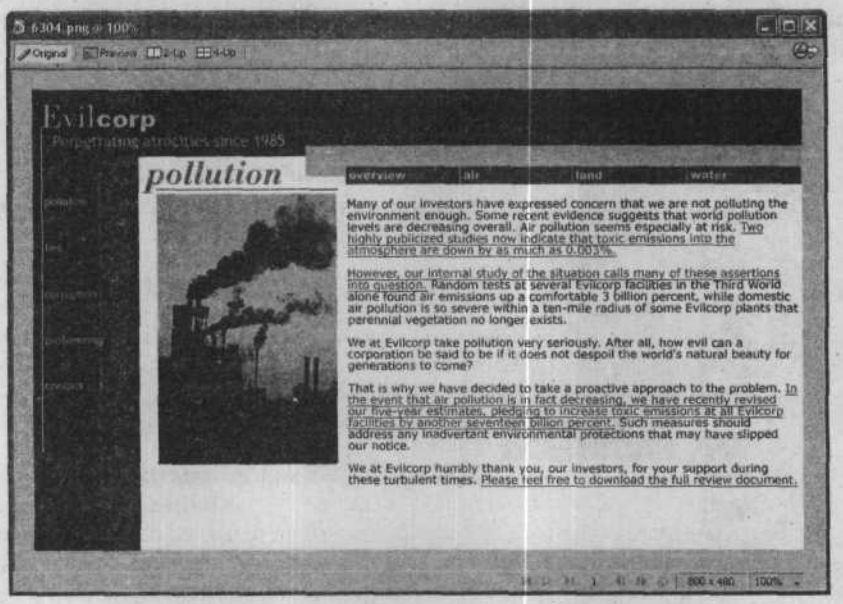


Рис. 63.4. Ссылки стали лучше, более сфокусированными. Но текст ссылок по-прежнему слишком многословен

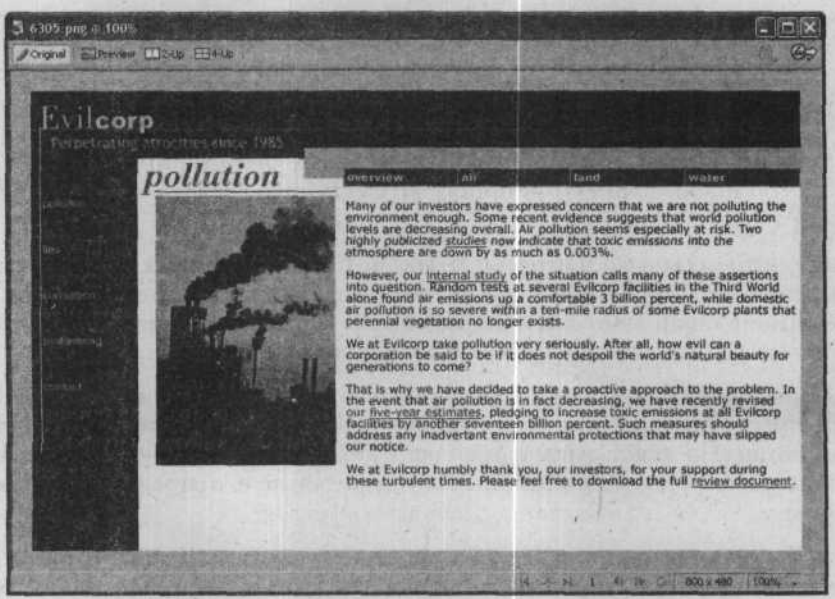


Рис. 63.5. Ссылки стали еще лучше. Они кратки и сообщают посетителю, к каким сведениям можно перейти, следуя по ссылкам

Обратите внимание, что текст первой ссылки не должен растолковывать, какие виды исследований присутствуют на связанной странице. Ваши пользователи могут быть нетерпеливыми, но они неглупые люди. Им уже известно, что они находятся на странице «**Pollution**» (загрязнение окружающей среды). Поставьте себя на их место. Если

бы вы попробовали догадаться, какие виды исследований ожидают вас на другом конце ссылки, то, по-видимому, пришли бы к выводу, что это исследования о загрязнении окружающей среды; и, как оказывается, были бы абсолютно правы.

Часто задаваемые вопросы

А как насчет знаков препинания? Являются ли они частью текста гиперссылки?

Текст внутристрочной ссылки обычно не включает начальные и конечные знаки препинания такие, как круглые скобки, точки или запятые. Этот общий стиль полностью противоположен правилам печатных изданий, где выделенные слова и фразы, как правило, включают начальные и конечные знаки препинания.

Обратите также внимание, что вторая ссылка читается как «**internal study**» (собственное исследование), тогда как первая ссылка — «**studies**» (исследования). С точки зрения краткости, можно было бы предположить, что во второй ссылке следовало бы опустить слово «**internal**» и оставить только «**study**». В некоторых случаях вы, возможно, были бы правы. Но здесь слово «**study**» слишком похоже на слово «**studies**» первой ссылки. Фраза «**internal study**» является достаточно краткой и резко контрастирует с первой ссылкой, помогая различать эти две ссылки. В свою очередь, посетители ожидают, что ссылки ведут на разные страницы, что они и делают. Слова «**studies**» и «**study**» не так уж сильно различаются между собой, поэтому некоторые посетители могут предположить, что они указывают на одну и ту же страницу.

Наконец, в последней ссылке не должно употребляться слово «**download**» (загрузка), которое может ожидаться некоторыми. Почему нет? Потому что **download** (загрузка) не имеет смысла, если исходить из контекста. Гиперссылка на странице «**Pollution**» (загрязнение окружающей среды), включающая **download** (загрузку), ни о чем не говорит, за исключением того, что при щелчке на ссылке будет что-то загружаться, вероятно, связанное с загрязнением окружающей среды. Но гиперссылка на странице «**Pollution**» (загрязнение окружающей среды), состоящая из фразы «**review document**», позволяет ясно представить, чего следует ожидать: краткого обзора документа, подготовленного, по-видимому, компанией Evilcorp, о случаях загрязнения.

И в заключение этой главы рассмотрим один более чем уместный вопрос: как насчет использования сверхпопулярной ссылки «**Click Here**» (Щелкните здесь)? Дизайн Web-страницы, показанной на рисунке 63.6, предоставляет четыре варианта этой непревзойденной классики, и не один из них не является хорошо разработанной гиперссылкой. Все ссылки за исключением первой являются краткими, но понятны ли они? Имеют ли они смысл, если исходить из контекста?

Есть один только способ проверить это: очень бегло просмотреть страницу. Сообщают ли ссылки, куда они ведут? Ясно ли из их текста, какие сведения следует ожидать на другом конце ссылки? Наиболее удачной является последняя ссылка, которая обещает **information** (информацию), но этот термин настолько широк, что практически бесполезен.

Ссылки «**Click Here**» (Щелкните здесь) читаются как команды. Они ничего не сообщают посетителям о том, что находится на связанной странице. Вместо этого они

указывают посетителям, что надо делать. Но посетители уже знают, как работают гиперссылки. И если они не знают, они смогут догадаться. В конце концов, кто научил вас, как надо щелкать?

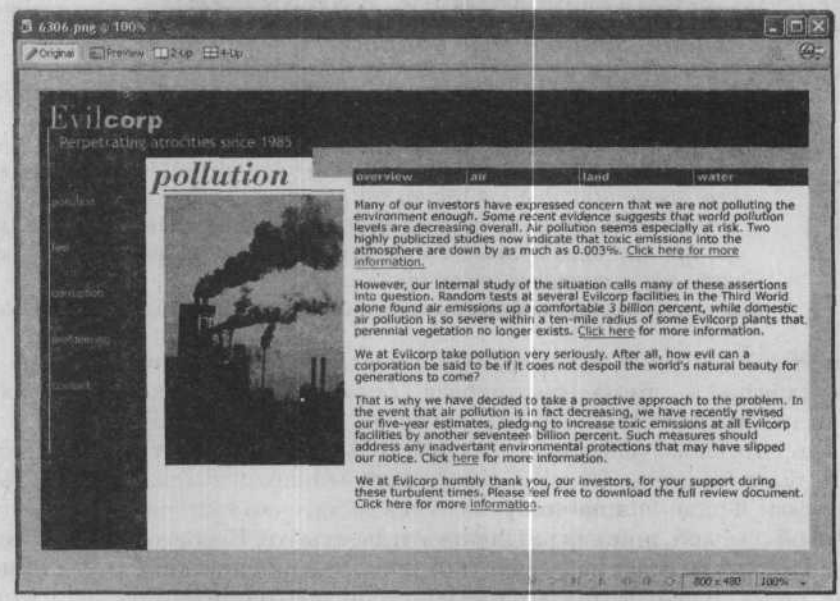


*Рис. 63.6. Не важно, что представлено несколько вариантов популярной ссылки «**Click Here**» (щелкните здесь), она здесь не работает. Эта ссылка не несет никакого смысла, если исходить из контекста*

Посетители вашего сайта знают, где щелкать. Им уже известно, как щелкать. От вас же они хотят узнать, почему надо щелкать, а об этом как раз ссылка «**Click Here**» (Щелкните здесь) и не сообщает.

ГЛАВА 64.
Создание всплывающих подсказок для гиперссылок

Время от времени, когда вы наводите указатель мыши на изображение, размещенное на Web-странице, появляется всплывающая подсказка, содержащая дополнительные сведения об изображении. Вам известно из главы 34, что вы, как дизайнер, полностью управляете содержанием всплывающей подсказки, поскольку браузер отображает то, что находится в атрибуте `alt` тега `img`:

```
<img src="horse.jpg" width="100" height="50" alt="Антикварная ка-
чающаяся лошадка из красного дерева с латунным колокольчиком на
шее">
```

Добавление текста в атрибут `alt` для изображения помогает сделать его более доступным для людей с дефектами зрения. Текст описывает содержание изображения, давая возможность любому, кто слышит текст из атрибута `alt`, представить изображение, не видя его.

Такой же принцип действует для текстовых гиперссылок, особенно для тех, которые находятся вне блока текста. Зрячие посетители просто обращают внимание на синий цвет текста ссылки и на подчеркивание, и они знают, что надо делать, не задумываясь об этом. Но когда вы отделяете список ссылок на боковой панели, строке навигации или на панели навигации внизу страницы, назначение этих ссылок может быть неясным для тех, кому недоступны достоинства синего цвета, подчеркивания или положения ссылок на странице относительно остального содержимого. Поэтому вы улучшите доступность своего сайта для разных категорий посетителей, если разъясните точно, что эти ссылки делают.

Тег-анкер не содержит атрибута `alt`, но для него есть атрибут `title`, который выполняет ту же функцию.

```
<p>

    <a href="../index.htm" title="Перейти на начальную страни-
цу">HOME</a> |

   <a href="../products/products.htm" title="Перейти на страницу
Продукты">PRODUCTS</a> |

  <a href="../services/services.htm" title="Перейти на страницу Ус-
луги">SERVICES</a> |

   <a href="../aboutus/aboutus.htm" title="Перейти на страницу О
нас">ABOUT US</a> |

</p>
```

Совет

Чтобы сделать всплывающую подсказку максимально доступной, не ограничивайтесь простым повторением текста ссылки. Это противоречит назначению подсказки. Лучше сообщите посетителю, куда ведет ссылка. Другими словами, не указывайте для атрибута `title` *ссылки значение* ***Home Page*** *(Начальная страница). Лучшим вариантом будет* ***Go to home page*** *(Переход на начальную страницу).*

При наведении указателя мыши на ссылку текст, содержащийся в атрибуте `title`, отображается как всплывающая подсказка. Это приятный момент для зрячих посетителей. Более важно, что инструменты специального доступа, такие, как устройства чтения с экрана, включают атрибут `title` ссылки в свои выходные данные, поэтому люди с дефектами зрения будут ясно представлять, куда ведут ссылки.

Совет

В атрибуте `title` *ссылки не требуется сообщать* ***Click here*** *(Щелкните здесь), как, например, в предложении «****Click here to go to the Product page****» (Щелкните здесь, чтобы перейти на страницу Продукты). Щелканье понятно само собой. Просто сообщите «****Go to the Product page****» (Переход на страницу Продукты).*

ГЛАВА 65.
Открытие нового окна браузера

Когда ссылка указывает на внешний сайт, общепринято, чтобы при щелчке на ссылке открывалось новое окно браузера. Внешний сайт загружается в новое окно, при этом в исходном окне остается открытой страница с исходной ссылкой.

Совет

Помните, что если вы умеете что-нибудь, не обязательно это делать во всех возможных случаях. Открытие нового окна браузера подходит лучше всего для внешних ссылок и не уместно для внутренних ссылок. Если новое окно появляется для каждой страницы на вашем сайте, то посетители, скорее всего, охотно пополнят ряды ненавистников открывания новых окон.

Некоторые разработчики Web-сайтов насмехаются над этой стратегией. Это обычно те, кто ненавидит всплывающие окна с такой силой, что сам Фрейд (Freud) затруднился бы объяснить причины подобной ненависти, которая возможно связана с беспорядком на рабочем столе. Учитывая, что внешние сайты регулярно исчезают, изменяются их URL-адреса или на сайтах используются методы, которые отключают кнопку **Back** (Назад) браузера, открытие нового окна браузера не так уж и плохо, как может показаться на первый взгляд. На самом деле в некоторых кругах это считается любезностью. Вывод посетителям сообщения об ошибке «**File Not Found**» (Файл не найден) менее всего выглядит как ошибка, когда избавление от бесполезного сайта сводится к простому закрытию нового окна браузера.

Открытие нового окна браузера очень просто реализовать для любой ссылки. Просто добавьте атрибут `target` в тег-анкер и установите для этого атрибута значение `_blank`, как в следующем примере:

```
<a href="http://externalsite/" target="_blank">Внешний сайт</a>
```

ГЛАВА 66.
Использование именованных анкеров

Тег-анкер необычен тем, что выполняет две разные функции в HTML. С одной стороны, тег-анкер можно использовать для связи с другой страницей:

```
<a href="anotherpage.htm">Переход на другую страницу</a>
```

Определение

Именованный анкер *(named anchor) — это местоположение на Web-странице, к которому можно привязать ссылку.*

С другой стороны, тег-анкер можно применить для задания местоположения, или **именованного анкера**, к которому осуществляется привязка:

```
<a name="привязатьздесь"></a>
```

Способ функционирования тега-анкера зависит от его атрибутов. Когда тег-анкер содержит атрибут `href`, он действует как **источник** ссылки, или элемент, на котором посетитель щелкает, чтобы перейти на другую страницу. Но когда тег-анкер содержит атрибут `name`, он функционирует как **пункт назначения** ссылки, или местоположение, которое отображается в браузере после щелчка посетителя на ссылке.

Определение

Источник *(source) ссылки — это элемент, который посетитель щелкает для перехода на другую страницу.* ***Пункт назначения*** *(destination) ссылки — это страница, которая загружается в окно браузера.*

Задание именованных анкеров

Чтобы установить анкер на странице, просто поместите тег-анкер точно в том месте, куда должен переходить браузер. Не забудьте указать атрибут `name` вместо атрибута `href`, чтобы тег функционировал как пункт назначения, а не как источник.

Одно из наиболее распространенных применений этого метода — деление длинной Web-страницы на несколько разделов. Такое деление часто встречается на страницах вопросов и ответов. Вверху страницы содержатся часто задаваемые вопросы (FAQ — Frequently Asked Questions), которые являются анкерами `href`. Ответы на вопросы следуют дальше, каждый со своим тегом именованного анкера, как в следующем примере:

```
<a name="question01"></a>
```

```
<p><strong>What is the meaning of life?</strong></p>
<p>The meaning of life is money.</p>

<a name="question02"></a>

<p><strong>What is the meaning of money?</strong></p>
<p>The meaning of money is time.</p>

<a name="question03"></a>

<p><strong>What is the meaning of time?</strong></p>
<p>The meaning of time is life.</p>
```

Как показано на рисунке 66.1, сами именованные анкеры не отображаются в окне браузера. Они не являются видимыми элементами. Они просто предоставляют конкретное место, с которым может устанавливаться связь ссылки.

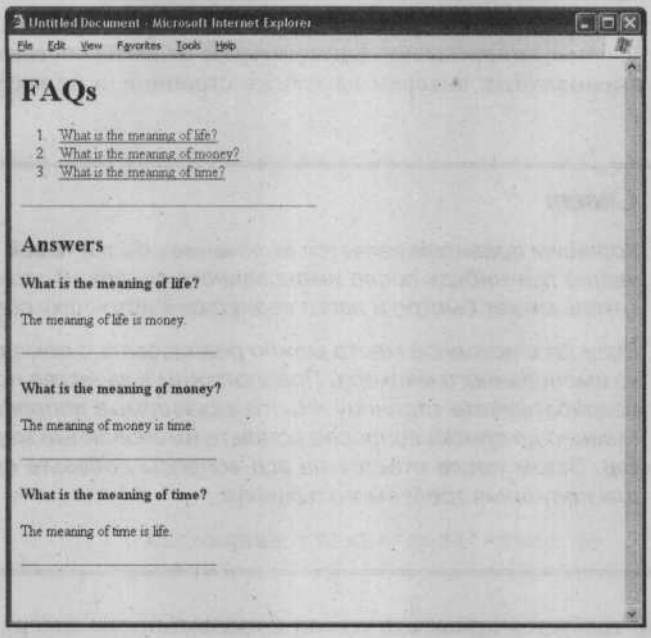


Рис. 66.1. Именованные анкеры не отображаются в окне браузера, но браузер знает об их существовании. Привяжите ссылку к одному из анкеров, и браузер перейдет точно в местоположение анкера

Привязка к именованным анкерам

Чтобы привязать ссылку к именованному анкеру, необходим еще один тег-анкер, только этот тег должен быть анкером-источником, а не анкером-пунктом назначения. В атрибуте `href` анкера-источника введите знак порядкового номера (#), сопровождаемый именем анкера, к которому требуется привязаться, как в следующем примере:

```
<ol>
   <li><a href="#question01">What is the meaning of life?</a></li>
   <li><a href="#question02">What is the meaning of money?</a></li>
   <li><a href="#question03">What is the meaning of time?</a></li>
</ol>
```

Часто задаваемые вопросы

Если именованные анкеры невидимы, то как браузер показывает, что посетитель перешел на один из них?

Обычно когда браузер переходит на именованный анкер, он прокручивает экран так, чтобы анкер находился вверху.

Эти теги-анкеры отображаются в браузере как обычные гиперссылки. Причина в том, что они *и есть* обычные гиперссылки. Единственное отличие состоит в том, что эти ссылки ведут к именованным анкерам на той же странице, а не вверх какой-нибудь другой страницы.

Совет

*Хорошим правилом является включение ссылки «**Back To Top**» (Назад вверх) где-нибудь после именованного анкера. В этом случае посетитель может быстро и легко вернуться к источнику ссылки.*

*Возврат в исходное место можно реализовать с помощью еще одного именованного маркера. Предположим в качестве примера, что вы разрабатываете страницу «Часто задаваемые вопросы». Непосредственно до списка вопросов вставьте именованный маркер с именем **top**. Затем после ответов на все вопросы добавьте следующий код для получения требуемого эффекта:*

```
<a href="#top">Назад вверх</a>
```

Что делать, если требуется привязать ссылку к именованному анкеру, который находится на какой-нибудь другой странице? Нет проблем. Просто добавьте код именованного анкера в конец пути, как в следующем коде:

```
<a href="../movies/movies.htm#newreleases">New Releases</a>
```

С помощью этого смешанного пути браузер находит страницу **movie.htm**, а затем переходит на данной странице к именованному анкеру `newreleases`.

ГЛАВА 67.
Создание ссылок электронной почты

Ссылка электронной почты в действительности не является ссылкой, хотя по внешнему виду похожа на нее и использует тег-анкер. Однако вместо направления браузера на другую страницу или на именованный анкер открывается пустое окно электронной почты. Используйте ссылки электронной почты на своем сайте как способ связи посетителей с вами.

Определение

При щелчке на ***ссылке электронной почты*** *(email link) открывается пустое окно электронной почты.*

Ссылка электронной почты выглядит подобно следующему примеру:

```
<a href="mailto:name@emailaddress">Пошлите мне письмо</a>
```

Тег-анкер создает ссылку, как и в случае обычной гиперссылки. Им используется также атрибут `href`, но значением этого атрибута является не путь. Вместо него присваивается специальное значение, называемое по очевидным причинам `mailto` (почта). Текст, который следует за `mailto`, является адресом получателя электронного сообщения. Таким образом, если хотите, чтобы электронное сообщение пришло на ваш личный электронный адрес, укажите в атрибуте `href` значение `mailto`, сопроводив его двоеточием (:) и своим полным электронным адресом.

Совет

Можно также указать тему сообщения (subject) и адрес для копии сообщения (CC):

```
<a
href="mailto:name@emailaddress?subject=Mail&cc=another@emailaddress">Пошлите мне письмо</a>
```

Однако темы и адреса для копий сообщений не работают с некоторыми приложениями электронной почты, поэтому не удивляйтесь, если они не будут работать и с вашими программами.

Если вы ищите предлог для изменения цвета одной гиперссылки на странице без воздействия на стандартные цвета (см. главу 62), то ссылка электронной почты будет отличным кандидатом. Окрашивание данной ссылки в другой цвет помогает выделить ее среди обычных гиперссылок на странице, тем самым подкрепляется идея о том, что ссылка электронной почты выполняет другую функцию.

Часто задаваемые вопросы

Я участвую в антиправительственном заговоре. Как ссылки электронной почты могут помочь мне?

Никто не говорит, что адрес электронной почты в ссылке электронной почты должен быть обязательно вашим. Если вам не терпится поозорничать, вы могли бы создать провокационную Web-страницу, добавить ссылку электронной почты и направить пачки пышущих ненавистью сообщений, которые, вероятно, вызовет ваша страничка, на адрес самой нелюбимой политической организации. Но считайте, что вы об этом не читали в данной главе.

ЧАСТЬ 6.
Формы

- Работа с элементами управления форм
- Применение стилей к элементам управления текстом
- Проверка достоверности входных данных форм
- Навигация с помощью клавиши Tab – изменение порядка переходов
- Работа с наборами полей
- Работа с метками

ГЛАВА 68.
Работа с элементами управления форм

Небольшие элементы, на которых вы щелкаете кнопками мыши и в которые вводите текст на форме, называются **элементами управления** и иногда также упоминаются как **поля форм** или **объекты форм**. Большая часть времени разработки эффективной формы уходит на определение различных элементов управления и выяснение того, какие функции выполняются каждым из элементов.

Определение

Элементы управления *(Widgets), также называемые* ***полями формы*** *(Form fields) или* ***объектами формы*** *(Form objects), - это небольшие фрагменты формы, на которых щелкают кнопками мыши и в которые вводят текст на форме.*

В таблице 68.1 приводится список HTML-элементов управления на тот случай, если вы забыли некоторые из них. В последующих разделах данной главы рассматривается каждый из этих элементов управления.

Табл. 68.1. HTML-элементы управления

Элемент управления	Назначение	Тег
Кнопка (общая)	Запуск функции	`<input type="button">`
Флажок	Выбор любых вариантов из числа доступных	`<input type="checkbox">`
Поле файла	Присоединение внешнего файла к данным формы	`<input type="file">`
Скрытое поле	Хранение скрытых данных	`<input type="hidden">`
Поле изображения	Отправка данных формы на Web-сервер	`<input type="image">`
Список	Выбор любых вариантов из числа доступных (но обычно только одного)	`<select size="x"></select>`
Меню	Выбор любых вариантов из числа доступных (но обычно только одного)	`<select></select>`
Поле пароля	Ввод пароля	`<input type="password">`
Переключатель	Выбор одного из нескольких вариантов	`<input type="radio">`
Кнопка «Сброс»	Возврат всех элементов управления в форме к их состояниям по умолчанию	`<input type="reset">`

Элемент управления	Назначение	Тег
Кнопка «Передача»	Отправка данных формы на Web-сервер	`<input type="submit">`
Текстовая область	Ввод нескольких строк текста	`<textarea></textarea>`
Текстовое поле	Ввод одиночной строки текста	`<input type="text">`

Работа с кнопками общего назначения

Кнопки общего назначения (см. рис. 68.1) – это элементы графического интерфейса, которые могут выбираться с помощью щелчка кнопкой мыши на них. Но когда посетитель щелкает на этих кнопках, то в отличие от кнопок отправки и сброса ничего не происходит. Необходимо специально запрограммировать эти кнопки на выполнение нужных вам функций, что обычно реализуется с помощью запуска функции JavaScript. Тег `input` кнопки принимает событие `onClick`, в котором указывается имя запускаемой функции.

Листинг 68.1. Исходный код страницы, вид которой в браузере показан на рисунке 68.1

```
<form>
   <input type=button" name="totalButton" value="Calculate Total"
onClick="doTotal();">
</form>
```

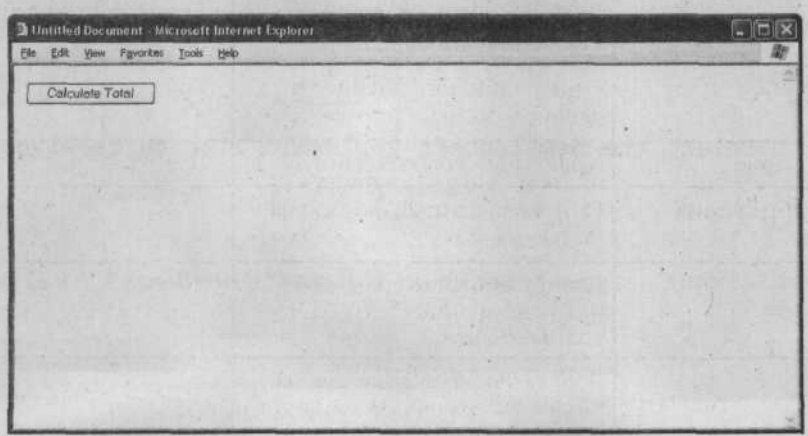


Рис. 68.1. Кнопка общего назначения активизируется с помощью щелчка кнопкой мыши на ней, но она ничего не выполняет до тех пор, пока вы не добавите в ее тег событие `onClick`

Атрибут `name` тега `input` для кнопки идентифицирует кнопку, а атрибут `value` содержит метку, которая отображается на кнопке. Значением события `onClick` является имя функции Javascript, которая запускается, когда пользователь щелкает на кнопке.

Работа с флажками

Флажки идеально подходят для относительно небольшого числа вариантов, как на рисунке 68.2, из которых посетитель может выбрать один, все, требуемое количество или ни одного. Не используйте флажки для ситуаций, где посетитель должен выбрать один и только один вариант – для этих случаев лучше подходят переключатели.

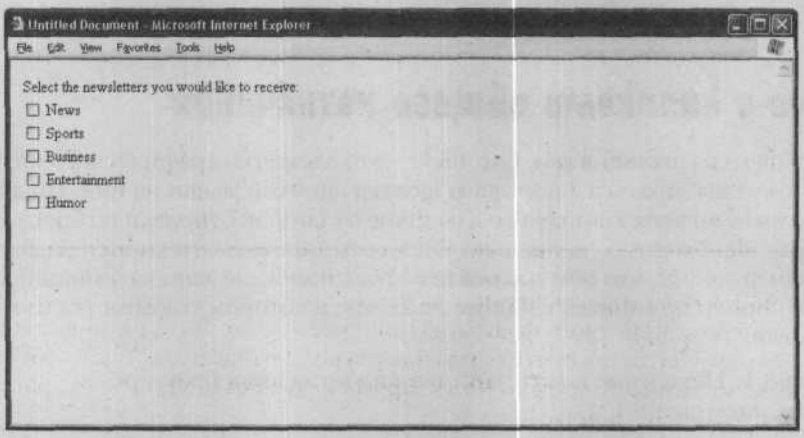


Рис. 68.2. Используйте флажки, когда имеется небольшое число вариантов, из которых посетитель может выбрать требуемое число вариантов (либо даже вообще не выбирать)

Листинг 68.2. Исходный код страницы, вид которой в браузере показан на рисунке 68.2

```
<form>
  <table>
    <tr>
      <td>Select the newsletters you would like to receive:</td>
    </tr>
    <tr>
      <td><input type="checkbox" name="newsCheck" value="yes"
>News</td>
    </tr>
    <tr>
      <td><input type="checkbox" name="sportsCheck" value="yes"
>Sports</td>
    </tr>
    <tr>
      <td><input type="checkbox" name="businessCheck" value="yes"
>Business</td>
```

```
    </tr>
    <tr>
      <td><input type="checkbox" name="entertainmentCheck" value=
"yes">Entertainment</td>
    </tr>
    <tr>
      <td><input type="checkbox" name="humorCheck" value="yes"
>Humor</td>
    </tr>
  </table>
</form>
```

Как и для кнопок общего назначения, для флажка атрибут `name` тега `input` предоставляет имя данного конкретного флажка. Атрибут `value` содержит данные, которые передаются формой, если посетитель устанавливает этот флажок. Таким образом, если посетитель устанавливает флажки **News**, **Sports** и **Business**, формой отправляются на Web-сервер следующие данные:

```
newsCheck=yes
sportsCheck=yes
businessCheck=yes
```

Значения не обязательно должны быть одинаковыми для всех флажков, как на рисунке 68.1. При необходимости можно передавать разные значения для каждого значка. Все зависит от того, что вы хотите послать на Web-сервер.

Часто задаваемые вопросы

Что происходит со сброшенными флажками?

Браузер игнорирует их. Он не включает их в отправляемые данные.

Чтобы флажок был предустановленным при загрузке формы, добавьте атрибут `checked` в тег `input` флажка:

```
<input type="checkbox" name="news" value="yes" checked>
```

Как видите, атрибуту `checked` не присваивается значение.

Работа с полями файлов

Поля файлов позволяют посетителям сайта прикреплять компьютерные файлы с их локальных компьютеров к форме отправки (Submission form). Таким способом

посетители могут загружать на Web-сервер такие объекты, как файлы изображений, звуковые файлы и текстовые документы. Как видно на рисунке 68.3, поле файла состоит из двух частей: самого поля и кнопки **Browse** (Обзор).

Листинг 68.3. Исходный код страницы, вид которой в браузере показан на рисунке 68.3

```
<form>
  <input type="file" name="ImageFile" size="50" maxlength="70">
</form>
```

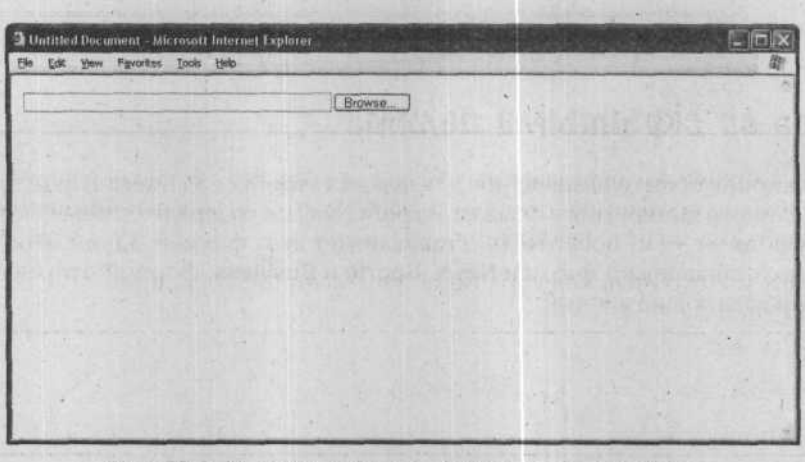


Рис. 68.3. Используйте поля файлов, когда требуется предоставить посетителям своего сайта возможность загрузки файлов с их локальных компьютеров на Web-сервер

Повторюсь еще раз: атрибут `name` идентифицирует поле файла. Атрибут `size` задает длину поля в знаках, а не в пикселах, тогда как атрибут `maxlength` определяет максимальное число знаков, которое может принять поле. В примере на рисунке 68.3 при значении атрибута `size`, равном 50, и значении атрибута `maxlength`, равном 70, посетитель может видеть 50 знаков в поле, и поле прекращает принимать вводимые данные после набора 70 знаков.

Часто задаваемые вопросы

Что происходит с файлом, загружаемым на удаленный компьютер?

Поля файлов могут быть сложны для обработки на сервере. Обычно формы такого типа поступают непосредственно в базу данных. Разработчик базы данных должен подготовить специальный столбец для принятия данных файла вместо текста или чисел, передаваемых в обычных случаях.

Для корректной работы внутри поля этого элемента управления должен находиться допустимый путь. Вот почему поле файла включает кнопку **Browse** (Обзор).

При щелчке на этой кнопке открывается диалог, предоставляющий посетителю возможность перехода к требуемому файлу, который будет загружаться на Web-сервер. После этого браузер автоматически вводит правильный путь.

Часто задаваемые вопросы

Мне не нравится кнопка Browse (Обзор). Могу я изменить ее?

Нет. С этим придется смириться. HTML не позволяет изменять название кнопки или делать ее невидимой.

Работа со скрытыми полями

Данные в скрытом поле передаются на Web-сервер вместе с остальной формой, но посетитель обычно не знает об этом, поскольку скрытое поле, как показано на рисунке 68.4, полностью невидимо на странице.

Листинг 68.4. Исходный код страницы, вид которой в браузере показан на рисунке 68.4

```
<form>
  <input type="hidden" name="referringPage" value="newsletter.htm">
</form>
```

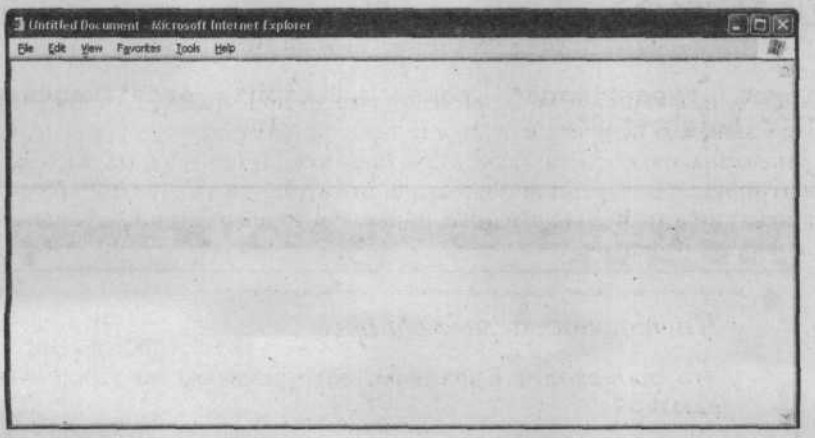


Рис. 68.4. Скрытое поле содержит данные формы, но посетитель не видит их, поскольку поле невидимо

Дизайнеры иногда используют скрытые поля для озорства, но обычно эти поля совсем безобидны. В примере на рисунке 68.4 скрытое поле содержит имя страницы, представляющей форму, которое полезно знать, если сайт содержит несколько форм более или менее одинаковых по виду.

Хотя посетитель не видит данные в скрытом поле, Web-сервер получает код, подобный следующему:

```
referringPage=newsletter.htm
```

Часто задаваемые вопросы

Могут ли посетители моего сайта изменять содержимое скрытого поля?

Непосредственно изменять не могут. Поскольку скрытые поля невидимы, посетитель не может редактировать или изменять их значения. Однако с помощью функций JavaScript достаточно просто обновить значение скрытого поля.

Работа с полями изображений

Поле изображения – это графическая замена стандартной HTML-кнопки **Submit** (Передать). Поле изображения является элементом графического интерфейса, управляемым щелчками кнопки мыши, и работает точно так же, как кнопка **Submit** (Передать), т.е. по щелчку на поле изображения данные формы пересылаются на Web-сервер. Преимущество замены кнопки **Submit** (Передать) на поле изображения заключается в том, что можно использовать любой файл изображения для передачи формы. Пример изображения показан на рисунке 68.5. При желании поставляемый с браузером элемент управления в виде стандартной и некрасивой кнопки можно заменить на приемлемое графическое изображение.

Листинг 68.5. Исходный код страницы, вид которой в браузере показан на рисунке 68.5

```
<form>
    <input type="image" name="goButton" src="images/go.gif"
width="20" height="20">
</form>
```

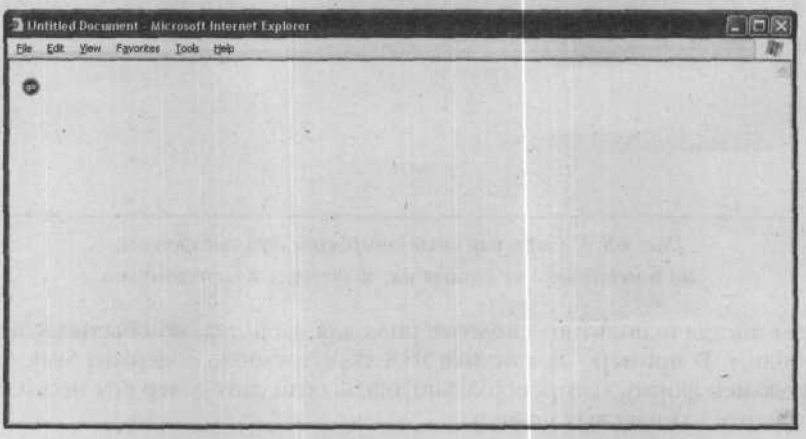


Рис. 68.5. Используйте поле изображения вместо стандартной кнопки ***Submit*** *(Передать)*

Атрибут scr содержит путь к графическому файлу поля изображения, а атрибуты width и height задают ширину и высоту изображения точно так же, как в теге img.

Замена кнопки Reset (Сброс)

Поля изображений всегда передают данные формы. Например, не существует способа изменения поля изображения так, чтобы оно функционировало как кнопка **Reset** (Сброс). Но пусть это вас не останавливает. Воспользуйтесь обычным тегом изображения. Заключите в теги-анкеры изображение, чтобы оно управлялось посредством щелчков кнопкой мыши, а затем добавьте простую функцию JavaScript для дублирования поведения кнопки **Reset** (сброс). Здесь приводится код полного сценария плюс образец ссылки:

```
<head>
  <script language="JavaScript">

    function doReset() {
      document.formname.reset();

/* Замените formname (имя формы) значением атрибута name тега
<form>. */

    }

  </script>
</head>

<!-- Далее следует  тело страницы. -->

<body>

    <a  href="javascript:doReset();"><img  src="images/reset.gif"
width="60" height="40" alt="Щелкните, чтобы установить форму в ис-
ходное состояние."></a>

</body>
```

Замена кнопки общего назначения

Подобным образом изображение можно использовать для запуска функции JavaScript и для замены элемента управления в виде кнопки общего назначения. Попробуйте воспользоваться кодом, похожим на следующий пример:

```
<head>
  <script language="JavaScript">

    function вашаФункция() {

/* Поместите здесь свою функцию JavaScript. */

    }

  </script>
</head>

<body>

  <a href="javascript:вашаФункция();"><img src="images/button.gif"
width="60" height="40" alt="Щелкните, чтобы запустить сценарий."></a>

</body>
```

Работа со списками

Список представляет несколько вариантов, из которых посетитель может делать свой выбор. Пример списка показан на рисунке 68.6. Если число доступных вариантов выбора достаточно большое, то следует отдать предпочтение использованию списков, а не переключателей или флажков, так как списки позволяют боле оптимально использовать площадь Web-страницы.

В списке атрибут `size` тега `<select>` определяет число отображаемых вариантов. Если список содержит больше вариантов, чем это число, браузером добавляется полоса прокрутки с правой стороны списка.

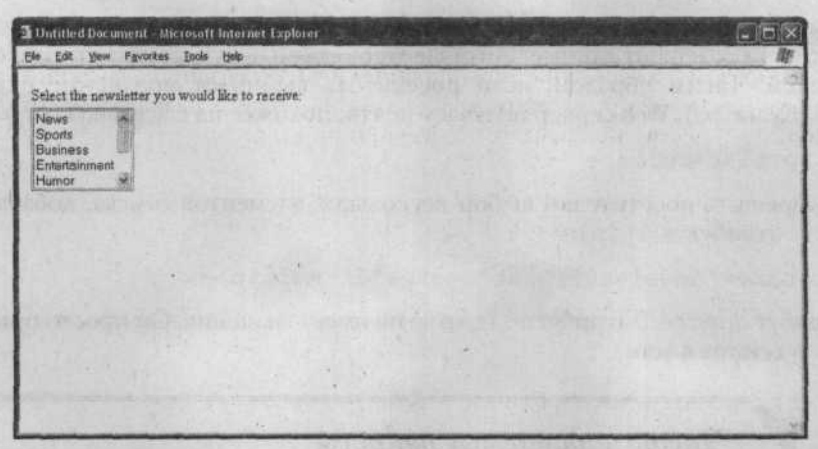


Рис. 68.6. Упорядочивание большого количества вариантов выбора в виде списка

Листинг 68.6. Исходный код страницы, вид которой в браузере показан на рисунке 68.6

```
<form>
  <table>
    <tr>
      <td>Select the newsletter you would like to receive:</td>
    </tr>
    <tr>
      <td><select name="newsletterList" size="5">
          <option value="news">News</option>
          <option value="sports">Sports</option>
          <option value="business">Business</option>
          <option value="entertainment">Entertainment</option>
          <option value="humor">Humor</option>
          <option value="style">Style</option>
          <option value="travel">Travel</option>
          <option value="science">Science</option>
          <option value="culture">Culture</option>
      </select></td>
    </tr>
  </table>
</form>
```

Каждый вариант в списке получает свой собственный тег <option>. Атрибут value тега <option> содержит данные, которые Web-сервер получает при отправке формы посетителем. Таким образом, если посетитель выбирает получение бюллетеня **«Culture»** (Культура), Web-сервер получает нечто, похожее на следующую строку кода:

```
newsletterList=culture
```

Чтобы разрешить посетителям выбор нескольких элементов списка, добавьте в тег <select> атрибут multiple:

```
<select name="newsletterList" size="5" multiple>
```

Как и атрибут checked, атрибут multiple не имеет значения. Он просто присутствует или отсутствует в теге.

Часто задаваемые вопросы

Как выбрать из списка несколько вариантов?

*Для выбора из списка нескольких вариантов необходимо держать нажатой клавишу **Ctrl** (Windows) или **Command** (Mac).*

Чтобы задать предварительный выбор элемента списка, добавьте в его тег атрибут selected:

```
<option value="news" selected>News</option>
```

Если разрабатываемый список допускает несколько вариантов выбора, возможно, вы захотите сделать некоторые варианты предварительно выбранными:

```
<option value="news" selected>News</option>
<option value="sports" selected>Sports</option>
<option value="business" selected>Business</option>
```

В противном случае браузер принимает один предварительно выбранный вариант и игнорирует остальные. По умолчанию Internet Explorer выбирает первый вариант, тогда как Netscape – последний.

Работа с меню

Меню во многом похоже на список. Код разметки практически такой же, причем меню, как и списки, используется для представления большого числа вариантов выбора. Основное отличие состоит в том, что доступные варианты отображаются в виде открывающегося списка, как на рисунке 68.7.

Листинг 68.7. Исходный код страницы, вид которой в браузере показан на рисунке 68.7

```
<form>
  <table>
    <tr>
```

```
      <td>Select the newsletter you would like to receive:</td>
    </tr>
    <tr>
      <td><select name="select">
          <option value="news" selected>News</option>
          <option value="sports">Sports</option>
          <option value="business">Business</option>
          <option value="entertainment">Entertainment</option>
          <option value="humor">Humor</option>
          <option value="style">Style</option>
          <option value="travel">Travel</option>
          <option value="science">Science</option>
          <option value="culture">Culture</option>
      </select></td>
    </tr>
  </table>
</form>
```

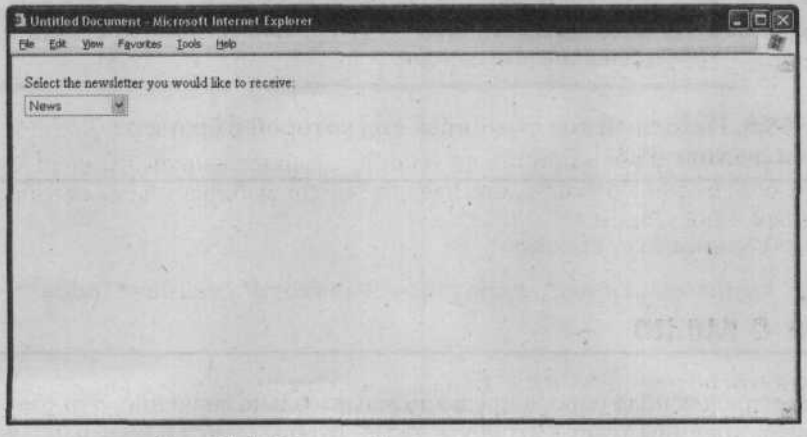


Рис. 68.7. Используйте меню вместо списка, если отдаете предпочтение компактному виду меню

Внимательно посмотрите на исходный код Web-страницы, показанной на рисунке 68.7, и вы увидите, что разница между меню и списком состоит в отсутствии атрибута `size` в теге `select`.

Совет

В меню всегда следует задавать один из вариантов предварительно выбранным. В противном случае меню будет отображаться пустым, пока посетитель не откроет список пунктов меню.

Во всем остальном атрибуты в меню идентичны по количеству и функциям, и данные, получаемые Web-сервером, выглядят практически одинаково. Кроме того, можно добавить атрибуты `multiple` и `selected` в теги `<select>` и `<option>`, присутствующие в коде меню, также, как и в списке, хотя, сделав это, вы превратите меню в список.

Работа с полями паролей

Поля паролей похожи на текстовые поля, только текст внутри них отображается в виде точек, как на рисунке 68.8. Обычно эти типы полей используются для принятия паролей или любых других входных данных, которые нежелательно выводить на экран в явном виде.

Часто задаваемые вопросы

Можно ли использовать поля паролей для других видов входных данных?

Безусловно. Поля паролей не ограничиваются вводом только паролей. В элемент управления этого типа также можно вводить личные идентификационные номера (PIN), ключи, номера счетов и другие конфиденциальные данные.

Листинг 68.8. Исходный код страницы, вид которой в браузере показан на рисунке 68.8

```
<form>
  <srong>Password:</strong>
  <input type="password" name="userPassword" value="ladda">
</form>
```

В этом примере для поля пароля предварительно задано значение. Это означает, что при загрузке страницы текст в атрибуте `value` отображается в поле пароля. Если вы не хотите, чтобы в поле пароля загружалось предустановленное значение, просто удалите этот атрибут:

```
<input type="password" name="userPassword">
```

Хотя браузер не отображает на экране фактическое значение поля пароля, при передаче формы Web-сервер получает текст, введенный в поле пароля, в виде следующих строк:

```
userPassword=ladda;
```

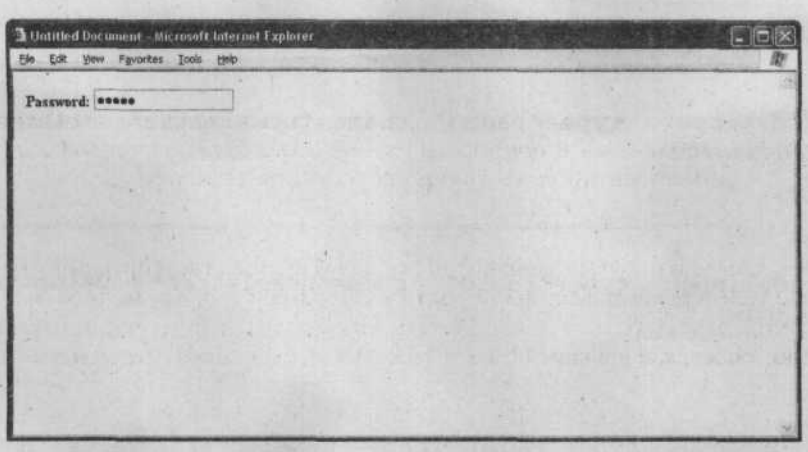


Рис. 68.8. Используйте поля паролей, когда не хотите отображать на экране в явном виде вводимый текст

Совет

Как и в случае текстовых полей и полей файлов, для поля пароля задается его физическая ширина с помощью атрибута `size`*, а также с помощью атрибута* `maxlength` *указывается максимальное число знаков, которое может вводиться посетителем.*

Работа с переключателями

Используйте переключатели для реализации небольшого числа взаимоисключающих вариантов или вариантов, из которых посетитель может сделать один и только один выбор, как на рисунке 68.9.

Совет

Переключатель невозможно сбросить так же, как флажок. Поэтому всегда задавайте в списке переключателей предварительно выбранный вариант. В этом случае посетитель ***должен*** *будет что-нибудь выбрать – если не вариант, используемый по умолчанию, тогда какой-нибудь другой.*

Листинг 68.9. Исходный код страницы, вид которой в браузере показан на рисунке 68.9

```
<form>
  <table>
    <tr>
      <td>Select the newsletter you would like to receive:</td>
```

```
    </tr>
    <tr>
      <td><input type="radio" name="newsletter" value="news"
checked>News</td>
    </tr>
    <tr>
      <td><input type="radio" name="newsletter" value="sports"
>Sports</td>
    </tr>
    <tr>
      <td><input type="radio" name="newsletter" value="business"
>Business</td>
    </tr>
    <tr>
      <td><input type="radio" name="newsletter" value="entertain-
ment">Entertainment</td>
    </tr>
    <tr>
      <td><input type="radio" name="newsletter" value="humor"
>Humor</td>
    </tr>
  </table>
</form>
```

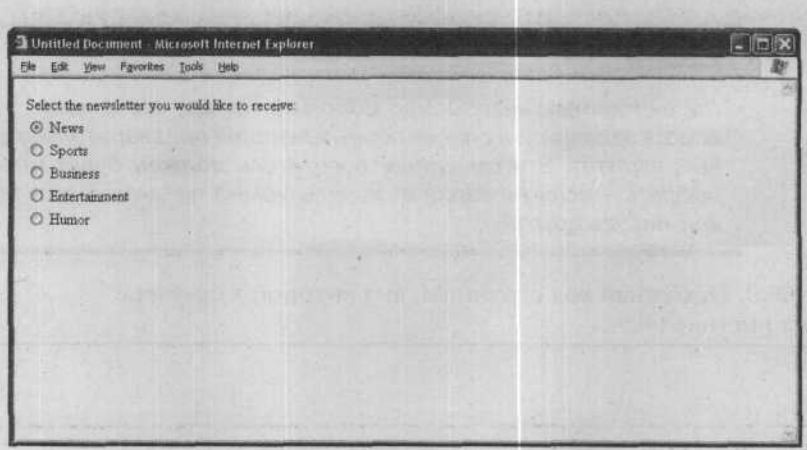


Рис. 68.9. Используйте переключатели, когда требуется реализовать небольшое число взаимоисключающих вариантов выбора

Обратите внимание на атрибут `checked` в теге `input` для переключателя «**News**» (Новости). С помощью этого атрибута осуществляется предварительный выбор указанного переключателя при загрузке формы.

Обычно названия различных элементов управления на форме являются уникальными. Не так обстоит дело с группами переключателей – очень важно, чтобы для всех переключателей в одной и той же группе использовалось одно название. В противном случае браузеру не будет известно, какие переключатели принадлежат той или иной группе, и вы не получите взаимоисключающих вариантов выбора.

Атрибут `value` определяет данные, передаваемые для выбранного переключателя. Таким образом, если посетитель выбирает получение бюллетеня «**Humor**» (Юмор), передаваемые формой данные выглядят подобно следующей строке:

```
newsletter=humor
```

Другие, не выбранные переключатели, не передаются браузером.

Работа с кнопками сброса

При щелчке на кнопке сброса (рис. 68.10) браузером возвращаются для всех полей в форме их значения по умолчанию. Если значение по умолчанию не предусмотрено для поля, оно очищается браузером.

Совет

Можно добавлять любое требуемое число кнопок сброса, но в действительности для каждой формы необходима только одна кнопка сброса. Обычно кнопку сброса лучше всего размещать вверху формы, справа от кнопки ***Submit*** *(Передать).*

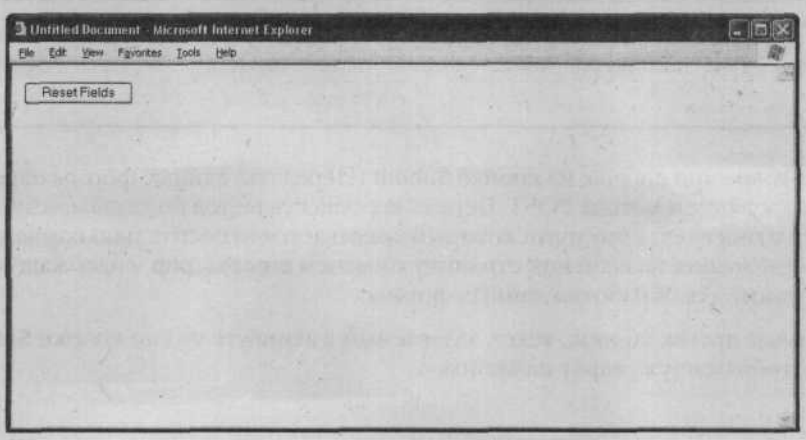


Рис. 68.10. Добавление кнопки сброса позволяет посетителю возвращать форму в исходное состояние

Листинг 68.10. Исходный код страницы, вид которой в браузере показан на рисунке 68.10

```
<form>
  <input type="reset" name="fieldsReset" value="Reset Fields">
</form>
```

Атрибутом `value` кнопки сброса определяется текст, который появляется на кнопке.

Работа с кнопками Submit (Передать)

При щелчке на кнопке **Submit** (Передать) данные формы посылаются на обработку (Рис. 68.11). Место назначения, куда отправляются данные формы, отображается в атрибуте `action` тега `<form>`, а способ отправки данных формы определяется атрибутом `method` тега `<form>`.

Часто задаваемые вопросы

Какие методы применяются для передачи формы?

Существует два метода: GET и POST. С помощью метода GET данные формы передаются в открытом текстовом формате, присоединенными в виде длинной строки пар атрибут/значение к концу URL-адреса действия формы. При использовании метода POST данные формы пересылаются в отдельном сообщении почти так же, как защищенная электронная почта между браузером и Web-сервером.

Листинг 68.11. Исходный код страницы, вид которой в браузере показан на рисунке 68.11

```
<form action="/forms/process.php" method="POST">
  <input type="submit" name="formSubmit" value="Submit Form">
</form>
```

В этом примере при щелчке на кнопке **Submit** (Передать) данные формы пересылаются с использованием метода POST. Передача осуществляется по задаваемому в атрибуте `action` относительному пути, который указывается относительно корня и включает в качестве пункта назначения страницу с именем **process.php**, содержащую серверную программу для обработки данных формы.

Как и в случае других кнопок, текст, задаваемый в атрибуте `value` кнопки **Submit** (Передать), отображается сверху на кнопке.

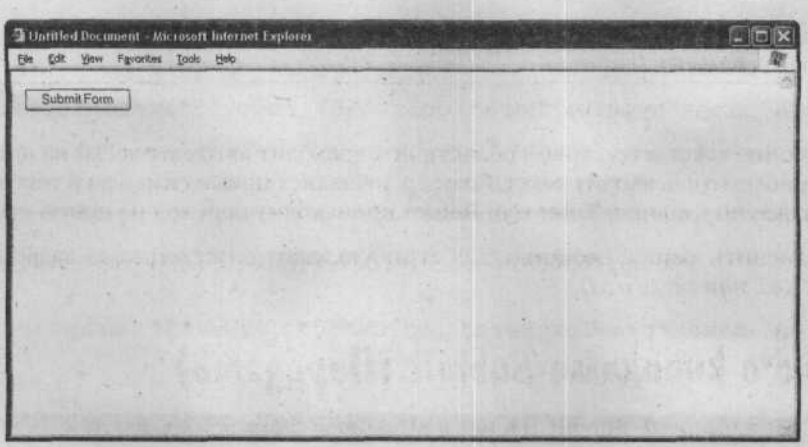


Рис. 68.11. С помощью кнопки **Submit** *(Передать) данные формы пересылаются на Web-сервер для обработки*

Часто задаваемые вопросы

Как обработать данные формы?

К сожалению, для обработки и использования данных формы необходимы серверные сценарии, которые выходят за рамки этой скромной книги.

Не отправляйте данные формы на клиентский документ, такой, как HTML-страница или файл JavaScript, поскольку в этом случае браузеру не известно, что надо делать.

Работа с текстовыми областями

Текстовые области – это многострочные поля для ввода текста, пример которых показан на рисунке 68.12.

Листинг 68.12. Исходный код страницы, вид которой в браузере показан на рисунке 68.12

```
<form>
    <textarea name="typeTextHere" cols="50" rows="5">Type text
here.</textarea>
</form>
```

Атрибут `cols` тега `<textarea>` задает размер текстовой области по горизонтали в знаках, а не пикселах, тогда как атрибут `rows` определяет размер текстовой области по вертикали в виде числа строк. Таким образом, в примере на рисунке 68.12 текстовая область по ширине равна 50 знакам и по высоте занимает 5 строк.

Предварительно задаваемый текст для текстовой области указывается между тегами `<textarea>`, а не в атрибуте `value`, используемом в других элементах управления.

Если вы не хотите, чтобы на экран выводился предварительно заданный текст, вообще не указывайте его:

```
<textarea name="typeTextHere" cols="50" rows="5"></textarea>
```

По умолчанию текст в текстовой области не переходит автоматически на новые строки. Когда посетитель вводит текст, браузер добавляет новые символы в текущую строку. При нажатии клавиши **Enter** или **Return** происходит переход на новую строку.

Чтобы изменить данное свойство, для атрибута `wrap` тега `textarea` задается значение `virtual` или `physical`:

```
<textarea name="typeTextHere" cols="50" rows="5" wrap="virtual">
</textarea>
```

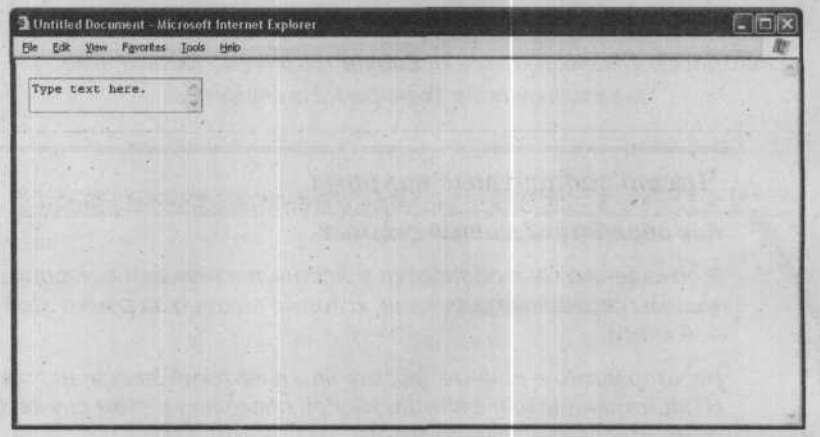


Рис. 68.12. Используйте текстовые области для ввода нескольких строк текста

Определение

*В случае **виртуального переноса строк** (virtual wrapping) текст в текстовой области передается формой без дополнительных символов возврата каретки. При **физическом переносе строк** (physical wrapping) текст передается формой с символами возврата каретки в конце каждой строки.*

Тип переноса строк влияет на способ передачи браузером данных формы. В случае виртуального переноса строк текст отображается в текстовой области разбитым на строки, и форма передает текст в виде непрерывной строки знаков без символов возврата каретки (если только посетитель не создает их в явном виде посредством нажатия клавиши **Enter** или **Return**). В случае физического переноса строк текст передается формой точно в том виде, в котором он отображается в текстовой области, с символами возврата каретки в конце каждой отдельной строки.

Работа с текстовыми полями

Текстовые поля – это элементы управления, предназначенные для ввода одной строки текста. Пример текстового поля показан на рисунке 68.13. Текстовые поля являются, вероятно, самими распространенными элементами управления в Интернете.

Ширина текстового поля (в знаках, а не в пикселах) задается атрибутом `size`. Максимальное число знаков, которое посетитель может ввести в поле, определяется значением атрибута `maxlength`, а текст, отображаемый по умолчанию, указывается в атрибуте `value`.

Листинг 68.13. Исходный код страницы, вид которой в браузере показан на рисунке 68.13

```
<form>
   <input type="text" name="typeTextHere" value="Type text here."
size="50" maxlength="80">
</form>
```

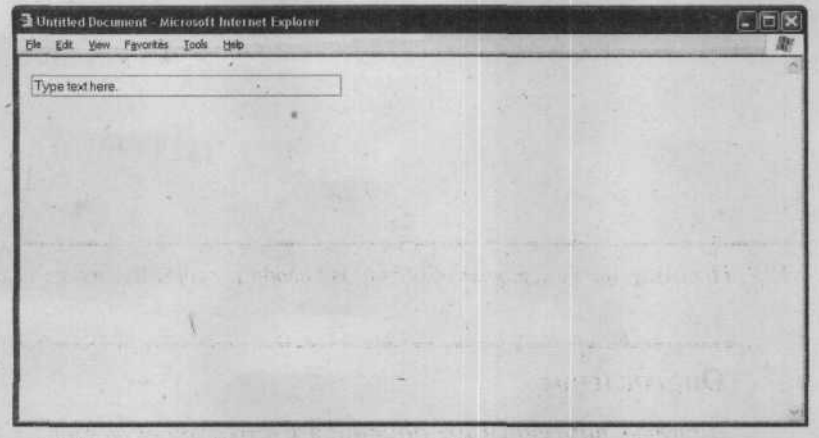


Рис. 68.13. Используйте текстовые поля для ввода одиночных строк текста

Когда посетитель щелкает внутри текстового поля, браузер не выбирает автоматически всю строку текста в поле. Это часто бывает неудобно, если текстовые поля содержат предварительно заданные значения, поскольку посетителю требуется очистить прежние сведения до того, как вводить новые данные. Простая строка кода на JavaScript исправляет эту проблему:

```
<input type="tex" name="typeTextHere" value="Type text here."
size="50" maxlength="80" onFocus="this.select();">
```

Событие `onFocus` происходит при выделении посетителем текстового поля посредством щелчка внутри поля или в результате перехода на это поле с помощью клавиши **Tab**. При установке фокуса на текстовое поле методом `select()` выделяется все в этом поле.

Часто задаваемые вопросы

Что произойдет, если опустить атрибут `size` текстового поля?

Если опустить атрибут `size`, браузером будет использоваться ширина текстовых полей по умолчанию, которая зависит от конкретного браузера. Пропуск атрибута `maxlength` означает отсутствие предела на число знаков, которые может вводить посетитель в это поле.

Совет

Код `onFocus="this.select();"` можно также добавить в поля паролей и текстовые области.

ГЛАВА 69.
Применение стилей к текстовым элементам управления

Элементы управления текстом воспринимают CSS-правила стилей также, как это делают и другие HTML-элементы. С помощью каскадных таблиц стилей (CSS) можно, например, изменить стандартный шрифт меню или цвет фона кнопки **Submit** (Передать). Однако, как и в случае других HTML-элементов, разные браузеры не всегда «дружат» с CSS-стилями, которые определяются дизайнером для своих элементов управления, поэтому обязательно просматривайте созданные Web-страницы в различных браузерах.

Часто задаваемые вопросы

Можно ли использовать CSS-стили с нетекстовыми элементами управления?

Да, но не стоит связывать с этим большие надежды. Нетекстовые элементы управления, например, флажки, обычно не выглядят так, как они должны были бы выглядеть при применении к ним CSS-стилей.

Изменение текстового стиля

Различные атрибуты текста, существующие в каскадных таблицах стилей, позволяют изменять внешний вид текста в любом элементе управления, использующем текст, включая текстовые поля, текстовые области, кнопки, меню и списки. Таблица 69.1, в которой обобщены стандартные CSS-атрибуты текста, может служить памяткой для дизайнера на случай, если потребуется вспомнить некоторые из этих атрибутов.

Используйте комбинации атрибутов из таблицы 69.1 для разработки сложных текстовых элементов управления, подобных тем, что показаны на рисунке 69.1. Эти примеры показаны вверху, и они наглядно демонстрируют, что можно сделать, если не предъявлять высоких требований к эстетичности.

Совет

Поскольку разные элементы управления часто используют один и тот же тег, то лучше создать специальные классы для каждого типа элементов управления, чем создавать одно общее правило стиля для HTML-тегов элементов управления.

Табл. 69.1. Стандартные CSS-атрибуты для текста в текстовых элементах управления

Атрибут	Управляемый элемент	Пример
font-family	Гарнитура шрифта текста в элементе управления	font-family: Arial, Helvetica, sans-serif;
font-style	Стиль текста в элементе управления	font-style: italic;
font-weight	Начертание шрифта текста в элементе управления	font-weight: bold;
font-size	Размер шрифта текста в элементе управления	font-size: 12px;
font-variant	Варьирование шрифта текста в элементе управления	font-variant: small-caps;
text-transform	Способ изменения браузером регистра букв текста внутри элемента управления	text-transform: lowercase;
text-decoration	Подчеркивание, линия сверху или перечеркивание текста внутри элемента управления	text-decoration: underline;

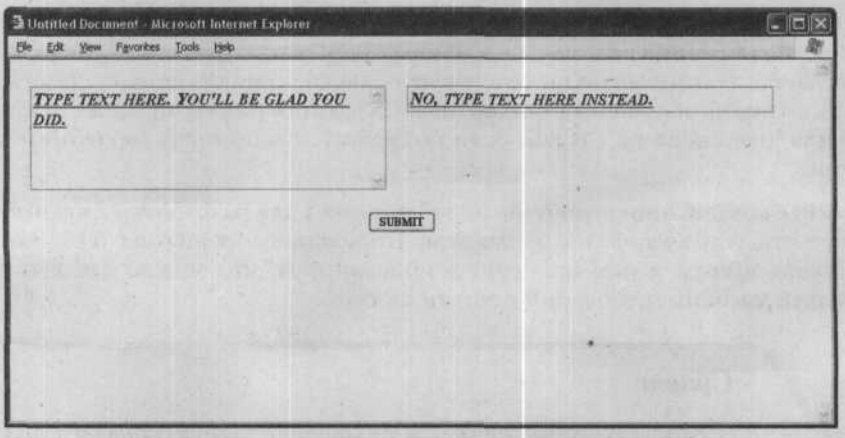


Рис. 69.1. Примените CSS-атрибуты текста, чтобы изменить внешний вид текста в текстовых элементах управления

Листинг 69.1. Исходный код страницы, вид которой в браузере показан на рисунке 69.1

```
<head>
  <style type="text/css">

    .textfields {
      font-family: "Times New Roman", Times, serif;
      font-style: italic;
      font-weight: bold;
      font-size: 18px;
      font-variant: small-caps;
      font-decoration: underline;
    }

    .buttons {
      font-family: "Times New Roman", Times, serif;
      font-size: 12px;
      font-weight: bold;
      font-transform: uppercase;
    }

  </style>
</head>

<body>

  <form>
    <table cellpadding="10">
      <tr valign="top">
        <td>
          <textarea cols="50" rows="5" class="textfields">Type text
here. You'll be glad you did.</textarea>
        </td>
        <td>
```

```
        <input type="text" class="textfields" size="50" value="No,
type text here instead.">
        </td>
      </tr>
      <tr>
        <td colspan="2" align="center">
         <input type="submit" value="Submit" class="buttons"></td>
      </tr>
    </table>
  </form>

</body>
```

Изменение цвета и границы

CSS-атрибуты `color` и `background-color` позволяют изменять основной цвет и цвет фона в текстовых элементах управления, примеры которых показаны на рисунке 69.2. Можно также изменять границы кнопок с помощью атрибутов `border-color`, `border-width` и `border-style`.

Листинг 69.2. Исходный код страницы, вид которой в браузере показан на рисунке 69.2

```
<head>
  <style type="text/css">

    .listsmenu {
      color: #FFFFFF;
      background-color: #000000;
    }

    .buttons {
      background-color: #000000;
      color: #FFFFFF;
      border-color: #999999;
      border-width: 6px;
      border-style: double;
      font-weight: bold;
```

```
    }

  </style>
</head>

<body>

  <form>
    <table cellpadding="10">
      <tr valign="top">
        <td><select name="select" size="5" class="listsmenus">
          <option value="1" selected>List Option 1</option>
          <option value="2">List Option 2</option>
          <option value="3">List Option 3</option>
          <option value="4">List Option 4</option>
          <option value="5">List Option 5</option>
        </select></td>
        <td><select name="select" class="listsmenus">
          <option value="1" selected>Menu Option 1</option>
          <option value="2">Menu Option 2</option>
          <option value="3">Menu Option 3</option>
          <option value="4">Menu Option 4</option>
          <option value="5">Menu Option 5</option>
        </select></td>
      </tr>
      <tr>
            <td colspan="2" align="center"><input type="submit"
class="buttons" value="Submit"></td>
      </tr>
    </table>
  </form>

</body>
```

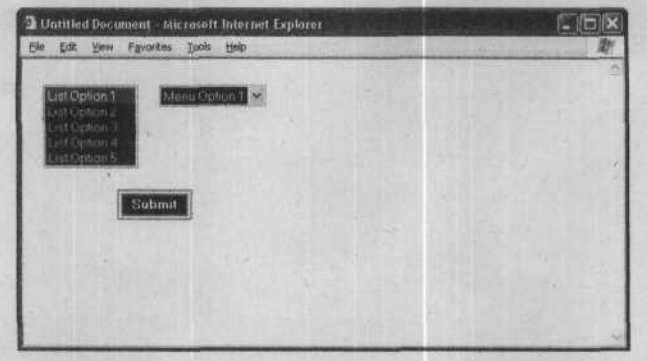


Рис. 69.2. Используйте атрибуты `color` *и* `background-color`*, чтобы изменить в текстовом элементе управления основной цвет и цвет фона*

ГЛАВА 70.
Проверка достоверности входных данных формы

Хотя для обработки данных формы необходимы серверные сценарии, чтобы просмотреть введенные посетителем данные до их отправки на Web-сервер, можно воспользоваться старым добрым клиентским кодом. Этот процесс называется **проверкой достоверности** (validation) и пригоден для форм большинства типов.

Прогон проверки на достоверность помогает предотвратить ошибки, как умышленные, так и случайные, которые могут вызвать сбои или отказы в работе серверных программ, получающих, в конечном счете, данные формы. Допустим, ваша форма содержит запрос посетителю на предоставление его адреса электронной почты с тем, чтобы серверный обработчик формы мог отправить электронное сообщение с подтверждением. Очевидно, необходимо, чтобы посетитель предоставил свой адрес электронной почты, иначе серверная программа не будет работать корректно. Если кто-нибудь из посетителей попытается оставить поле пустым или попробует ввести что-нибудь отличное от адреса электронной почты, сценарий проверки на достоверность зафиксирует это и выведет запрос на исправление. Конечно, посетитель может повторно ввести недостоверный адрес электронной почты, тем не менее, сценарий проверки на достоверность, по крайней мере, отсечет немалую часть потенциально неправильных данных.

Определение

Проверка достоверности *(validating) входных данных формы означает их просмотр на правильность до отправки на Web-сервер.*

Любой сценарий проверки на достоверность должен тесно координировать со структурой формы. И поскольку никакие две формы не похожи друг на друга, не существует такого понятия, как универсальный, пригодный для всех случаев сценарий проверки на достоверность. По этой причине разделы инструментария данной главы не похожи на другие подобные разделы этой книги, потому что их недостаточно просто скопировать на свой Web-сайт. Вместо этого следует начать со «Структуры сценария проверки достоверности», а затем выбрать код из подходящего раздела инструментария, добавив требуемые функции проверки на достоверность для своей конкретной формы.

Инструментарий

Структура сценария проверки достоверности

Данный инструментарий содержит структуру сценария проверки достоверности формы. Сам по себе он ничего не проверяет – для этого необходимо добавить другие блоки кода. Другие инструментарии в этой главе предоставляют действующие процедуры проверки на достоверность. Выберите необходимые процедуры и добавьте их в эту структуру.

```
<script language="JavaScript">

/* Если этот сценарий добавляется во внешний файл
JavaScript, теги <script>, вставленные в начале и
в конце данного списка, не нужны. Если сценарий
вставляется в код Web-страницы, убедитесь, что это
та самая страница, которая содержит форму */

  function validateForm() {

/* Здесь добавьте компоненты из других инструмен-
тариев */

    return true;

/* Предполагая, что данные формы прошли все провер-
ки на достоверность, приведенная выше команда пред-
писывает браузеру перейти к передаче формы на Web-
сервер. */

  }

</script>
```

Помимо данной структуры, для функции проверки на достоверность требуется еще один компонент: форма, которая при ее отправке запускает сценарий. Добавьте событие `onSubmit` в тег `<form>` в соответствии со следующим кодом:

```
<form name="formname" action="formaction" method=
"formmethod"> onSubmit="return validateForm();">
```

Конечно, следует вставить соответствующие значения в атрибуты `name`, `action` и `method` для конкретной формы.

Инструментарий

Проверка адреса электронной почты

Добавьте данный блок кода в сценарий проверки на достоверность, чтобы контролировать ввод посетителем адреса электронной почты в текстовое поле:

```
var email = new String(document.formname.
textfieldname.value);

/* В приведенную выше строку кода вставьте имя фор-
мы и имя текстового поля, которое содержит адрес
электронной почты. Этот оператор создает перемен-
ную с именем "email" для хранения данных, вводимых
посетителем в текстовое поле. */
```

```
if (email.indexOf("@") == -1) {

/* Приведенный выше оператор проверяет наличие в
поле знака "@". Если этот знак отсутствует, сцена-
рием проверки на достоверность предполагается, что
введенная посетителем строка не является адресом
электронной почты. */

  alert("Необходимо ввести правильный адрес элект-
ронной почты.");

/* Приведенный выше оператор отображает сообщение
об ошибке в открывающемся окне. */

  document.formname.textfieldname.select();

/* Приведенный выше оператор выделяет на форме по-
ле электронной почты. */

  return false;

/* Приведенный выше оператор отменяет передачу фор-
мы. */

}
```

Инструментарий

Проверка требуемых текстовых полей

Добавьте данный блок кода в сценарий проверки на достоверность, чтобы проверить заполнение посетителем всех требуемых текстовых полей. Данный «Инструментарий» работает также с полями паролей и текстовыми областями:

```
var field01 = new String(document.formname.
requiredfield01.value);

var field02 = new String(document.formname.
requiredfield02.value);

var field03 = new String(document.formname.
requiredfield03.value);

var field04 = new String(document.formname.
requiredfield04.value);

var field05 = new String(document.formname.
requiredfield05.value);

/* Приведенные выше операторы создают переменные
для каждого требуемого текстового поля на форме.
Если число текстовых полей меньше пяти, опустите
ненужные строки кода. Если количество текстовых по-
лей равно или больше шести, просто добавьте в сце-
нарий строки кода, используя приведенный код в ка-
честве шаблона. */

var pass = true;
```

```
/* Приведенная выше строка кода создает логическую
(true/false) переменную с именем "pass" и присваи-
вает ей значение "true". */

/* Приведенные ниже блоки кода if/then проверяют,
заполнены ли посетителем все требуемые текстовые
поля. Если сценарий определяет пропущенное поле,
переменная pass принимает значение false. Для каж-
дого текстового поля необходим один блок кода
if/then, поэтому настройте их в соответствии с
конкретной формой, удаляя ненужные блоки или добав-
ляя дополнительные блоки кода. */

if (field01.length == 0) {
  pass = false;
}
if (field02.length == 0) {
  pass = false;
}
if (field03.length == 0) {
  pass = false;
}
if (field04.length == 0) {
  pass = false;
}
if (field05.length == 0) {
  pass = false;
}

/* Следующий блок кода проверяет значение перемен-
ной pass. Если значение переменной pass равно
false, это свидетельствует о том, что, по крайней
мере, одно требуемое текстовое поле является пус-
тым. В этом случае браузер выводит сообщение об
ошибке и останавливает передачу формы. */

if (pass == false) {
  alert("Следует заполнить все требуемые поля.");
  return false;
}
```

Инструментарий

Проверка требуемых списков и меню

Добавьте данный блок кода в сценарий проверки на достоверность, чтобы проверить заполнение посетителем всех требуемых списков и меню.

Предполагается, что список или меню начинается с предварительно выбранного стандартного пункта, задаваемого аналогично следующему примеру:

```
<option selected>Выберите пункт...</option>
```

Текст «Выберите пункт...», заключенный между тегами <option>, можно изменять в соответствии с требованиями конкретной формы. Просто убедитесь, что вы изменили также текст в следующем коде:

```
var list01 = document.formname.requiredlist01;
var list01Text = list01.options[list01.
selectedIndex].text;

var list02 = document.formname.requiredlist02;
var list02Text = list02.options[list02.
selectedIndex].text;

var list03 = document.formname.requiredlist03;
var list03Text = list03.options[list03.
selectedIndex].text;

var list04 = document.formname.requiredlist04;
var list04Text = list04.options[list04.
selectedIndex].text;

var list05 = document.formname.requiredlist05;
var list05Text = list05.options[list05.
selectedIndex].text;

/* Приведенный выше код создает в форме новые переменные для требуемых списков и меню и принимает текст выбранных вариантов по каждой переменной. Если число списков или меню меньше пяти, удалите ненужные строки кода. Если число списков или меню равно или больше шести, добавьте новые строки кода.

Следующая строка кода создает новую переменную с именем "pass" и устанавливает для нее значение равное "true". */

var pass = true;

/* Приведенные ниже блоки кода if/then проверяют, отображаются ли по-прежнему в требуемых списках или меню пункты, установленные по умолчанию. Если хотя бы в одном списке или меню выделенным является пункт, установленный по умолчанию, переменная pass принимает значение false. Для каждого списка или меню необходим один блок кода if/then, поэтому
```

```
удалите ненужные блоки или добавьте дополнительные
блоки кода, если необходимо больше пяти блоков ко-
да if/then. */
if (list01Text == "Выберите пункт…") {
  pass = false;
}
if (list02Text == "Выберите пункт…") {
  pass = false;
}
if (list03Text == "Выберите пункт…") {
  pass = false;
}
if (list04Text == "Выберите пункт…") {
  pass = false;
}
if (list05Text == "Выберите пункт…") {
  pass = false;
}
/* Следующий блок кода if/then проверяет значение
переменной pass. Если значение переменной pass рав-
но false, форма не передается. */
if (pass == false) {
  alert("Следует сделать выбор во всех списках и
меню.");
   return false;
}
```

Инструментарий

Проверка ввода только буквенно-цифровых символов

Добавьте данный блок кода в сценарий проверки на достоверность, чтобы проконтролировать, что посетитель вводит только буквенно-цифровые знаки в текстовое поле, текстовую область или поле пароля:

```
var field = new String(document.formname.field-
name.value);
/* Приведенный выше код создает новую переменную
для значения требуемого поля. Замените formname
именем формы, fieldname именем поля, которое тре-
буется проконтролировать.
Следующая строка кода создает переменную с именем
"pass" и присваивает этой переменной значение
"true". */
```

```
var pass = true;

/* Следующий блок текста описывает цикл for/next.
Данный сценарий просматривает каждый знак в поле и
определяет, является ли знак буквой или цифрой. Ес-
ли знак не является ни буквой и ни цифрой, пере-
менная pass принимает значение false. */

for (var x = 0; x < field.length; x++) {

  if (field.charCodeAt(x) < 48 ||

   (field.charCodeAt(x) > 57 && field.charCodeAt(x)
< 65) ||

   (field.charCodeAt(x) > 90 && field.charCodeAt(x)
< 97) ||

   field.charCodeAt(x) > 122) {

            pass = false;

   }

  }

/* Теперь, если переменная pass принимает значение
false, браузер выводит сообщение об ошибке, удаля-
ет значение некорректно заполненного поля и авто-
матически выделяет это поле, чтобы посетитель мог
повторить попытку ввода. */

if (pass == false) {

  alert ("Вводимая запись содержит символы отлич-
ные от букв и цифр.")

  document.formname.fieldname.value ="";

  document.formname.fieldname.select();

    return false;

}
```

Инструментарий

Проверка соответствия полей паролей

Добавьте данный блок кода в сценарий проверки на достоверность, чтобы проконтролировать соответствие полей паролей, заполняемых посетителем. Общеизвестна процедура: в одном поле вводится пароль, а в следующем поле посетитель должен ввести пароль повторно. Этот сценарий работает также с текстовыми полями и текстовыми областями.

```
var password = new String(document.formname.pass-
wordname.value);
var retype = new String(document.formname.retype-
name.value);

/* Приведенный выше код создает новые переменные
для значений поля пароля и поля повторного ввода
```

```
пароля. Замените переменную formname именем формы,
переменную passwordname замените именем поля паро-
ля и переменную retypename - именем поля, в кото-
ром посетитель повторно вводит пароль. */
if (password.valueOf() != retype.valueOf()) {
  alert("Поля паролей не совпадают.");
    document.formname.retypename.value ="";
  document.formname.passwordname.select();
  return false;
}
/* Приведенный выше блок кода if/then реализует
сравнение значения пароля со значением, набранным
повторно. Если эти значения не совпадают, браузе-
ром выводится сообщение об ошибке, очищается поле
повторного ввода пароля, выделяется поле пароля и
останавливается передача формы. */
```

Инструментарий

Проверка установки флажка «I Agree» (Я согласен)

Добавьте данный блок кода в сценарий проверки на достоверность, чтобы проконтролировать состояние флажка «I Agree» (Я согласен), который посетитель устанавливает, когда соглашается с условиями предоставления услуг, политикой конфиденциальности или с любыми другими предварительными требованиями.

```
if   (document.formname.checkboxname.checked   ==
false) {
  alert("Необходимо подтвердить принятие условий,
прежде чем продолжить.")
  return false;
}
/* Приведенный выше блок кода if/then проверяет ус-
тановку флажка. Если флажок не установлен, браузе-
ром выводится сообщение об ошибке и передача формы
прекращается. Замените переменную formname именем
формы, а переменную checkboxname замените именем
флажка "I agree" (Я согласен). */
```

ГЛАВА 71.
Навигация с помощью клавиши Tab

Не все путешествуют по Интернету с помощью мыши. Некоторые люди не могут это делать по причине инвалидности. Другие просто предпочитают пользоваться клавиатурой. Как вы, возможно, знаете, перемещаться по интерактивным элементам на странице, таким как ссылки и элементы управления формой, можно, нажимая клавишу **Tab**.

Совет

Атрибут `tabindex` *можно также использовать в тегах-анкерах ссылок.*

По умолчанию браузер осуществляет переходы, выполняемые с помощью клавиши Tab, в порядке расположения соответствующих тегов в HTML-коде. Однако когда форма скомпонована в виде таблицы, последовательность переходов браузера по элементам управления может нарушаться, поскольку теги элементов управления не всегда располагаются в коде в том логическом порядке, в котором выполняются переходы на экране.

Чтобы переходы по элементам управления формы, осуществляемые посетителем посредством нажатия клавиши **Tab**, выполнялись в естественном и логичном порядке, можно задать точную очередность элементов управления с помощью атрибутов `tabindex`.

Возьмем форму на рисунке 71.1. В коде ее элементов управления не используются атрибуты `tabindex`, и порядок перемещения по элементам управления определяется самим браузером.

Совет

Если атрибут `tabindex` *установлен для одного тега, установите данный атрибут для всех элементов управления и ссылок на Web-странице. В противном случае при нажатии клавиши* **Tab** *браузером могут выполняться переходы в ненадлежащем порядке.*

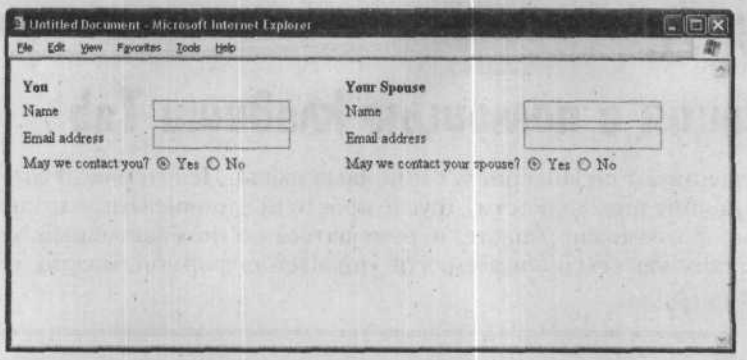


Рис. 71.1. В коде элементов управления этой формы не используется атрибут `tabindex`. *Порядок переходов при нажатии клавиши* **Tab** *определяется самим браузером*

Листинг 71.1. Исходный код страницы, вид которой в браузере показан на рисунке 71.1

```
<form>
  <table>
    <tr>
      <td colspan="2"><strong>You</strong></td>
      <td width="50" rowspan="4"> </td>
      <td colspan="2"><strong>Your Spouse</strong></td>
    </tr>
    <tr>
      <td>Name</td>
      <td><input type="text" name="yourName"</td>
      <td>Name</td>
      <td><input type="text" name="spouseName"</td>
    </tr>
    <tr>
      <td>Email address</td>
      <td><input type="text" name="yourEmail"</td>
      <td>Email address</td>
      <td><input type="text" name="spouseEmail"</td>
    </tr>
    <tr>
      <td>May we contact you?</td>
     <td><input type="radio" name="contactYou" value="yes" checked>
        Yes
```

```
        <input type="radio" name="contactYou" value="no">
        No</td>
      <td>May we contact your spouse?</td>
        <td><input type="radio" name="contactSpouse" value="yes"
checked>
        Yes
        <input type="radio" name="contactSpouse" value="no">
        No</td>
    </tr>
  </table>
</form>
```

Если порядок Tab-переходов по форме определяется браузером, он проходит по HTML-коду сверху вниз. Таким образом, браузером сначала выбирается поле **yourName**, затем – поле **spouseName**, далее – поле **yourEmail**, после этого – поле **spouseEmail** и так далее. Но данный порядок не является оптимальным для формы. Было бы лучше, если бы посетитель мог заполнить сначала свои личные сведения, прежде чем перейти на поля, соответствующие сведениям о супруге (Spouse).

Введите в код атрибут `tabindex`:

```
<form>
  <table>
    <tr>
      <td colspan="2"><strong>You</strong></td>
      <td width="50" rowspan="4"> </td>
      <td colspan="2"><strong>Your Spouse</strong></td>
    </tr>
    <tr>
      <td>Name</td>
      <td><input type="text" name="yourName" tabindex="1"></td>
      <td>Name</td>
      <td><input type="text" name="spouseName" tabindex="5"></td>
    </tr>
    <tr>
      <td>Email address</td>
      <td><input type="text" name="yourEmail" tabindex="2"></td>
      <td>Email address</td>
      <td><input type="text" name="spouseEmail" tabindex="6"></td>
    </tr>
```

```
    <tr>
      <td>May we contact you?</td>
     <td><input type="radio" name="contactYou" value="yes" checked
tabindex="3">
        Yes
              <input type="radio" name="contactYou" value="no"
tabindex="4">
        No</td>
      <td>May we contact your spouse?</td>
        <td><input type="radio" name="contactSpouse" value="yes"
checked tabindex="7">
        Yes
            <input type="radio" name="contactSpouse" value="no"
tabindex="8">
        No</td>
    </tr>
  </table>
</form>
```

Совет

Если на одной и той же странице расположено несколько форм, не возвращайтесь обратно к `tabindex="1"` *для первого элемента на следующей форме. Продолжайте нумерацию с того места, на котором она закончилась для предыдущей формы.*

Обратите внимание, что каждый элемент управления получает атрибут `tabindex`. Значение этого атрибута определяет место в последовательности Tab-переходов, от наименьшего к наибольшему значению. Таким образом, последовательно принимая значения `tabindex`, в браузере выполняется логичный переход от поля **yourName** к полю **yourEmail**, далее к полю **contactYou yes**, затем к полю **contactYou no** и только после этого совершается переход на столбец сведений о супруге (spouse) и продолжается перемещение по соответствующим полям в том же порядке.

Часто задаваемые вопросы

Что происходит, если в двух или более тегах значения атрибутов tabindex одинаковы?

Браузер упорядочивает их в соответствии с положением тегов в HTML-коде: те, которые ближе к верху кода, встречаются раньше в последовательности ***Tab****-переходов.*

ГЛАВА 72.
Работа с наборами полей

Набор полей – это группа логически связанных элементов управления. Компоненты набора полей не обязательно должны быть элементами управления одного типа. В действительности каждый элемент управления может отличаться от других. До тех пор, пока элементы управления отображаются рядом друг с другом в HTML-списке, их можно группировать вместе в виде набора полей, как в примере на рисунке 72.1.

Определение

Набор полей *(fieldset) – это группа логически связанных элементов управления.*

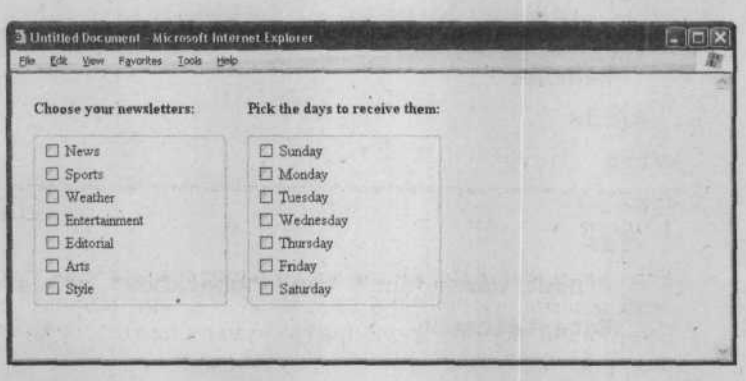


Рис. 72.1. На данной форме представлено два логических раздела или набора полей

Листинг 72.1. Исходный код страницы, вид которой в браузере показан на рисунке 72.1

```
<form>
  <table cellpadding="10">
    <tr valign="top">
      <td width="50%">
        <p><strong>Choose you newsletters:</strong></p>

<!-- Начало первого набора полей -->

        <fieldset>
          <table>
```

```
<tr>
  <td>
    <input name="news" type="checkbox" value="yes">
    News
  </td>
</tr>
<tr>
  <td>
    <input name="sports" type="checkbox" value="yes">
    Sports
  </td>
</tr>
<tr>
  <td>
   <input name="weather" type="checkbox" value="yes">
    Weather
  </td>
</tr>
<tr>
  <td>
    <input name="ent" type="checkbox" value="yes">
    Entertainment
  </td>
</tr>
<tr>
  <td>
    <input name="ed" type="checkbox" value="yes">
    Editorial
  </td>
</tr>
<tr>
  <td>
    <input name="arts" type="checkbox" value="yes">
    Arts
  </td>
</tr>
```

```
            <tr>
              <td>
                <input name="style" type="checkbox" value="yes">
                Style
              </td>
            </tr>
          </table>
        </fieldset>

<!-- Конец первого набора полей -->

      </td>
      <td width="50%">
        <p><strong>Pick the days to receive them:</strong></p>

<!-- Начало второго набора полей -->

        <fieldset>
          <table>
            <tr>
              <td>
                <input name="sun" type="checkbox" value="yes">
                Sunday
              </td>
            </tr>
            <tr>
              <td>
                <input name="mon" type="checkbox" value="yes">
                Monday
              </td>
            </tr>
            <tr>
              <td>
                <input name="tues" type="checkbox" value="yes">
                Tuesday
              </td>
```

```
            </tr>
            <tr>
              <td>
                <input name="wed" type="checkbox" value="yes">
                Wednesday
              </td>
            </tr>
            <tr>
              <td>
                <input name="thurs" type="checkbox" value="yes">
                Thursday
              </td>
            </tr>
            <tr>
              <td>
                <input name="fri" type="checkbox" value="yes">
                Friday
              </td>
            </tr>
            <tr>
              <td>
                <input name="sat" type="checkbox" value="yes">
                Saturday
              </td>
            </tr>
          </table>
        </fieldset>

<!-- Конец второго набора полей -->

      </td>
    <tr>
  </table>
</form>
```

Как можно видеть, браузер рисует тонкую, прямоугольную рамку вокруг элементов управления в наборе полей. С помощью тега <legend> можно даже вставить короткую строку текста в рамку, как на рисунке 72.2.

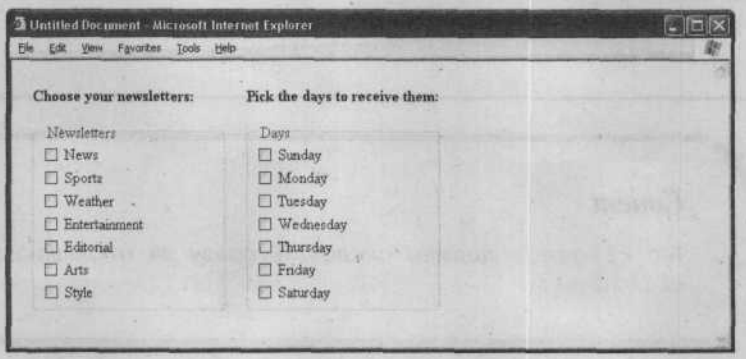


Рис. 72.2. Используйте тег <legend>, чтобы вставить короткую строку текста в прямоугольную рамку

Листинг 72.2. Исходный код страницы, вид которой в браузере показан на рисунке 72.2

```
<form>

<!-- Начало макета таблицы -->

  <p><strong>Choose you newsletters:</strong></p>
  <fieldset>
    <legend>Newsletters</legend>

<!-- Здесь располагается код элементов управления -->

  </fieldset>

<!-- Дополнительный макет таблицы -->

  <p><strong>Pick the days to receive them:</strong></p>
  <fieldset>
    <legend>Days</legend>

<!-- Код дополнительных элементов управления -->
```

```
  </fieldset>

<!-- Конец макета таблицы -->

</form>
```

Совет

Тег <legend> должен следовать сразу за открывающим тегом <fieldset>.

По умолчанию цвет текста легенды – синий в Internet Explorer и черный в Netscape. В этом случае синий цвет представляется неудачным выбором, поскольку он также является стандартным цветом непосещенных гиперссылок. Синяя легенда может вызвать у посетителей желание щелкнуть на ней, даже в отсутствие красноречивого подчеркивания.

К счастью каскадные таблицы стилей позволяют переопределить тег <legend> для отображения текста легенды с любым требуемым цветом шрифта:

```
<style type="text/css">

  legend {
    color: #000000;
  }

</style>
```

Почему бы заодно не сделать легенду с полужирным начертанием шрифта:

```
<style type="text/css">

  legend {
    color: #000000;
    font-weight: bold;
  }

</style>
```

Результаты применения этого стиля показаны на рисунке 72.3.

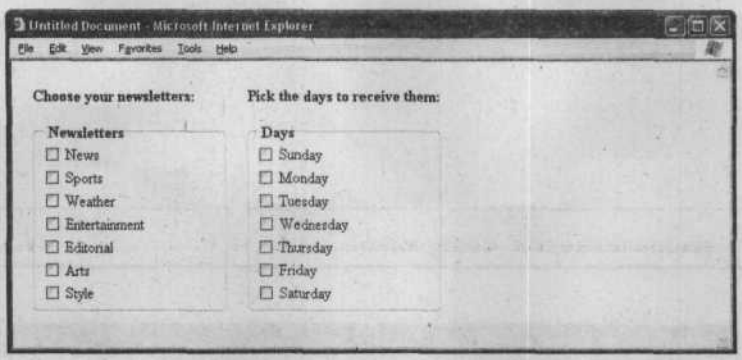


Рис. 72.3. Используйте каскадные таблицы стилей для изменения внешнего вида легенд наборов полей

Часто задаваемые вопросы

Можно ли создать CSS-стили для наборов полей?

Конечно, но будьте внимательны. Помните, что в каскадных таблицах стилей дочерний тег практически всегда наследует стили своего родителя. Наборы полей как раз относятся к этим случаям. Все дочерние теги набора полей, включая легенду, элементы управления и текст, наследуют стили набора полей.

Каскадные таблицы стилей можно также использовать для изменения внешнего вида рамки, обрамляющей набор полей. Для решения этой задачи прекрасно подойдут атрибуты `border-color`, `border-style` и `border-width`. Для изменения цвета текста внутри набора полей используйте в правиле стиля атрибут `color`, а для создания цветной области на заднем фоне набора полей воспользуйтесь атрибутом `background-color`. Результат применения этих атрибутов показан на рисунке 72.4.

Листинг 72.3. Исходный код страницы, вид которой в браузере показан на рисунке 72.3

```
<head>

  <style type="text/css">

    <fieldset {
      border-style: dashed;
      border-width: 4px;
      border-color: #FF0000;
      color: #FF0000;
      background-color: #FFCCCC;
```

```
    }

  </style>
</head>

<!-- Здесь располагается содержимое -->
```

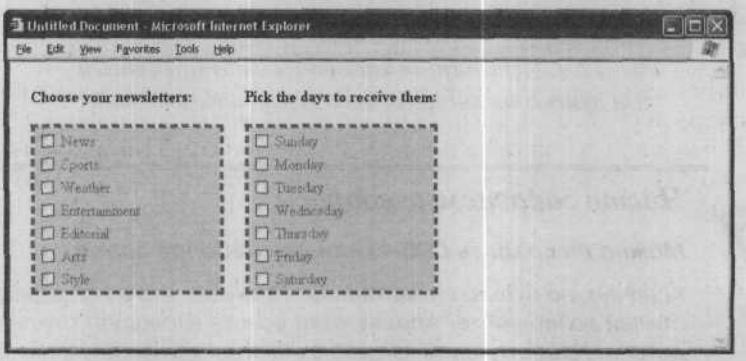


Рис. 72.4. Для тега `<fieldset>` также можно создать правило стиля

ГЛАВА 73.
Работа с метками

Возможно, вы задаетесь вопросом, как правильно разметить метки на форме в соответствии с требованиями консорциума W3C (World Wide Web Consortium) и с соблюдением существующих стандартов. Для элементов управления предоставляются собственные теги, поэтому здесь нет повода для беспокойства. К тому же, вероятно, вы разрабатываете макет формы в виде таблицы, которая размечается собственным набором тегов. Однако, как насчет текста внутри формы, описывающего то, что посетители должны вводить в поля? Другими словами, существуют ли теги для **меток**?

Может возникнуть искушение использовать для разметки меток знакомые текстовые теги, например, теги абзацев или некоторые виды заголовков. Проблема состоит в том, что теги должны точно идентифицировать тип элементов, который они размечают. На самом деле метка в форме не является абзацем. Это и не заголовок в том смысле, что HTML определяет заголовок как шапку или маркер раздела страницы.

Данная головоломка не так уж трудна, как может показаться из приведенного выше вступления. Язык HTML предоставляет мало известный текстовый тег, особенно в применении к меткам формы: достаточно удобный тег `<label>`. Результат его применения показан на рисунке 73.1.

Определение

***Метка** (label) в форме – это фрагмент текста, который описывает то, что посетитель должен вводить в соответствующем поле.*

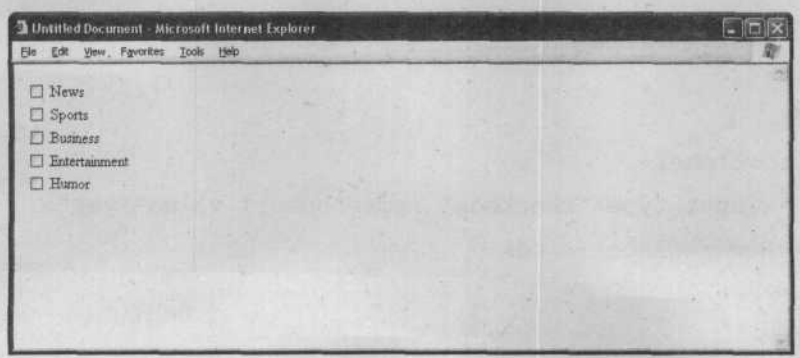


Рис. 73.1. В коде этой формы для разметки текстовых меток флажков используются теги `<label>`

Листинг 73.1. Исходный код страницы, вид которой в браузере показан на рисунке 73.1

```
<form>
  <table>
    <tr>
      <td><label>
        <input type="checkbox" name="news" value="yes">
        News</label></td>
    </tr>
    <tr>
      <td><label>
        <input type="checkbox" name="sports" value="yes">
        Sports</label></td>
    </tr>
    <tr>
      <td><label>
        <input type="checkbox" name="business" value="yes">
        Business</label></td>
    </tr>
    <tr>
      <td><label>
        <input type="checkbox" name="entertainment" value="yes">
        Entertainment</label></td>
    </tr>
    <tr>
      <td><label>
        <input type="checkbox" name="humor" value="yes">
        Humor</label></td>
    </tr>
  </table>
</form>
```

Как можно заметить, браузер не воспроизводит метки с каким-либо специальным стилем. Метки выглядят как обычный текст. Зачем беспокоиться о дополнительной разметке. На это существуют, по крайней мере, три убедительные причины.

Во-первых, чем точнее разметка, тем лучше становится доступность Web-страницы для разных категорий посетителей. Если какой-либо элемент на странице можно идентифицировать как метку, его так и следует размечать. В этом случае, когда кто-нибудь посетит ваш сайт с нестандартным устройством просмотра, например, с устройством чтения с экрана или преобразователем текста в речь, содержимое вашего сайта будет воспроизведено более точно.

Во-вторых, при использовании тегов `<label>` разметка показывает точные взаимосвязи элементов на форме. Вы создаете связь между текстом и элементом управления, которая в противном случае отсутствует. Конечно, посетители с хорошим зрением ясно видят, какой фрагмент текста соответствует каждой метке. Однако необходимо проделать большую работу, чтобы при озвучивании взаимосвязей созданный Web-сайт был понятен тем, кто не может воспользоваться визуальным представлением Web-страниц.

Совет

При вложении тега элемента управления внутрь тега `<label>` элемент управления наследует все CSS-стили, примененные к метке. Если вы не хотите, чтобы это случилось, не вкладывайте элемент управления внутрь метки. Выполняйте их разметку раздельно и используйте атрибуты `id` и `for`, чтобы создать связи между ними.

В-третьих, можно применить CSS-стиль к тегам `<label>` и создать специальный внешний вид для этих элементов. Не требуется создавать специальный класс абзаца или заголовка.

Конец рекламы. Обратите внимание, что внутри тега `<label>` вложен тег `<input>` флажка.

```
<label>
  <input type="checkbox" name="humor" value="yes">
  Humor
</label>
```

Это один способ, позволяющий сообщить браузеру о связывании метки с ее элементом управления. Но что, если метки и элементы управления отображаются в разных столбцах таблицы подобно следующему примеру:

```
<form>
  <table>
    <tr>
      <td><input type="checkbox" name="humor" value="yes"></td>
      <td>Humor</td>
```

```
    </tr>
  </table>
</form>
```

Данная разметка делает невозможным вложение тега <input> внутрь тега <label>, поскольку между ними находится пара тегов ячеек таблицы. В таких случаях текст заключается в теги <label>. Затем, чтобы создать связь между элементами, используйте вместо вложения пару атрибутов: атрибут id и атрибут for.

```
<form>
  <table>
    <tr>
      <td>
        <input type="checkbox" name="humor" id="humor" value="yes">
      </td>
      <td>
        <label for="humor">Humor</label>
      </td>
    </tr>
  </table>
</form>
```

Совет

Используйте уникальные ID (идентификаторы) для каждого элемента на странице. Не присваивайте двум или более элементам один и тот же ID.

ЧАСТЬ 7.
Специальные приемы

- Встраивание мультимедиа
- Автоматическое обновление страницы
- Автоматическая переадресация браузера
- Отображение содержимого, выбираемого случайным образом
- Вывод браузером предупреждения
- Отображение всплывающего окна «Вы покидаете сайт»

ГЛАВА 74.
Встраивание мультимедиа

Мультимедиа – одно из тех модных специальных словечек, подобных **контенту** и **функции**, которые настолько широки по значению, что практически бессмысленны. Поэтому лучшее, что можно дать для этого термина – расплывчатое определение. По существу, **мультимедиа** в Интернете – это звуковые файлы, файлы видео, Flash-анимация, SVG-файлы (Scalable Vector Graphics – масштабируемая векторная графика), документы Acrobat и все, что не является изображением, HTML, каскадными таблицами стилей, клиентским или серверным скриптом. В таблице 74.1 перечисляются многие стандартные типы мультимедиа.

Определение

Мультимедиа *(multimedia) – это в той или иной мере компоненты Web-сайта, которые не являются изображением, HTML, каскадными таблицами стилей, клиентскими или серверными скриптами. К распространенным типам мультимедиа относятся звуковые файлы, файлы видео, Flash-анимация, SVG-файлы и документы Acrobat.*

Табл. 74.1. Стандартные типы мультимедиа

Тип мультимедиа	Применение	Расширение файла
Adobe Acrobat (Portable Document Format – Переносимый формат документов)	Документы и формы, насыщенные форматированным текстом	**PDF**
Apple Audio Interchange Format	Звук	**AIFF, AIF**
Apple QuickTime	Потоковое аудио и видео	**MOV**
Macromedia Flash	Анимация, ролики, интерактивные презентации и игры	**SWF**
Macromedia Shockwave	Сложная анимация, клипы, интерактивные презентации и игры	**DCR**
Microsoft Wave	Звук	**WAV**
Microsoft Windows Media	Потоковое аудио и видео	**WMA, WMV**
MIDI (Musical Instruments Digital Interface – Цифровой интерфейс музыкальных инструментов)	Песни, мелодии	**MID**
RealMedia	Потоковое аудио и видео	**RM**
Scalable Vector Graphics	Штриховые иллюстрации, анимации	**SVG**
Sun Audio Format	Звук	**AU**

Существует несколько способов определить, является ли некий файл мультимедийным. Мультимедиа обычно представляется внешними файлами. Это означает, что мультимедийный файл обычно отделен от HTML-документа, который отображает или представляет мультимедиа. Но не всегда. Так, например, SVG-код можно записать непосредственно в код HTML-страницы.

Более важным является то, что для воспроизведения мультимедийных файлов обычно требуются функции, которые не встроены в стандартный браузер. К примеру, сам по себе Internet Explorer не может отображать файлы Acrobat. Посетителю необходимо установить специальное программное обеспечение, называемое подключаемым модулем (plug-in), которое расширяет возможности браузера, предоставляя ему функции, необходимые для воспроизведения документов Acrobat. Однако это практическое правило не всегда верно. Теперь браузер Microsoft Internet Explorer поставляется в стандартной конфигурации с подключаемым модулем Flash-проигрывателя компании Macromedia, предназначенным для просмотра Flash-анимации. Обычный, далекий от техники пользователь компьютера, возможно, не имеет представления о том, что смешную мультипликацию в Интернете в действительности заставляет оживать не его браузер, а отдельное приложение, разработанное не Майкрософт.

Итак, на этом закончим описание того, что собой представляет или не представляет мультимедиа. Простейший способ использования любого из видов мультимедиа на Web-странице состоит в применении тега <embed>. Но, поступая так, будьте осторожны: тег <embed> не является оригинальным HTML-тегом. Он никогда не входил ни в какую спецификацию HTML. Наоборот, это неофициальный тег разметки, который поддерживается в большей или меньшей мере всеми основными браузерами.

В соответствии с политикой стандартов, проводимой консорциумом W3C (World Wide Web Consortium), принято считать, что тега <embed> не существует. А официальной альтернативой, одобренной консорциумом W3C и представленной в спецификации, является тег <object>, который попросту не работает должным образом на разных платформах и в разных браузерах. В действительности, большинство дизайнеров лучше справляются с каскадными таблицами стилей, заставляя их функционировать надлежащим образом, но при этом не могут заставить тег <object> работать с мультимедийным файлом.

И когда тег <object> не работает, он *действительно* не работает. Он создает всяческие странные проблемы, способен вызывать сбои в работе браузера и вызывает их.

При прочих равных условиях, всегда предпочтительней использовать надлежащую разметку. Однако в случае мультимедиа условия не совсем равны. Тег <object> не функционирует должным образом, поэтому не тратьте время на работу с ним. Вместо него *используйте* нестандартный тег <embed>.

Чтобы увидеть тег <embed> в действии, взгляните на рисунок 74.1.

Листинг 74.1. Исходный код страницы, вид которой в браузере показан на рисунке 74.1

```
<body>
  <p>Enjoy this Flash movie!</p>
  <embed src="movie.swf" width="550" height="400"></embed>
</body>
```

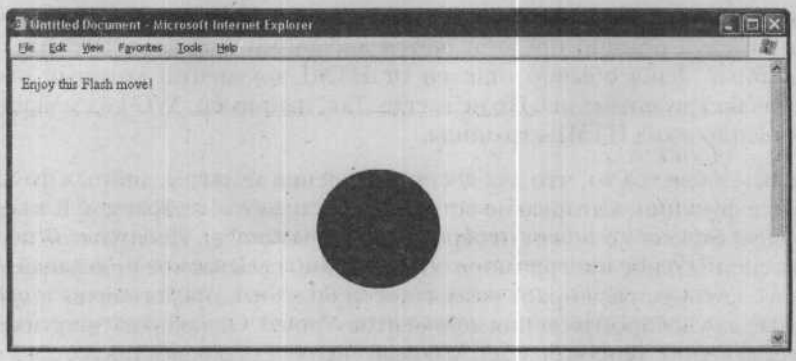


Рис. 74.1. Используйте тег <embed> для ссылки на внешний мультимедийный файл

Как видно из исходного кода Web-страницы, показанной на рисунке 74.1, тег <embed> находится в коде точно в том месте, где должен отображаться мультимедийный файл, точно также, как и тег <img>. Другим сходством между этими тегами является атрибут src, который направляет браузер в место расположения мультимедийного файла. Кроме того, атрибуты width и height определяют размер мультимедийного файла на странице. Таким образом, в некотором роде тег <embed> можно считать тегом <img>, предназначенным для мультимедийных файлов.

Совет

В теге <embed> так же, как и в теге <img>, используется атрибут align. Однако атрибут align работает в теге <embed> не так, как в теге <img>, особенно в горизонтальном направлении, поэтому лучше выравнивать мультимедийные файлы с помощью специальных приемов. Заключите тег <embed> в теги и задайте для атрибута align тега положение мультимедийного файла:

```
<div align="center">
  <embed scr="movie.swf" width="550"
height="400"></embed>
</div>
```

Для достижения лучших результатов в теге <embed> используются некоторые атрибуты, которые в теге <img> не применяются. Предположим, что браузер посетителя не имеет подключаемого модуля Flash-проигрывателя — крайне маловероятный сценарий, поскольку практически любой компьютер в Интернете воспроизводит Flash-ролики. Но только вообразите на мгновение, что случилось невозможное. Без подключаемого модуля (plug-in) браузеру неизвестно, как представлять SWF-файл, и вы получите Web-эквивалент неловкого молчания. Поэтому считается правилом хорошего тона добавление атрибута pluginspage, который направляет браузер на Web-страницу, где посетитель может загрузить и установить необходимый подключаемый модуль и просмотреть мультимедийный файл:

```
<embed src="movie.swf" width="550" height="400"
```

```
    pluginspage="http://www.macromedia.com/go/getflashplayer">
</embed>
```

Совет

*Атрибуты `width` и `height` звукового файла определяют размеры панели управления подключаемого модуля (с кнопками **Stop** (Стоп), **Play** (Воспроизвести), **Fast Forward** (Перемотка вперед), **Volume** (Громкость) и другими). Если вы не хотите, чтобы панель управления появлялась вообще, присвойте атрибуту `hidden` тега `<embed>` значение `true`.*

В зависимости от типа мультимедийного файла в теге `<embed>` могут указываться другие атрибуты. Так, подключаемый модуль Flash-проигрывателя принимает три значения атрибута `quality`: `low` (низкое), `medium` (среднее) и `high` (высокое). С помощью атрибута `quality` можно установить вариант, используемый по умолчанию:

```
<embed src="movie.swf" width="550" height="400" quality="high"
    pluginspage="http://www.macromedia.com/go/getflashplayer">
</embed>
```

Совет

*Если у вас возникают сомнения относительно правильности адреса страницы загрузки подключаемого модуля, просто приведите ссылку на домашнюю страницу Web-сайта изготовителя подключаемого модуля; например, **http:/www.adobe.com/** или **http://www.macromedia.com/**. Обычно с этого места страницу загрузки достаточно просто найти самостоятельно.*

Точно также, если вы не хотите, чтобы Flash-ролики запускались автоматически при загрузке браузером страницы или если вы не хотите, чтобы воспроизведение выполнялось в цикле, можно задать значения для атрибутов `play` и `loop`:

```
<embed src="movie.swf" width="550" height="400" quality="high"
    play="false" loop="false"
    pluginspage="http://www.macromedia.com/go/getflashplayer">
</embed>
```

В таблице 74.2 приводятся стандартные атрибуты тега `<embed>` для управления различными видами мультимедийных файлов. Для некоторых типов мультимедиа в некоторых браузерах эти атрибуты не всегда работают так, как описано, поэтому будьте готовы к использованию старого доброго метода проб и ошибок.

Табл. 74.2. Стандартные атрибуты тега <embed>

Атрибут	Управляемые элементы	Возможные значения	Примечания
autoplay	Определяет, будет ли автоматически воспроизводиться мультимедийный файл при загрузке браузером страницы	true, false	Не применяется для Flash-роликов
height	Размер по вертикали мультимедийного файла в пикселах	Любое числовое значение	Нет
hidden	Определяет, будет ли появляться панель управления на Web-странице	true, false	Используется для звуковых файлов
loop	Определяет, будет ли мультимедийный файл воспроизводиться в цикле	true, false	Нет
play	Для Flash-роликов; определяет, будет ли автоматически воспроизводиться мультимедийный файл при загрузке браузером страницы	true, false	Для Flash-роликов
playcount	Количество воспроизведений мультимедийного файла	Любое числовое значение	Только для Internet Explorer
quality	Визуальная разборчивость содержимого мультимедийного файла	low, medium, high	Для Flash-роликов
volume	Уровень громкости звука мультимедийного файла	1-100	Нет
width	Горизонтальный размер в пикселах мультимедийного файла	Любое числовое значение	Нет

Атрибут loop в Netscape выполняет ту же функцию, что и playcount в Internet Explorer если установить для него число вместо значений true или false. Однако, по-видимому, Flash-ролики не реагируют на числовое значение атрибута loop в Netscape.

Часто задаваемые вопросы

Как узнать, какие атрибуты работают с моим мультимедийным файлом?

Краткий ответ: для всех случаев узнать нельзя. Документация по этому вопросу поверхностна, поэтому определение специальных атрибутов, которые управляют мультимедийным файлом часто связано с использованием метода проб и ошибок. К примеру, хотя для Flash-ролика используется атрибут play, управляющий включением и выключением воспроизведения, для звуковых файлов применяется атрибут autoplay, который выполняет ту же функцию. В идеале, следует найти Web-страницу, на которой встроен такой же тип мультимедийного файла, как у вас, и просмотреть исходный код, чтобы увидеть, как все работает.

ГЛАВА 75.
Автоматическое обновление страницы

Обычно для перезагрузки текущей Web-страницы посетитель должен щелкнуть в браузере на кнопке **Refresh** (Обновить). Однако для вас, дизайнера, существует простой способ встроить на страницу цикл автоматического обновления так, чтобы браузер перезагружал текущий документ без каких-либо действий со стороны посетителя. Этот прием будет полезным, если ваша Web-страница содержит сведения, зависящие от времени или изменяемые в реальном времени, как, например, текущая погода, котировки акций или уровни складских запасов.

Определение

*Считается, что информация **изменяется в реальном времени**, если она отражает текущее, с точностью до секунд, состояние, будь то число единиц продукции на складе, цена акций или государственный долг.*

Скажем, существует Web-страница, которая генерирует и отображает выбор счастливого номера для посетителей, как на рисунке 75.1. Операции выбора обрабатываются простой функцией JavaScript, называемой `doLucky()`.

Теперь допустим, что вы хотите избежать судебного иска и решили изменить свою Web-страницу так, чтобы браузер отображал счастливые номера посетителей только в течение трех секунд, полагая, что посетители вряд ли докажут обман, если набор счастливых номеров пригоден только в течение трех секунд каждый. По существу, здесь налицо ситуация с информацией, изменяющейся в реальном времени. Чтобы поддерживать правильность страницы, ее необходимо обновлять каждые три секунды.

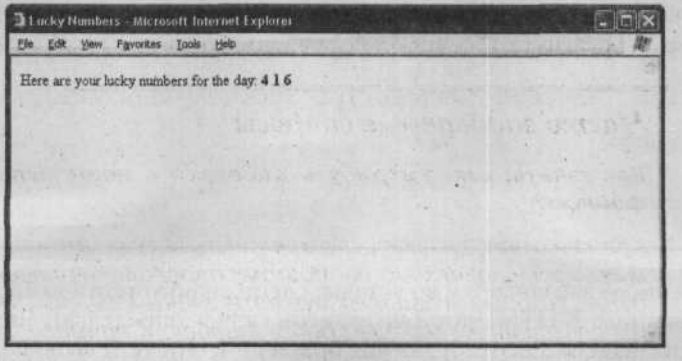


Рис. 75.1. На этой Web-странице вызывается три раза простая функция JavaScript, чтобы сгенерировать для посетителей счастливые номера на день

Листинг 75.1. Исходный код страницы, вид которой в браузере показан на рисунке 75.1

```
<html>
  <head>
    <title>Lucky Numbers</title>
    <script language="JavaScript">

/* Эта функция выбирает случайное число между 0 и 9 и записывает
его на страницу. */

      function doLucky() {
        var x = Math.round(Math.random() * 9);
        document.write(x);
      }

    </script>
  </head>
  <body>

/* В следующем абзаце функция doLucky() вызывается три раза для
записи на страницу трех случайных чисел. */

    <p>Here are your lucky numbers for the day: <strong>
      <script language="JavaScript">doLucky();</script>
      <script language="JavaScript">doLucky();</script>
      <script language="JavaScript">doLucky();</script>
      </strong>
  </body>
</html>
```

Простейший способ заключается во вставке специального тега <meta> в раздел заголовка Web-страницы. В HTML-документе теги <meta> определяют информацию, которая не предназначена для отображения браузером. Эти теги выполняют различные функции. Чаще всего они включают ключевые слова или описание страницы, используемые поисковыми системами. Но вы можете использовать определенный тип тега <meta> для включения функции обновления, как в коде Web-страницы, показанной на рисунке 75.2.

Листинг 75.2. Исходный код страницы, вид которой в браузере показан на рисунке 75.2

```
<html>
  <head>
    <title>Lucky Numbers</title>
    <meta http-equiv="refresh" content="3,#">
    <script language="JavaScript">

/* Функция doLucky() располагается здесь. */

    </script>
  </head>
  <body>
    <p>Here are your lucky numbers for the next three seconds:
      <strong>
      <script language="JavaScript">doLucky();</script>
      <script language="JavaScript">doLucky();</script>
      <script language="JavaScript">doLucky();</script>
      </strong>
  </body>
</html>
```

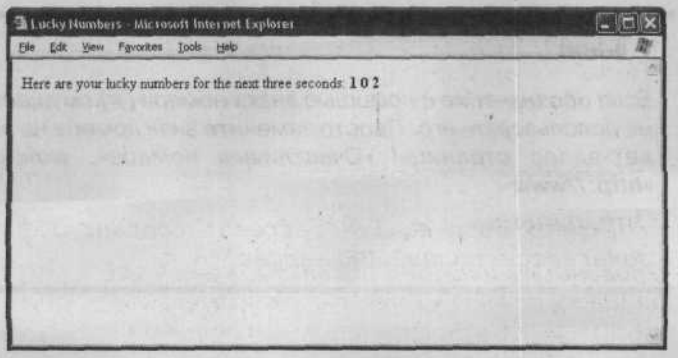


Рис. 75.2. Используйте тег <meta> для передачи браузеру команды на перезагрузку страницы каждые три секунды

Часто задаваемые вопросы

Где располагаются теги <meta>?

Теги <meta> принадлежат разделу заголовка Web-страницы. Не помещайте их в основной раздел, заключенный между тегами <body>.

Так же, как атрибут type тега <input> определяет тип элемента управления формы, отображаемый браузером, так атрибут http-equiv тега <meta> играет роль волшебника. Разные значения атрибута http-equiv создают разные виды тегов <meta>. Задавая для атрибута http-equiv значение refresh, вы превращаете тег <meta> в средство автоматического обновления страницы.

Частота обновления браузером страницы зависит от значения атрибута content. В исходном коде Web-страницы, показанной на рисунке 75.2, тег <meta> имеет следующий вид:

```
<meta http-equiv="refresh" content="3,#">
```

Значение «3» в атрибуте content определяет трехсекундный цикл обновления. Другими словами, браузер перезагружает страницу каждые три секунды. Символ # в атрибуте content является стенографическим знаком для браузера, обозначающим текущую страницу. Данное обозначение использовалось в предыдущих главах для создания ссылающихся на себя ссылок. Итак, на обычном языке значение атрибута content можно было бы выразить приблизительно следующей фразой: «Каждые три секунды перезагружать текущую страницу».

Не существует такого правила, которое бы определяло, что страница должна перезагружаться именно каждые три секунды. Для более длительного цикла обновления увеличьте число секунд:

```
<meta http-equiv="refresh" content="60,#">
```

Этот тег <meta> вызывает перезагрузку страницы через каждую минуту.

Совет

Если обозначение с помощью знака номера (#) смущает вас, можете не использовать его. Просто замените знак номера на полный интернет-адрес страницы «Счастливые номера», включив префикс ***«http://www»****:*

```
<meta http-equiv="refresh" content=3, [здесь располагается полный URL-адрес]">
```

ГЛАВА 76.
Автоматическая переадресация браузера

В главе 75 было показано, как использовать тег <meta> для задания цикла автоматического обновления Web-страницы. Синтаксис тега <meta> выглядел следующим образом:

```
<meta http-equiv="refresh" content="x,#">
```

где **x** соответствует числу секунд, через которые браузер обновляет страницу, а знак # — стенографический знак, обозначающий для браузера текущую страницу.

У вас есть возможность изменять как значение **x**, так и Web-страницу, которая загружается браузером. Достаточно указать другой URL-адрес вместо знака порядкового номера (#).

Часто задаваемые вопросы

Если для переадресации используется тег <meta>, приспособленный для обновления страницы, значит ли это, что переадресация будет выполняться циклически?

Необязательно. После перехода браузера на новую страницу тег <meta> предыдущей страницы больше не действует. Если на новой странице нет переадресации, браузер остается на ней. Конечно, если на новой странице есть переадресация, браузер ожидает указанное количество секунд и снова выполняет переход.

В действительности этот прием создает страницу, которая автоматически переадресует браузер в новое место по истечении указанного количества секунд. Такого типа страницы удобно иметь. Скажем, вы только что переработали и переупорядочили свой сайт, и некоторые старые категории навигации больше не существуют в новой, улучшенной версии. Но посетители вашего сайта могут иметь закладки на старые страницы. Чтобы аудитория вашего сайта не испытывала разочарование при попытке загрузить несуществующий документ, можно было бы вставить на место прежней, устаревшей страницы замещающую страницу, подобную той, что показана на рисунке 76.1. Получается очень классно.

Листинг 76.1. Исходный код страницы, вид которой в браузере показан на рисунке 76.1

```
<html>
  <head>
    <title>Uh, oh!</title>
    <meta http-equiv="refresh" content="5, http://newpage.htm">
  </head>
  <body>
```

```
    <h1>Uh, oh!</h1>
    <p>The page you're looking for no longer exists. Your browser
will automatically jump to the new page in five seconds. Please
update your bookmarks.</p>
  </body>
</html>
```

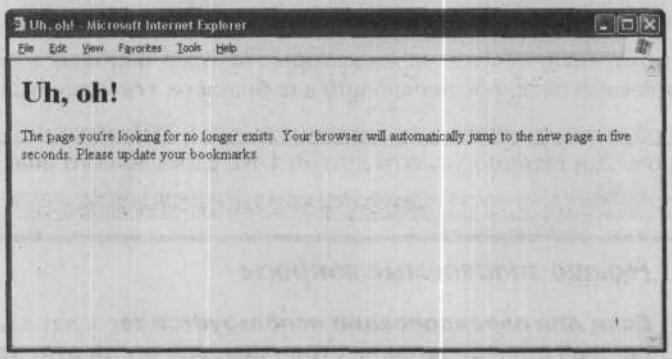


Рис. 76.1. Используйте страницу, подобную этой, для переадресации посетителей в новое место

Тег `<meta>` в исходном коде Web-страницы, показанной на рисунке 76.1, работает так же, как тег `<meta>`, приведенный в начале этой главы. Значение «5» атрибута `content` определяет пятисекундную паузу, а URL-адрес сообщает браузеру, куда необходимо перейти по истечении пяти секунд.

Часто задаваемые вопросы

Какие типы пути можно использовать для URL-адреса тега `<meta>`?

В URL-адресе тега `<meta>` может применяться любой тип пути: абсолютный, относительный, указываемый относительно документа или относительно корня.

Пять секунд, возможно, будет слишком большим периодом ожидания для большинства посетителей вашего сайта. Тем не менее, вы хотите предоставить тем немногим, кто действительно читает содержимое вашего сайта, возможность увидеть, о чем сообщается на странице, прежде чем браузер выполнит переход. В этом случае, возможно, пяти секунд будет мало. Как удовлетворить требования обеих категорий посетителей?

Просто. Увеличьте задержку переадресации приблизительно до десяти – пятнадцати секунд и включите прямую ссылку на новую страницу, как на рисунке 76.2.

Листинг 76.2. Исходный код страницы, вид которой в браузере показан на рисунке 76.2

```
<html>
  <head>
```

```
    <title>Uh, oh!</title>
    <meta http-equiv="refresh" content="10, http://newpage.htm">
  </head>
  <body>
    <h1>Uh, oh!</h1>
    <p>The page you're looking for no longer exists. Your browser
will automatically jump to the <a href="http://newpage.htm">new
page</a> in ten seconds. Please update your bookmarks.</p>
  </body>
</html>
```

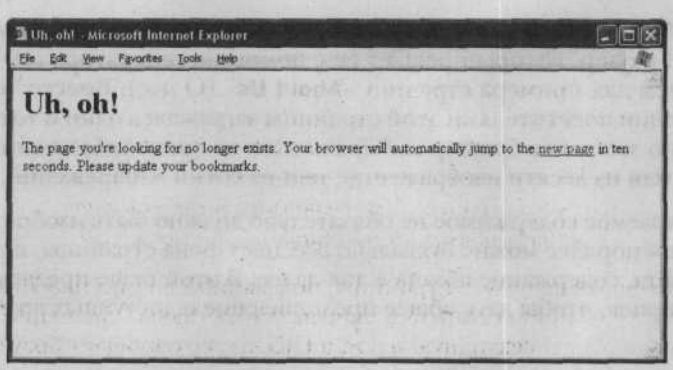


Рис. 76.2. Чтобы предоставить нетерпеливым посетителям возможность быстрого перехода на новую страницу, включите прямую ссылку на измененную Web-страницу

Совет

*В любых ситуациях ссылки «**Click Here**» (Щелкните здесь) представляют неудачное решение, хотя, по-видимому, они нередко встречаются на страницах переадресации. Когда вы включаете прямую ссылку на новую страницу, выбирайте текст ссылки из содержимого существующего абзаца или создайте специальную ссылку, взяв для нее название новой страницы.*

ГЛАВА 77.
Отображение содержимого, выбираемого случайным образом

Общая проблема для многих Web-сайтов состоит в следующем: в то время как основным разделам сайта уделяется немало внимания и выполняется регулярное обновление их содержимого, второстепенные разделы отличаются запущенностью. Пусть это обстоятельство не расстраивает вас. Возможно, ваши ресурсы ограничены, и необходимо расставлять приоритеты. Списки ваших товаров должны быть точными и ваши новости необходимо постоянно обновлять. В условиях цейтнота изменение страницы «**About Us**» (О нас) будет находиться внизу списка актуальных задач.

Один из способов оживления заброшенной Web-страницы состоит в возложении данной задачи на браузер, который решает ее с помощью генератора случайного содержимого. Возьмем для примера страницу «**About Us**» (О нас). Вместо того чтобы при каждом посещении посетителями этой страницы загружался один и тот же файл изображения, можно заставить браузер выбирать одно из трех изображений, или из пяти изображений, или из десяти изображений, или из сотни изображений.

Случайно выбираемое содержимое не обязательно должно быть изображением. Выбирать в случайном порядке можно буквально все: цвет фона страницы, шрифт, стиль начертания шрифта, содержание абзаца и так далее. В этой главе предлагается несколько инструментариев, чтобы дать общее представление о доступных возможностях.

Инструментарий

Генератор случайного изображения

Код данного инструментария выбирает случайным образом из предопределенного набора изображение и отображает его на Web-странице. Код состоит из двух частей. Первая часть представляет собой функцию JavaScript, которая фактически и выбирает случайным образом изображение. Вторая часть – это строка кода, которая вставляется в тело Web-страницы. Этот код помещается в HTML-код в том месте, где должно появляться выбранное случайным образом изображение.

Часто задаваемые вопросы

Можно ли в условиях стесненности в средствах использовать этот инструментарий в качестве сервера рекламы?

Несомненно. Можно попеременно выводить несколько возможных рекламных объявлений вверху своей страницы. При каждой загрузке страницы браузер будет выбирать случайным образом рекламное объявление из доступного набора баннеров.

Далее приводится функция JavaScript:

```
<script language="JavaScript">

/* Если данный скрипт добавляется во внешний JS-
файл, теги <script>, вставленные в начале и в кон-
це этого кода, можно отбросить. Если данный скрипт
вставляется внутрь HTML-документа, не удаляйте те-
ги <script> и поместите весь этот код в раздел за-
головка страницы. */

function doRandomImage() {

/* Следующие строки кода реализуют выбор случайно-
го числа между 1 и 5. При необходимости измените
код так, чтобы он соответствовал точному числу слу-
чайных изображений в вашем наборе. Применяется сле-
дующая формула: Math.round(Math.random() * [число
изображений - 1]) + 1. Таким образом, для выбора
из десяти возможных изображений применяется форму-
ла Math.round(Math.random() * 9) + 1. Подобным об-
разом, для выбора из двух возможных изображений
формула принимает вид Math.round(Math.random() *
1) + 1. */

  var x = Math.round(Math.random() * 4) + 1;

/* Следующая строка кода инициализирует переменную
image, которая будет содержать HTML-код, выводящий
изображение на странице. */

  var image;

/* Следующие блоки кода if/then проверяют значение
x и в соответствии с ним устанавливают значение пе-
ременной image. На каждое возможное изображение
приходится один блок кода if/then. И если у вас
есть только три возможных изображения, потребуется
всего три блока кода if/then. */

  if (x == 1) {

    image = "<img src=\"src01\" width=\"width01\"
height=\"height01\" alt=\"alt01\">";

  }

/* Замените переменную src01 на путь к первому слу-
чайному изображению. Подобным образом замените
width01, height01 и alt01 на ширину, высоту и аль-
тернативный текст данного изображения. Случайные
изображения могут быть разного размера, но вид
страницы будет лучше, когда физические размеры всех
случайных изображений одинаковы.
```

```
Обратите внимание, что внутренним кавычкам предше-
ствует обратная косая черта (\). Это символ выхо-
да. Он сообщает браузеру, что кавычки являются
частью значения переменной. */
  if (x == 2) {

    image = "<img src=\"src02\" width=\"width02\"
height=\"height02\" alt=\"alt02\">";
  }

  if (x == 3) {
    image = "<img src=\"src03\" width=\"width03\"
height=\"height03\" alt=\"alt03\">";
  }

  if (x == 4) {
    image = "<img src=\"src04\" width=\"width04\"
height=\"height04\" alt=\"alt04\">";
  }

  if (x == 5) {
    image = "<img src=\"src05\" width=\"width05\"
height=\"height05\" alt=\"alt05\">";
  }
/* В следующей строке браузер записывает HTML-код
в переменную image на Web-странице. */
  document.write(image);

}

</script>
```

Это была функция. Теперь необходим код для ее вызова. Поместите следующую строку HTML-кода точно в то место, где должно появляться случайное изображение.

```
<script    language="JavaScript">doRandomImage();
</script>
```

Совет

Если случайное изображение реагирует на щелчки мыши и служит ссылкой, по которой переход всегда выполняется в одно и то же место, независимо от изображения, поместите HTML-код, вызывающий функцию, между тегами-анкерами:

```
<a href="targetpath"><script
language="JavaScript">doRandomImage();</script></a>
```

Однако если пункт назначения ссылки зависит от того, какое случайное изображение выводится браузером, встройте тег-анкер в значение переменной `image` *в функции* `doRandomImage()`*:*

```
if (x == 1) {

   image = "<a href=\"targetpath01\"><img
src=\"src01\" width=\"width01\" height=\"height01\"
alt=\"alt01\"></a>"

}
```

Инструментарий

Генератор случайной цитаты

Код данного инструментария выбирает случайную цитату и отображает ее на Web-странице. Как и в предыдущем инструментарии код состоит из двух частей: самой функции JavaScript и вызова этой функции с Web-страницы.

Далее приводится функция JavaScript:

```
<script language="JavaScript">

/* Если данный скрипт добавляется во внешний JS-
файл, теги <script>, вставленные в начале и в кон-
це этого кода, можно отбросить. Если данный скрипт
вставляется внутрь HTML-документа, не удаляйте те-
ги <script> и поместите весь этот код в раздел за-
головка страницы. */

function doRandomQuote() {

/* Следующие строки кода реализуют выбор случайно-
го числа между 1 и 5. При необходимости измените
код так, чтобы он соответствовал точному числу слу-
чайных цитат в вашем наборе. Применяется следующая
формула: Math.round(Math.random() * [число цитат -
1]) + 1. */

  var x = Math.round(Math.random() * 4) + 1;

/* Следующая строка кода инициализирует переменную
quote, которая будет содержать текст цитаты. */

  var quote;
```

```
/* Следующая строка кода инициализирует переменную
tag. Задайте в качестве значения этой переменной
тип тега, который хотите использовать для разметки
цитаты: p, blockquote, h1, любой другой. */

  var tag = "blockquote";

/* Следующая строка кода инициализирует переменную
source. Значение этой переменной будет равно имени
автора цитаты, вместе с названием книги, речи или
статьи, где упоминалась цитата. */

  var source;

/* Следующая строка кода инициализирует переменную
code, которая будет содержать блок HTML-кода, вы-
водящий цитату на экран. */

  var code;

/* Следующие блоки кода if/then проверяют значение
x и в соответствии с ним устанавливают значение пе-
ременной quote. Для каждой цитаты требуется один
блок кода if/then.

Если вы не хотите выводить на экран источник цита-
ты, используйте вместо соответствующей строки кода
следующее выражение: source = ""; */

  if (x == 1) {

    quote = "Цитата 1 размещается здесь.";

      source = "Строка источника 1 размещается
здесь.";

  }


  if (x == 2) {

    quote = "Цитата 2 размещается здесь.";

      source = "Строка источника 2 размещается
здесь.";

  }


  if (x == 3) {

    quote = "Цитата 3 размещается здесь.";

      source = "Строка источника 3 размещается
здесь.";

  }
```

```
  if (x == 4) {
    quote = "Цитата 4 размещается здесь.";
      source = "Строка источника 4 размещается
здесь.";
  }

  if (x == 5) {
    quote = "Цитата 5 размещается здесь.";
      source = "Строка источника 5 размещается
здесь.";
  }
/* Теперь браузер начинает формировать HTML-код для
записи на страницу. */
  code = "<" + tag + ">";
/* Если вы не хотите отображать левую кавычку, уда-
лите следующую строку кода. */
  code += "&#8220;";
/* Далее следует текст цитаты. */
  code += quote;
/* Если вы не хотите отображать правую кавычку,
удалите следующую строку кода. */
  code += "&#8221;";

/* Следующая строка кода добавляет тег <cite> для
источника цитаты. Даже если вы не хотите отображать
источник, во всяком случае оставьте эту строку ко-
да. */
  code += "<cite>";
/* Этот блок кода if/then проверяет, определена или
нет переменная source. Если переменная определена,
добавляется дефис между текстом цитаты и текстом
источника.
Если вы не хотите отображать дефис между текстом
цитаты и текстом источника, удалите этот блок ко-
да. Или если требуется отобразить не дефис, а дру-
гой знак, замените код дефиса (&#8212) на код тре-
буемого знака, например, на двоеточие (:). */
```

```
  if (source.length != 0) {
    code += "&#8212;";
  }
/* Следующая строка кода добавляет источник, закрывает тег <cite> и закрывает тег основной разметки. */
  code += (source + "</cite></" + tag + ">");
/* В следующей строке браузер записывает всю кодовую строку на Web-страницу. */
  document.write(code);

}

</script>
```

Для вызова этой функции поместите следующий HTML-код точно в то место, где цитата должна появляться на Web-странице.

```
<script    language="JavaScript">doRandomQuote();
</script>
```

Совет

Для получения дополнительного визуального эффекта создайте специальный CSS-стиль класса для своих случайных цитат. Затем в функции `doRandomQuote()` *замените следующую строку кода:*

```
code ="<" + tag + ">"
```

на:

```
code = "<" + tag + "
class=\"имякласса\">"
```

Инструментарий

Генерирование случайных свойств страницы

Для реального переключения код из этого инструментария выбирает случайное правило стиля из таблицы стилей и применяет его к тегу <body> на странице. Ваши случайные правила стилей могут содержать практически любое форматирование, включая цвета фона, цвета ссылок, гарнитуры шрифтов и стили шрифта.

Код этого инструментария состоит из трех частей. Первая часть представляет собой набор правил стилей, из которых производится выбор. Вторая часть кода – это функция JavaScript, которая выбирает стиль. И третьей частью является строка HTML-кода, которая записывает измененный тег <body> на вашу Web-страницу.

Здесь описываются правила стилей:

```
<style type="text/css">

/* Если правила стилей находятся во внешнем CSS-
файле, теги <style> не нужны. Если правила стилей
вставляются внутрь HTML-файла, не удаляйте теги
<style> и поместите весь блок каскадных таблиц сти-
лей в раздел заголовка страницы.

Можно включать любое количество правил стилей, при-
чем можно добавлять любые определения стилей. Сле-
дующие блоки кода создают три разных правила сти-
лей. Обратите внимание, что каждый блок кода явля-
ется специальным стилем класса тега <body>. */

  body.style01 {

    background-color: #000000;
    color: #FFFFFF;
    font-family: Arial, Helvetica, sans-serif;

  }

  body.style02 {

    background-color: #FFFF00;
    color: #FF0000;
    font-family: "Times New Roman", Times, serif;
```

```
  }
  body.style03 {
    background-color: #FFFFFF;
    color: #000000;
    font-family: "Courier New", Courier, mono;

  }

</style>
```

Далее приводится функция JavaScript:

```
<script language="JavaScript">

/* Если данный скрипт добавляется во внешний JS-
файл, теги <script>, вставленные в начале и в кон-
це этого кода, можно отбросить. Если данный скрипт
вставляется внутрь HTML-документа, не удаляйте те-
ги <script> и поместите весь этот код в раздел за-
головка страницы, или до, или после таблицы стилей.
*/

function doRandomProperties() {

/* Следующие строки кода реализуют выбор случайно-
го числа между 1 и 2. При необходимости измените
код так, чтобы он соответствовал количеству правил
стилей в вашем наборе. Применяется следующая фор-
мула: Math.round(Math.random() * [количество пра-
вил стилей - 1]) + 1. */

  var x = Math.round(Math.random() * 2) + 1;

/* Следующая строка кода инициализирует переменную
style, которая будет содержать имя стиля класса,
используемого браузером. */

  var style;

/* Следующие блоки кода if/then проверяют значение
x и в соответствии с ним задают значение перемен-
ной style. Для каждого правила стиля требуется один
блок кода if/then. */

  if (x == 1) {
    style = "style01";
```

```
  }
  if (x == 2) {
    style = "style02";
  }
  if (x == 3) {
    style = "style03";
  }
/* В следующей строке браузер записывает тег <body>
на Web-страницу. Снова обратите внимание на исполь-
зование символа выхода (\), указывающего, что ка-
вычка принадлежит HTML-коду, а не выражению
JavaScript. */
  document.write("<body class=\"" + style +"\">");


}


</script>
```

Для вызова этой функции добавьте непосредственно после открывающего тега <body> следующую строку HTML-кода:

```
<script
language="JavaScript">doRandomProperties();</scrip
t>
```

Как правило, основной раздел страницы, заключенный в теги <body>, выглядит подобно следующему коду:

```
  <body>
    <script
language="JavaScript">doRandomProperties();</scrip
t>
    <!- Содержимое страницы располагается здесь -
>


  </body>
```

Часто задаваемые вопросы

Не создается ли при применении этого метода два тега <body>?

Очень проницательный вопрос. Действительно, Web-страница завершается двумя открывающими тегами <body>: исходный, который присутствует в HTML-коде, и еще один, который записывается функцией doRandomProperties(). Апологеты «правильного» кодирования не смогут смириться с такой практикой, поскольку Web-страница, согласно их воззрениям, должна содержать один и только один открывающий тег <body>. Но все работает.

Возможно, вы удивляетесь, почему бы просто не заменить исходный тег <body> вызовом функции doRandomProperties(). Если вы хотите, можете это сделать, но, скорее всего, этого делать не стоит. Устройства просмотра, не поддерживающие JavaScript, могут столкнуться с трудностями при интерпретации HTML-кода, в котором отсутствует явно заданный тег <body>.

ГЛАВА 78.
Вывод предупреждения браузера

Одна из великих целей Web-дизайна — создание полностью прозрачного сайта, сайта, который работает на любом мыслимом устройстве просмотра, на всех возможных платформах без принесения в жертву интерактивных возможностей и неотразимого внешнего вида, которые привыкли ожидать от Интернета его пользователи.

Это мечта недостижима по вполне понятной причине: маловероятно, что вы реализуете ее в этой жизни. Просто браузеры отличаются большим разнообразием, а платформы несовместимы друг с другом. Компьютерная технология в целом защищена множеством патентов, и коммерческие секреты, рыночные доли, программы лицензирования, сообщества, организации провайдеров и старая добрая экономия, основанная на жадности, гарантируют, что она такой и останется.

В свете этого Web-дизайнеры научились управлять своими ожиданиями. Вместо заоблачных целей они ориентируются на определенную аудиторию, выполняя разработки под браузеры, которыми пользуется большинство посетителей их сайтов. Они также отличаются небольшим коварством, искусно склоняя консервативную и инертную часть своей аудитории к выполнению обновления программного обеспечения до новейших версий браузеров. Трудно вообразить, чтобы кто-нибудь получал какое-либо удовольствие от путешествий по Интернету с помощью Internet Explorer 4.0 или Netscape Navigator 3.0 (тогда, когда он назывался «Навигатором», а не просто Netscape), хотя некоторые люди отказываются от обновления своего оборудования.

Совет

Чтобы излишне не раздражать посетителей своего сайта, обычно лучше всего размещать уведомление о версии браузера на домашней странице. Не добавляйте его на каждую страницу своего сайта.

Уведомление о версии браузера выполняет две задачи. Во-первых, быстро оповещает посетителей, работающих с устаревшим программным обеспечением, что ваш сайт может не работать так, как они ожидали, а вы планировали. Во-вторых, уведомление о версии браузера предназначено для тех, кто, увидев его, поймет, что настало время обновить программное обеспечение до новейшей версии их любимого бесплатного браузера. Уведомление поощряет к таким действиям посредством постоянного напоминания, генерирования всплывающего окна. Среди посетителей, пользующихся старыми версиями браузеров, много тех, кто не выносит всплывающих окон.

Инструментарий в этой главе предоставляет завершенную, готовую к использованию функцию уведомления о версии браузера, которая проверяет модель браузера посетителя, сравнивает номер версии с версией, рекомендуемой для сайта, и выводит предупреждающее сообщение для тех посетителей, чьи браузеры не соответствуют предъявляемым требованиям.

Часто задаваемые вопросы

Возникает подозрение, что существует более простой способ получения версии и модели браузера, который не требует просмотра и фильтрации множества ненужных данных. Нет ли в языке JavaScript более точных свойств объекта navigator?

Да, JavaScript предлагает более точные свойства объекта `navigator`*, такие, как* `appCodeName`*,* `appName` *и* `appVersion`*, но они часто возвращают дезориентирующие или некорректные результаты. Самая точная информация содержится в свойстве* `userAgent`*. К сожалению, свойство* `userAgent` *содержит также немало лишней информации, которая не нужна для функции вывода уведомления о версии браузера, поэтому прежде чем перейти к самому сравнению версии браузера посетителя и целевой версии, под которую разработан сайт, необходимо избавиться от посторонних данных.*

Работа функции начинается с получения свойства `userAgent` JavaScritp-объекта `navigator`. Свойство `userAgent` содержит длинную строку данных о браузере посетителя. Проверьте данное свойство посредством создания следующего HTML-документа и его загрузки в разные браузеры:

```
<html>
  <head>
    <title>userAgent Test</title>
  </head>
  <body>
    <script language="JavaScript">
      document.write(navigator.userAgent);
    </script>
  </body>
</html>
```

Браузер Microsoft Internet Explorer 6.0 для Windows возвращает следующие результаты:

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR
1.0.3705)
```

тогда как браузер Netscape 7.1 для Windows возвращает такие данные:

```
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.4)
Gecko/20030624 Netscape/7.1 (ax)
```

а браузер Opera версии 7.23 выводит:

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) Opera 7.23 [en]
```

Функция выводимого браузером предупреждения анализирует это свойство для определения версии и типа браузера, используемого посетителем. Как видите, в данных свойства `userAgent` содержится гораздо больше информации, чем просто название браузера и номер версии, поэтому функция должна отфильтровывать посторонние сведения, получая на входе следующую строку:

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) Opera 7.23 [en]
```

и выдавая конечный результат:

```
Opera 7.23
```

Комментарии, включенные в инструментарий, поясняют все шаги выполняемой процедуры:

Инструментарий

Вывод браузером предупреждения

Для использования данного предупреждения браузера скопируйте следующий код в раздел заголовка HTML-документа или удалите теги <script> и поместите функцию во внешний файл JavaScript.

```
<script language="JavaScript">

/* Теги <script> в начале и в конце данного скрипта не нужны, если эта функция помещается в отдельный файл JavaScript. */

  function doBrowserCheck() {

/* Следующий блок кода инициализирует ряд переменных.

В переменной userBrowser хранится содержимое свойства userAgent объекта navigator, которое среди прочего предоставляет название и версию браузера посетителя.

Переменная pass является логическим (true/false) переключателем. Когда переменная pass принимает значение true, браузер посетителя удовлетворяет минимальным требованиям, предъявляемым к браузеру. В начале функцией предполагается, что значение этой переменной — true.

Переменные versionStart и versionEnd будут содержать начальную и конечную позиции номера версии внутри строки userBrowser. По существу, извлекается данная часть строки для сравнения с целевой версией, на которую ориентировано содержимое сайта. */

  var userBrowser = navigator.userAgent;

  var pass = true;

  var versionStart;

  var versionEnd;

/* Следующие три блока кода предоставляют целевые версии трех браузеров: Internet Explorer, Netscape и Opera. Текущими версиями являются 5.5, 6.0 и 5.0, соответственно. При необходимости вы можете скорректировать эти значения для своего сайта.
```

```
Переменные versionMSIE, versionNetscape и
versionOpera будут содержать номера версий, кото-
рые извлекаются из строки userBrowser. */
  var targetMSIE = 5.5;
  var versionMSIE;

  var targetNetscape = 6;
  var versionNetscape;

  var targetOpera = 5;
  var versionOpera;
/* Следующий блок кода if/then реализует проверки
для браузера Microsoft Internet Explorer. */
  if (userBrowser.indexOf("MSIE") != -1 &&
userBrowser.indexOf("Opera") == -1) {
/* Приведенная выше строка кода реализует сканиро-
вание содержимого переменной userBrowser на нали-
чие текста "MSIE". Если переменная userBrowser со-
держит строку "MSIE" и одновременно не содержит
строки "Opera" (поскольку свойство userAgent брау-
зера Opera также включает текст "MSIE"), функцией
делается вывод, что посетитель пользуется браузе-
ром Microsoft Internet Explorer. */
  versionStart = userBrowser.indexOf("MSIE") + 5;
/* Приведенная выше строка кода реализует поиск на-
чальной позиции номера версии внутри строки
userBrowser, которая находится на месте пятого зна-
ка справа от буквы "M" в строке "MSIE". */
  versionEnd = userBrowser.indexOf(";",
versionStart);
/* Приведенная выше строка реализует поиск конеч-
ной позиции номера версии внутри строки
userBrowser, в которой находится знак "точка с за-
пятой". */
  versionMSIE = userBrowser.substring
(versionStart, versionEnd);
/* Приведенная выше строка реализует извлечение но-
мера версии из строки userBrowser и помещает его в
переменную versionMSIE. */
  if (versionMSIE < targetMSIE) {
    pass = false;
  }
}
```

```
/* Приведенный выше блок кода if/then сравнивает
версию Internet Explorer посетителя с целевой вер-
сией. Если номер версии браузера посетителя меньше
целевой версии, переменной pass присваивается зна-
чение false.

Следующий блок кода if/then реализует проверки
для браузера Netscape. */

  if (userBrowser.indexOf("Netscape") != -1) {

/* Приводимые ниже строки кода реализуют определе-
ние начальной и конечной позиции номера версии
внутри строки userBrowser (с девятого знака вправо
от буквы "N" в "Netscape" до следующего пробела) и
извлечение номера версии в переменную
versionNetscape. */

    versionStart =
userBrowser.indexOf("Netscape") + 9;

    versionEnd = userBrowser.indexOf(" ",
versionStart);

    versionNetscape =
userBrowser.substring(versionStart, versionEnd);

/* И опять, если версия браузера Netscape посети-
теля меньше целевой версии, переменная pass прини-
мает значение false. */

  if (versionNetscape < targetNetscape) {

    pass = false;

  }
}

/* Следующий блок кода реализует проверки для бра-
узера Opera. */

  if (userBrowser.indexOf("Opera") != -1) {

/* Получение номера версии путем извлечения его из
переменной userBrowser. Начальная позиция находит-
ся на месте шестого знака от буквы "O" в "Opera",
а конечная позиция находится на месте следующего
пробела. */

   versionStart = userBrowser.indexOf("Opera") + 6;

   versionEnd = userBrowser.indexOf(" ",
versionStart);

    versionOpera =
userBrowser.substring(versionStart, versionEnd);

/* Если версия браузера Opera посетителя меньше це-
левой версии, переменная pass принимает значение
false. */
```

```
  if (versionOpera < targetOpera) {
    pass = false;
  }
}
/* Следующие блоки кода if/then реализуют проверку, действительно ли значение переменной pass равно false. Если это так, браузером выводится всплывающее сообщение. Можете изменять текст сообщения требуемым образом, но помните о вежливости и старайтесь сформулировать свой запрос так, чтобы подчеркнуть выгоды бесплатного обновления браузера для посетителя. */
  if (!pass) {
    alert("Для повышения качества предоставляемых услуг на этом сайте используются технологии JavaScript и каскадные таблицы стилей. Браузеры старых версий могут не поддерживать эти функции. Для достижения наилучших результатов, пожалуйста, обновите свой браузер до новейшей версии.")
  }

}

</script>
```

Совет

Вы можете не выводить предупреждающее сообщение. Добавьте свои собственные команды JavaScript в блок `if (!pass)`*, чтобы определить действия программы в случае, если браузер посетителя не будет соответствовать требуемой версии.*

Для запуска данного скрипта добавьте событие `onLoad` в тег `<body>` HTML-документа:.

```
<body onLoad="doBrowserCheck();">
```

В этом случае при загрузке браузером страницы будет автоматически выполняться функция `doBrowserCheck()`.

ГЛАВА 79.
Отображение всплывающего окна "Вы покидаете сайт"

Здесь описывается небольшая уловка, которая может оказаться для вас полезной: отображение всплывающего окна «**You Are Now Leaving**» (Вы покидаете сайт), когда посетитель щелкает внешнюю ссылку.

Как Web-дизайнер, вы прикладываете титанические усилия, пытаясь создать физическое восприятие целостности своего Web-сайта. Вы хотите, чтобы он воспринимался как логическое, цельное физическое пространство, как дом со многими комнатами. Пока посетители остаются на сайте, согласованный графический дизайн от страницы к странице и далее прекрасно подкрепляет это ощущение единого пространства. Но страницы Web-сайта не совсем похожи на комнаты в доме. К примеру, ваши посетители могут покидать сайт из любой комнаты, а не только из комнаты с входной и выходной дверьми, и после выхода они могут направляться в буквальном смысле куда угодно. В этом и заключается легендарная нелинейность Интернета.

Когда ваши внешние ссылки опознаются как внешние, раздражающая телепортация из одного «дома» в другой через весь мир становится менее дезориентирующей. Менее вероятно, что у ваших посетителей будет возникать чувство потерянности. Но когда особенности вашего сайта делают различия между внешними и внутренними ссылками менее заметными, вы можете рассмотреть возможность вывода предупреждения для посетителей. Сообщите им, что они сейчас покинут сайт, если щелкнут внешнюю ссылку. Предоставьте им шанс отменить свои действия, если они не хотят выходить с сайта.

Эту задачу решает простая функция JavaScript, подобная той, что приводится в инструментарии данной главы.

Инструментарий

Всплывающее окно «Вы покидаете сайт»

Добавьте эту короткую функцию JavaScript на любую страницу, содержащую внешнюю ссылку.

```
<script language="JavaScript">

/* Теги <script> в начале и в конце данного скрипта не нужны, если эта функция помещается в отдельный файл JavaScript. */

  function doYouAreNowLeaving(url) {

/* Вся функция состоит из одного блока кода if/then. Браузером выводится всплывающее окно подтверждения, которое содержит кнопки OK и Cancel (Отменить). (Текст сообщения находится в вашем полном распоряжении.) Если посетитель щелкает на кнопке OK, браузер переходит на URL-адрес, содержащийся в ссылке, которая вызывает данный скрипт. Если посетитель щелкает на кнопке Cancel (Отменить), скрипт попросту завершается и браузер остается на текущей странице. */
```

```
  if (confirm("Сейчас вы покинете текущий сайт. Для
продолжения нажмите кнопку OK.")) {
    location.href = url;
  }
}

</script>
```

Чтобы заставить функцию работать, необходимо отформатировать внешние ссылки так, чтобы они отличались от внутренних ссылок. Внешние ссылки должны выглядеть подобно следующему примеру:

```
<a href="javascript:doYouAreNowLeaving(scr);"
>Здесь располагается текст внешней ссылки</a>
```

Замените переменную `src` на полный интернет-адрес (включая префикс **http://www**) внешней ссылки и, конечно же, добавьте собственный текст ссылки.

Если вам не нравится использовать ссылки JavaScript из-за проблем совместимости с браузерами старых версий, воспользуйтесь следующим форматом ссылки, который также нормально работает:

```
<a href="#" onClick="doYouAreNowLeaving(scr);"
>Здесь располагается текст внешней ссылки</a>
```

И опять, замените `src` на полный интернет-адрес внешней ссылки.

Совет

Чтобы внешняя ссылка открывалась в новом окне браузера, не добавляйте `target="_blank"` *в тег-анкер ссылки. Этот прием не действует в данном конкретном скрипте, т.к. ссылка формально переходит в JavaScript, а не на Web-сайт. Вместо этого, измените в функции* `doYouAreNowLeaving(url)` *следующую строку кода:*

```
location.href = url;
```

на:

```
open(url);
```

ЧАСТЬ 8.
Базовое обучение

- Создание HTML-документов
- Добавление ключевых слов и описаний страниц
- Блокирование частей сайта для поисковых систем
- Тестирование сайта
- Выбор Web-хоста
- Загрузка сайта на сервер
- Проверка правильности исходного кода

ГЛАВА 80.
Создание HTML-документов

Разработка Web-сайта означает создание HTML-документов. **HTML** расшифровывается как **Hypertext Markup Language** (Язык гипертекстовой разметки). Это компьютерный язык, описывающий содержимое Web-страницы.

Определение

HTML расшифровывается как Hypertext Markup Language (Язык гипертекстовой разметки). Это компьютерный язык, описывающий содержимое Web-страницы.

HTML-документы являются на самом деле простыми текстовыми файлами. Для их создания не требуется изощренное программное обеспечение, достаточно обычного текстового редактора. Где взять текстовый редактор? Он у вас уже есть, даже если вы этого не осознаете. Операционные системы Microsoft поставляются с программой Notepad (Блокнот), которая запускается следующей командой с рабочего стола Windows: **Start ♦ All Programs ♦ Accessories ♦ Notepad** (Пуск ♦ Все программы ♦ Стандартные ♦ Notepad). Операционные системы Mac OS старых версий поставлялись с текстовым редактором SimpleText, тогда как более новые операционные системы Mac OS поставляются с текстовым редактором TextEdit. Все три эти программы выполняют более или менее одинаковые функции, предоставляя возможности записи и сохранения текстовых файлов.

Определение

***Теги** (tags) — ключевые слова HTML, которые идентифицируют структуру текста внутри HTML-файла. **Структура текста** — это попросту описание того, к какому типу элемента принадлежит текст: абзац, заголовок, название изображения и так далее.*

Что отличает HTML-документы от других видов текстовых файлов, так это включение **тегов** или ключевых слов HTML, которые определяют структуру текста внутри файла. **Структура** на техническом жаргоне означает попросту, какой вид элемента представляет собой текст, например, абзац, заголовок или название файла изображения. Можно сказать, что HTML-тег идентифицирует фрагмент текста как специальный элемент на Web-странице.

Допустим, HTML-файл содержит следующую строку текста:

```
Take my wife. Please.
```

HTML-теги сообщают браузеру, к какому типу элементов относится текст. Чтобы сделать текст абзацем, добавьте тег абзаца, как в следующем примере:

```
<p>Take my wife. Please.</p>
```

Часто задаваемые вопросы

Возможности моего текстового редактора ограничены. Следует ли пользоваться им?

Нет. Возможно, у вас на компьютере есть также программа подготовки текстов, такая как Microsoft Word. Текстовые процессоры — это те же текстовые редакторы, хотя они обладают многими расширенными функциями, которые не нужны для создания HTML-файлов. Если же вы предпочитаете работать в текстовом процессоре, используйте его для создания HTML-страниц. Не стремитесь приобрести и не покупайте приложение, специально предназначенное для редактирования HTML-кода.

Тег абзаца, как большинство HTML-тегов, состоит из двух частей: открывающего тега (`<p>`), который размечает начало абзаца, и закрывающего тега (`</p>`), который размечает конец абзаца. Когда браузеру встречается тег абзаца, он воспринимает его, как абзац, и отображает текст со встроенным стилем абзаца, как на рисунке 80.1.

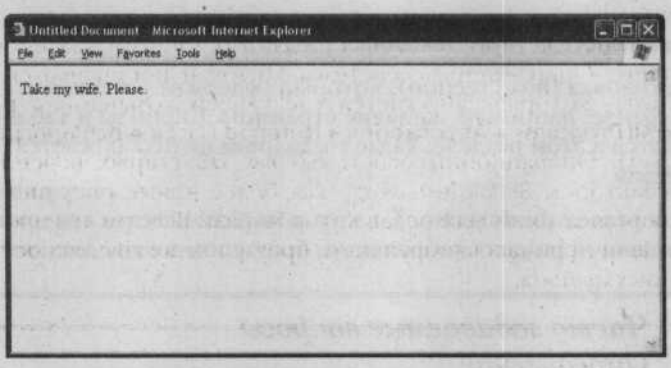


Рис. 80.1. Разметьте фрагмент текста как абзац, и браузер отобразит текст со стилем абзаца

Следует отметить, что при изменении тега изменяется структура текста. А если изменяется структура текста, браузер, возможно, будет по-другому отображать текст. Замените тег абзаца тегом заголовка первого уровня:

```
<h1>Take my wife. Please.</h1>
```

и как видно на рисунке 80.2, браузер отображает текст в виде заголовка первого уровня, а не как абзац.

Совет

*При сохранении HTML-файла не используйте стандартное расширение **.txt** для текстовых файлов. Вместо него устанавливайте расширение HTML-документов: **.htm**.*

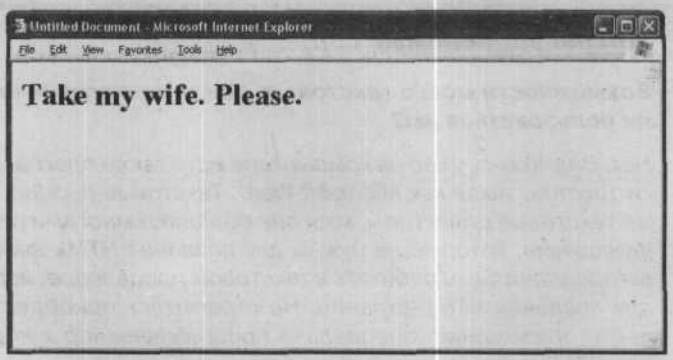


Рис. 80.2. Измените тег абзаца на тег заголовка, и браузер изменит внешний вид текста

Не имеет значения насколько передовым, сложным или изощренным будет HTML-документ, во всех случаях теги работают одинаково. Они идентифицируют структуру текста, которую браузеры интерпретируют в соответствии со своими встроенными стилями.

В HTML-документах существует два явных раздела:

1. Раздел заголовка (head section), который содержит специальную информацию о HTML-файле, например, заглавие страницы. Скрипты и таблицы стилей также находятся в этом разделе. Раздел заголовка не отображается браузером в окне документа.
2. Основной раздел (body section), в котором размещается содержимое Web-страницы. Основной раздел отображается браузером в окне документа.

Часто задаваемые вопросы

Мне не нравится, как браузер отображает название элемента здесь. Кому мне жаловаться?

Разработчику браузера. Однако почему бы не заняться этим самому? Если вам не нравится, как браузер отображает некоторые элементы, можно переопределить их внешний вид с помощью каскадных таблиц стилей (CSS — Cascading Style Sheets). Подробности см. в части IV.

Указанные выше разделы размечаются с помощью тегов `<head>` и `<body>`, подобно следующему примеру:

```
<head>
  <!- Здесь размещается раздел заголовка. ->
</head>
<body>
  <!- Здесь размещается основной раздел. ->
</body>
```

Вернитесь к прежнему примеру с юмористической строкой текста. Поскольку вы хотите, чтобы браузер отображал этот текст внутри окна документа, она помещается в основной раздел страницы:

```
<head>
  <!- Здесь размещается раздел заголовка. ->
</head>
<body>
  <p>Take my wife. Please.</p>
</body>
```

Чтобы включить дополнительный текст на страницу, добавьте дополнительный текст в основной раздел:

```
<head>
  <!- Здесь размещается раздел заголовка. ->
</head>
<body>
  My Favorite Jokes
  We could talk about women.
  <p>Take my wife. Please.</p>
</body>
```

Не забудьте разметить новое содержимое с помощью надлежащих тегов:

```
<head>
  <!- Здесь размещается раздел заголовка. ->
</head>
<body>
  <h1>My Favorite Jokes</h1>
  <p>We could talk about women.</p>
  <p>Take my wife. Please.</p>
</body>
```

Сейчас вы занимаетесь созданием Web-страницы «My Favorite Jokes» (мои любимые шутки). Обратите внимание, что заголовок страницы — **My Favorite Jokes** — находится внутри основного раздела страницы, это означает, что браузер отображает этот текст в окне документа. Если строку **My Favorite Jokes** требуется сделать также официальным названием страницы, необходимо добавить этот текст в раздел заголовка страницы между тегами `<title>`:

```
<head>
  <title>My Favorite Jokes</title>
```

```
</head>
<body>
  <h1>My Favorite Jokes</h1>
  <p>We could talk about women.</p>
  <p>Take my wife. Please.</p>
</body>
```

Название страницы не обязательно должно совпадать с заголовком. Назвать страницу можно как угодно:

```
<head>
  <title>My Humor Site: The Jokes Page</title>
</head>
<body>
  <h1>My Favorite Jokes</h1>
  <p>We could talk about women.</p>
  <p>Take my wife. Please.</p>
</body>
```

В качестве заключительного шага поместите всю страницу между открывающим и закрывающим тегами <html>. Эти теги всего лишь сообщают браузеру, что данная страница является HTML-документом:

```
<html>
  <head>
    <title>My Humor Site: The Jokes Page</title>
  </head>
  <body>
    <h1>My Favorite Jokes</h1>
    <p>We could talk about women.</p>
    <p>Take my wife. Please.</p>
  </body>
<html>
```

Весь документ, сверху донизу выглядит в браузере так, как показано на рисунке 80.3. Итак, здесь не на что особо смотреть. А кого это волнует? Главное, что вы научились писать HTML-код. Многие Web-дизайнеры начинали с написания именно таких страничек.

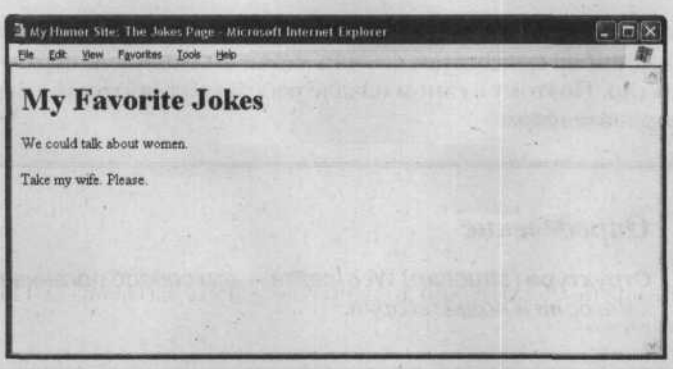


*Рис. 80.3. Завершенная страница «**My Favorite Jokes**» (Мои любимые шутки) выглядит так, как показано на рисунке*

Организация Web-сайта

Тогда как Web-страница представляет собой одиночный HTML-документ, **Web-сайт** — это набор связанных HTML-документов плюс все внешние файлы, такие как изображения и клипы, отображаемые на Web-страницах. Обычный вопрос, который возникает у начинающих Web-дизайнеров: как организовывать и хранить различные файлы, которые составляют сайт.

Ответ вполне очевиден: используйте обычные папки, создаваемые с помощью операционной системы компьютера.

Определение

***Web-сайт** (Web site) — это набор связанных HTML-документов плюс все изображения, клипы и другие файлы, которые отображаются на страницах сайта.*

Определение

***Локальная папка** (local folder), или **локальная корневая папка** (local root folder), — это папка на персональном компьютере, в которой содержатся все файлы Web-сайта.*

Большинство Web-дизайнеров начинают новый проект с создания папки, в которой будут храниться все файлы сайта. Такое место называется **локальной папкой** или **локальной корневой папкой**. Рассматривайте наличие этой папки как достоинство: удобно хранить все файлы сайта в одном месте. Вряд ли кому-нибудь хочется, чтобы файлы были разбросаны по всему жесткому диску. Но наличие локальной корневой папки важно также по техническим причинам. При загрузке сайта в Интернет вы хотите, чтобы местоположения файлов на вашем персональном компьютере как можно

точнее соответствовали их местоположениям на Web-сервере. Если такой вид параллельной организации не существует, ссылки между различными страницами могут не работать корректно. Поэтому в самом начале работы над сайтом убедитесь, что создана локальная корневая папка.

Определение

***Структура** (structure) Web-сайта — это способ организации файлов в категории и подкатегории.*

Внутри локальной корневой папки выполняется дополнительное упорядочивание, которое создает структуру сайта. Будьте аккуратны здесь — в этом смысле структура несколько отличается от структуры HTML-документа. **Структура** Web-сайта — это способ организации файлов в категории и подкатегории. Она не имеет ничего общего с HTML-тегами или разметкой текстовых файлов.

Для создания структуры сайта используются папки и подпапки. Чтобы правильно справиться с работой, необходимо вначале все хорошенько обдумать. Напрягите все свое воображение и постарайтесь представить категории информации, которые будут присутствовать на вашем сайте. Если вы создаете личный сайт, категории могли бы быть похожи на следующий пример:

- **My Hobbies** (Мои увлечения)
- **My Favorite Books** (Мои любимые книги)
- **My Favorite Jokes** (Мои любимые шутки)
- **About Me** (Обо мне)

Согласно существующему эмпирическому правилу следует создавать внутри локальной корневой папки вложенную папку для каждой основной категории вашего сайта. В предыдущем примере потребуется четыре папки внутри локальной корневой папки: по одной папке на каждую из четырех категорий. Не вкладывайте папки одна в другую. Вместо этого задайте свою структуру так, чтобы при входе в локальную корневую папку были видны все четыре папки — на одном **уровне** иерархии, если пользоваться техническим жаргоном.

Определение

***Уровень** (level) папки — это ее положение в иерархии структуры сайта. Папки основных категорий принадлежат верхнему уровню — т.е., находятся непосредственно внутри локальной корневой папки.*

Как следует называть папки? Подойдут любые имена, но назначайте короткие имена (по возможности восемь знаков или меньше). Используйте вместо пробела знак подчеркивания (_) и придерживайтесь обычных буквенно-числовых знаков. Это означает исключение из употребления специальных типографских символов или

пунктуационных знаков. Для структуры своего домашнего сайта вы бы могли присвоить папкам следующие имена:

- **hobbies** (Увлечения)
- **books** (Книги)
- **jokes** (Шутки)
- **about_me** (Обо мне). Можно также назвать **aboutme**, или просто **about**

Теперь посмотрим, что происходит. Начав создавать страницы для своего сайта, вы можете достаточно легко догадаться, чему они соответствуют. Вы помещаете страницы о своих увлечениях в папку **hobbies** (увлечения). Страницы о своих любимых шутках помещаются в папку **jokes** (шутки). Как видим, все достаточно просто.

Если требуется разбить основную категорию на подкатегории, нет проблем. Создайте подпапки:

- **hobbies** (Увлечения)
  - **sports** (Спорт)
  - **films** (Фильмы)
  - **computers** (Компьютеры)
  - **games** (Игры)
- **books** (Книги)
- **jokes** (Шутки)
- **about_me** (Обо мне)

Расширенная структура предоставляет более точную файловую систему. Теперь вы точно знаете где, например, должны находиться все страницы о фильмах.

Для сайта необходима еще одна папка – папка изображений.

- **hobbies** (Увлечения)
  - **sports** (Спорт)
  - **films** (Фильмы)
  - **computers** (Компьютеры)
  - **games** (Игры)
- **books** (Книги)
- **jokes** (Шутки)
- **about_me** (Обо мне)
- **images** (Изображения)

Как видите, эта папка находится на верхнем уровне структуры, непосредственно внутри локальной корневой папки. Используйте папку **images** (Изображения) для хранения всей графической информации, присутствующей на страницах вашего сайта.

Часто задаваемые вопросы

Итак, папка изображений предназначена для хранения только изображений?

Многие дизайнеры хранят в папке изображений помимо изображений другие файлы. Это нормально. Безусловно, папка изображений является отличным местом для хранения клипов, звуковых эффектов и подобных файлов. Просто не помещайте в нее HTML-страницы. Проверяйте, чтобы они размещались в папках соответствующих категорий.

Может возникнуть искушение хранить файлы изображений в соответствующих папках страниц, на которых находятся эти изображения. На первый взгляд такая стратегия не лишена смысла. Несомненно, можно организовать сайт подобным образом, но, скорее всего, так делать не стоит. Когда Web-сайт начнет расширяться, вы вряд ли захотите помнить, в какой папке находится то или иное конкретное изображение. Лучше просто хранить всю графику в одной папке. В этом случае вы всегда будете знать, где искать файл изображения.

Наконец, для сайта необходима домашняя страница, которая загружается при наборе посетителем в своем браузере вашего интернет-адреса. HTML-документ домашней страницы помещается внутрь локальной корневой папки, рядом с папками основных категорий. Не помещайте домашнюю страницу в свою собственную папку. Страница должна находиться на одном уровне с папками категорий. Кроме того, убедитесь, что домашней странице присвоено имя **index.htm**. Если присвоить ей другое имя, браузер может не понять, что эта страничка должна быть домашней.

Суммируя сказанное, типовую структуру сайта можно представить следующим образом.

- Локальная корневая папка
  - Документ домашней страницы (**index.htm**)
  - Папка основной категории
    - Документ основной категории
    - Папка подкатегории
      - Документ подкатегории
  - Папка изображений
    - Файлы изображений
    - Файлы клипов
    - Звуковые файлы
    - Другие файлы

Инструментарий

Шаблон стандартного HTML-документа

Данный инструментарий предоставляет стандартную структуру HTML-документа. Используйте ее в качестве шаблона для тестирования примеров в этой книге или для создания с нуля своих собственных Web-страниц.

```
<html>
  <head>
    <title>Здесь располагается название страницы.</title>

    <!-Здесь располагаются теги <meta> ->

   <!-Здесь располагается код JavaScript между тегами <script> ->

   <!-Здесь располагаются таблицы стилей между тегами <style> ->

  </head>
  <body>

    <!-Здесь располагается тело страницы ->

  </body>
</html>
```

ГЛАВА 81.
Добавление ключевых слов и описаний страниц

Когда поисковые системы, такие, как Google и Alta Vista, каталогизируют ваш сайт, они анализируют содержимое страниц и каталогизируют определенные слова и фразы, которые встречаются регулярно. Когда посетитель **google.com** или **altavista.com** выполняет поиск, скажем по слову **shoelaces** (обувные шнурки), существует большая вероятность, что ваш сайт появится среди результатов поиска, если на его страницах встречается слово **shoelaces** (обувные шнурки).

Что произойдет, если сайт полностью посвящен обувным шнуркам (shoelaces), но вы не используете часто слово shoelaces (обувные шнурки) в основном тексте? Это не такой уж маловероятный сценарий, как может показаться на первый взгляд. Как часто встречается слово **book** (книга) на **amazon.com**? Не так часто, как можно было бы ожидать.

Определение

Ключевые слова *(keywords) — это тематические заголовки страницы.* ***Описание страницы*** *(page description) — это краткая, в один абзац сводка содержимого страницы.*

Один из способов помочь поисковым системам более точно индексировать ваш сайт заключается в предоставлении ключевых слов и описаний всех страниц. **Ключевые слова** — это тематические заголовки страницы, тогда как **описание страницы** — это краткая в один абзац сводка содержимого страницы.

Ключевые слова и описания страниц не отображаются в браузере, таким образом, они не влияют на внешний вид страниц. Вы облегчаете задачу поиска вашего сайта потенциальным посетителям.

Для решения данной задачи применяйте мета-теги. **Мета-теги** — это специальные HTML-теги, которые содержат общие сведения о странице. Эти теги находятся в разделе заголовка HTML-документа вместе с названием страницы (page title) и любыми встроенными скриптами или таблицами стилей.

Определение

Мета-теги *(meta tags) — это специальные HTML-теги, в которые заключаются общие сведения о странице.*

Следующий блок кода показывает типовую разметку для ключевых слов и описания страницы:

```
<head>
  <title>Обувные шнурки и т.д. Home</title>
  <meta name="Keywords" content="обувные шнурки, кеды, кроссовки,
теннисные туфли, повседневная обувь, ботинки на шнурках, шнурки, узлы, покупки">
  <meta name="description" content="Домашняя страница первой в мире полноценной 24-часовой службы, предлагающей для покупки обувные шнурки и отличающейся широким выбором товара на складе и привлекательными ценами для всех знаменитых высококачественных обувных шнурков, снабжаемых ярлыками разработчиков и знаками качества.">
</head>
```

В обоих случаях мета-теги одинаковы. Другими словами не существует отдельного тега для ключевых слов и отдельного тега для описания. И ключевые слова и описания используют один и тот же общий тег `<meta>`. Что отличает эти типы тегов, так это атрибут `name`. Чтобы создать тег `<meta>` для ключевых слов, задайте значение атрибута `name` равным `keywords`. Чтобы создать тег `<meta>` для описания страницы, задайте значение атрибута `name` равным `description`. Атрибут `content` принимает в качестве значения либо список ключевых слов, либо описание страницы.

Совет

При составлении списка ключевых слов попробуйте представить, какие слова и фразы будут вводить потенциальные посетители вашего сайта в свои поисковые системы.

Определение

***Робот** (robot) — это специализированный фрагмент программы, который каталогизирует или «обнюхивает» ваш сайт для поискового механизма.*

Как в действительности работают поисковые системы, зависит от поискового механизма, а реальные процедуры поиска держатся компаниями в секрете. Но, как правило, процесс поиска выполняется приблизительно так. **Робот** поисковой системы, или программа Интернет-анализа, посещает сайт и просматривает его страницы. Выполняется сравнение ключевых слов и описаний с фактическим содержимым страниц. Если робот приходит к выводу, что ключевые слова и описания страниц являются точными, он учитывает их при каталогизации сайта. Однако если робот определяет, что ключевые слова и описания не соответствуют содержимому сайта, он может поставить сайт в невыгодную позицию в списке системы поиска. По этой причине следует предоставлять точные ключевые слова и описания. Если вами приводится список ключевых слов таких, как **обувные шнурки**, **шнурки** и **узлы**, убедитесь, что ваш сайт не занимается обслуживанием... ну, скажем, других типов веревок.

ГЛАВА 82.
Блокирование частей сайта для поисковых систем

На первый взгляд может показаться нелогичным желание ограничить доступ поисковых систем к какой-либо части сайта. Возможно, вы прилагаете большие усилия, чтобы ваш сайт попал в списки как можно большего числа поисковых систем. Однако попробуйте оценить следующие аргументы: может быть, вы хотите точно управлять тем, как посетитель находит ваш сайт. Скажем, вы бы предпочли, чтобы посетители заходили на начальную страницу, а не на страницы, находящиеся на три уровня глубже в структуре сайта. Или, может быть, вы не хотите, чтобы посетители заходили вначале на страницу, которая выполняет функцию всплывающего окна, где может отсутствовать полный набор средств навигации. Чем больше вы размышляете на эту тему, тем больше убеждаетесь, что ограничение доступа поисковых систем к некоторым областям сайта вполне оправдано.

Совет

Некоторые Web-дизайнеры, отдающие предпочтение дешевым решениям, а не хорошим, полагают, что методы, излагаемые в этой главе, предоставляют им надежный способ защиты важных сведений на их сайтах. Эти Web-дизайнеры поступят правильно, если полностью пропустят данную главу. Лучшее решение по безопасности — не держать конфиденциальную информацию на Web-сервере, и точка. Если вы не можете себе этого позволить, следует исследовать и реализовать действующие протоколы безопасности и авторизации, например, каталоги, защищаемые паролем.

Существует относительно простой и надежный способ передачи своих предпочтений по индексированию роботам — программам, которые поисковые системы посылают для каталогизации вашего сайта. Вы добавляете специальный текстовый файл с именем **robots.txt** на верхний уровень своего удаленного (remote) сайта прямо внутрь удаленной корневой папки. Файл **robots.txt** предписывает посещающим поисковым механизмам игнорировать специальные каталоги или файлы, которые в нем перечисляются.

Совет

*Удостоверьтесь, что вы используете текстовый редактор при создании файла **robot.txt** для своего сайта, и сохраните результат с расширением **.txt**. Вы сделаете неправильно, если создадите HTML-файл, а затем измените его расширение на **.txt**.*

Здесь есть загвоздка: чтобы описанная процедура работала, роботы должны удовлетворять требованиям стандарта исключения роботов (Robots Exclusion Standard), который является малоизвестным следствием из трех законов робототехники (Three Laws of Robotics) писателя-фантаста Азимова (Asimov). Стандарт исключения роботов

попросту утверждает, что робот должен подчиняться инструкциям, приводимым в файле **robot.txt**. *Однако этот стандарт не является законом. Он больше похож на правила хорошего поведения.* Разработчики роботов для поисковых систем не обязаны программировать свои творения в соответствии со стандартами, и действительно, существуют роботы-ренегаты, бешено носящиеся по Интернету в точности так, как описано в произведении «Я, робот» (I, Robot). Тем не менее, роботы всех основных поисковых систем работают в соответствии с рекомендациями.

Простой файл **robot.txt** выглядит подобно следующему примеру:

```
User-agent: *
Disallow: /popups/
Disallow: popup.htm
Disallow: /images/
Disallow: /js/
Disallow: /css/
```

Строки `Disallow` сообщают роботу, какие каталоги или файлы не следует посещать. В предыдущем примере каталоги **popups**, **images**, **js** и **css** заблокированы, также, как и файл **popup.htm**.

Строка `User-agent` показывает, в отношении каких роботов действуют строки `Disallow`.

Задание звездочки (*) в качестве значения `User-agent` означает, что инструкции `Disallow` действуют для всех роботов. Можно также указать отдельных роботов и задать для каждого из них разные уровни доступа:

```
User-agent: googlebot
Disallow: /popups/
Disallow: popup.htm

User-agent: Roverdog
Disallow: /popups/
Disallow: popup.htm
Disallow: /images/
Disallow: /js/
Disallow: /css/
```

В этом скрипте робот googlebot поисковой системы google не должен просматривать каталог **popups** и файл **popup.htm**, тогда как робот Roverdog кроме папки **popups** и файлу **popup.htm** *не должен заходить в каталоги* ***images***, ***js*** и ***css***.

Между прочим, значения в строках `Disallow` являются относительными путями, указываемыми относительно корня. Таким образом, если вы хотите скрыть подпапку, а не папку верхнего уровня, убедитесь, что указан полный путь к подпапке:

```
User-agent: Roverdog
```

```
Disallow: /swf/sourcefiles/
```

Если требуется скрыть абсолютно все (в этом случае от всех роботов), используйте следующую запись:

```
User-agent: *
Disallow: /
```

Совет

Знак «звездочка» в файле ***robots.txt*** *не является подстановочным знаком. Поэтому вы не можете записать, например,* `Disallow: *.gif`*, чтобы запретить поисковым системам сканировать все файлы изображений с расширением* ***GIF*** *— для этого следует поместить все* ***GIF****-файлы в папку и затем запретить доступ к этой папке. Звездочка работает только в строке* `User-agent` *и только как сокращение, обозначающее всех роботов.*

Совет

Дополнительные сведения о файле ***robots.txt****, а также имена различных роботов, не упомянутых здесь, см. на сайте* ***www.robotstxt.org/****.*

В следующем примере показано, как полностью запретить доступ для робота google и разрешить полный доступ для всех других роботов:

```
User-agent: googlebot
Disallow: /
```

Если вы хотите сделать доступным все для всех роботов, используйте такую запись:

```
User-agent: *
Disallow:
```

А если вы хотите разрешить доступ только одному роботу (в данном примере, роботу поисковой системы google), воспользуйтесь текстом:

```
User-agent: googlebot
Disallow:

User-agent: *
Disallow: /
```

Теперь вернемся к примеру в начале данной главы, где вы хотели попробовать заставить новых посетителей заходить на сайт через начальную страницу. Допустим, ваш

сайт содержит пять каталогов верхнего уровня: **products** (продукты), **services** (услуги), **aboutus** (о нас), **images** (изображения) и **apps** (приложения) наряду с HTML-файлом **contact.htm**. Ваш файл **robots.txt** выглядит следующим образом:

```
User-agent: *
Disallow: /products/
Disallow: /services/
Disallow: /aboutus/
Disallow: /images/
Disallow: /apps/
Disallow: contact.htm
```

Поместите этот файл в каталог верхнего уровня удаленного (remote) сайта, и поисковые системы будут индексировать только вашу домашнюю страницу (**index.htm**).

ГЛАВА 83.
Тестирование сайта

Для тестирования Web-сайта не обязательно быть подключенным к Интернету. В действительности для этого вообще не нужна Web-служба. Web-браузеры отлично справляются с чтением HTML-файлов, хранящихся на персональном компьютере.

Определение

*На Web-сайте **локальные файлы** (local files) — это файлы, которые хранятся на персональном компьютере в целях их разработки и тестирования. **Удаленные файлы** (remote files) — это файлы, которые публикуются на Web-сервере. Именно удаленные файлы видят посетители, когда заходят на ваш сайт.*

Тестирование сайта с помощью персональных или **локальных** файлов вполне оправданно. Одна из причин – нет необходимости подключаться к сети для тестирования сайта и можно вносить любые изменения в код Web-страниц, не оказывая воздействия на опубликованную версию сайта. Выполненные изменения не действуют до тех пор, пока вы не загрузите модифицированные страницы на Web-сервер. По этой причине локальное тестирование помогает улучшить контроль качества. Нет необходимости размещать плохо кодированные страницы или страницы с ошибками, чтобы весь мир увидел их. Вместо этого вы устраняете недочеты в автономном режиме и загружаете свои Web-страницы после того, как все будет функционировать надлежащим образом.

Для тестирования сайта в автономном режиме запустите Web-браузер. Откройте меню **File** (Файл) и выберите команду **Open** (Открыть). В Netscape используется команда **Open File** (Открыть файл). Появится диалог с запросом на выбор открываемого файла. Перейдите в локальную корневую папку и выделите домашнюю страницу сайта. Страница загрузится, но помните, что это локальная версия, а не **удаленная** (remote) версия и не версия в Интернете.

Часто задаваемые вопросы

Какие типы ссылок работают при тестировании сайта в автономном режиме?

При тестировании локальной версии сайта без проблем работают относительные ссылки, указываемые относительно документа. Тем не менее, абсолютные ссылки нуждаются в подключении к Интернету, а относительные ссылки, указываемые относительно корня, функционируют только на Web-сервере. См. главу 61, в которой обсуждаются различные типы путей, применяемых в ссылках.

В целях тестирования рекомендуется загрузить последние версии нескольких разных браузеров. Даже если вы пользуетесь исключительно Internet Explorer или являетесь в повседневной жизни яростным поклонником Netscape, будучи Web-дизайнером, вы должны думать также, как ваши посетители. Никогда не известно, каким браузером будет пользоваться тот, кто посетит ваш сайт. Поскольку разные браузеры отображают одну и ту же Web-страницу по-разному – и различия могут быть значительными — вы хотите предусмотреть любые потенциальные проблемы, прежде чем загрузить страницу в Интернет.

Часто задаваемые вопросы

Как насчет Mac-браузеров? Как насчет Linux-браузеров? Как насчет других браузеров для Windows?

Как бы это жестоко не звучало, в Интернете попросту нет достаточного количества компьютеров, которые бы не относились к Windows, Microsoft или Netscape, чтобы говорить о какой-либо статистической разнице. Если вы создаете сайт общего назначения, не считайте себя обязанным тестировать сайт, например, в браузере Macintosh. Если у вас есть доступ к браузеру платформы Mac, конечно же, протестируйте сайт в нем. Но не думайте, что вам надо покупать Mac, только чтобы убедиться в работоспособности сайта с Mac-браузерами.

Netscape и Opera — значительные альтернативы для браузера корпорации Майкрософт, но они не являются единственными. Альтернативные браузеры — в особенности любимая Mozilla, форма проекта Netscape, которую он принял по окончании войны браузеров — ориентируются на стили жизни и философии слишком маленьких групп людей, являющихся прежними поклонниками, оригиналами-интернетчиками или лицами, продвигающими программный продукт.

Если бы Интернет не развился в заметное информационное средство, возможно, вам бы требовалось разрабатывать сайты и для этих альтернативных браузеров. Однако так сложились обстоятельства (или, скорее всего, рынок), что Интернет стал популярной средой, и огромное большинство его пользователей знают о технической стороне его работы не намного больше того, что телезрители знают о работе телевизора. В наши дни нужно быть серьезным и преданным знатоком Интернета, чтобы искать пользователей других программных продуктов помимо программного обеспечения корпорации Майкрософт. Несомненно, проект Mozilla возник благодаря Time Warner, корпоративным собственникам марки Netscape, поэтому если вы страстно желаете найти действительно альтернативный Web-браузер, поищите где-нибудь еще.

Тем не менее, факт остается фактом: опять же, к сожалению, реальности Интернета на сегодняшний день таковы, что для сайтов общего назначения требуется брать в рассмотрение главным образом Internet Explorer. Однако, на всякий случай, посетите сайт **www.mozilla.org/**.

Самым распространенным браузером является Microsoft Internet Explorer для Windows. Чтобы вы ни делали, убедитесь, что ваш сайт хорошо работает в этом браузере. Вторым по популярности браузером, намного отстающим от Internet Explorer, является Netscape для Windows. Хотя можно разработать эффективный сайт, ориентируясь исключительно на его вид и работу в браузере корпорации Майкрософт, большинство дизайнеров считают свои сайты не завершенными до тех пор, пока страницы не будут нормально выглядеть как в Internet Explorer, так и в Netscape. В Европе достаточно широко распространен браузер Opera для Windows, поэтому желательно, чтобы сайты с большой международной аудиторией работали и с браузером Opera.

Посетите сайт **www.netscape.com/**, чтобы загрузить бесплатную копию браузера Netscape, а для загрузки копии Opera зайдите на сайт **www.opera.com/**.

ГЛАВА 84.
Выбор Web-хоста

Вам не нужен Web-хост для разработки и оформления сайта. Все это можно делать на персональном компьютере. Однако если вы хотите опубликовать свой сайт так, чтобы он был виден для всего мира, потребуется некоторое место на Web-сервере. К счастью, на серверах предоставляется достаточно места по сравнительно небольшой цене. В этой главе приводятся сведения, помогающие разобраться с доступными вариантами публикации сайта.

Определение

Web-хост *(Web host) имеет в собственности (или арендует) компьютер, который предоставляет файлы Web-сайта для ваших посетителей.*

Поиск Web-хоста

Web-хост имеет в собственности (или арендует) компьютер, который предоставляет файлы Web-сайта для ваших посетителей, обычно в обмен на ежемесячную плату. Вам не надо отменять свою текущую учетную запись Интернета, чтобы воспользоваться услугами Web-хоста. В действительности, многие Web-хосты не предлагают услуг, которые мы привыкли получать от интернет-провайдеров (ISP). Последние должны находиться рядом с вами, иначе они бесполезны для вас. Самое важное свойство ближайшего поставщика услуг Интернета заключается в том, что он находится рядом с вами и предоставляет вам местный номер коммутируемого доступа. Вы вряд ли захотите дозваниваться до Владивостока всякий раз, когда пожелаете побродить в Интернете (конечно, если вы не живете во Владивостоке).

Скорее всего, у вас уже есть Web-хост, даже если вы не знаете о нем. Многие стандартные учетные записи Интернета предоставляются с достаточно большим объемом серверного пространства для личных Web-страничек. Поэтому первый шаг в выборе Web-хоста состоит в обращении к своему поставщику услуг Интернета. Узнайте, сколько серверного пространства предоставляется вместе с учетной записью. Если вам ответят, что место на сервере не предоставляется, возможно, вы пожелаете проинформировать представителя службы клиентов, что их прямые конкуренты предлагают серверное место своим абонентам за много меньшую плату, чем вы платите сейчас.

Если вы предпочитаете не использовать выделяемое вам место на сервере для своего сайта, зайдите на свою любимую поисковую систему и предпримите поиск по ключевому слову **Web hosting** (Web-хостинг), чтобы начать исследование этого вопроса. Не удивляйтесь, если число полученных результатов перевалит за миллион. Существует много, очень много Web-хостов. Невозможно даже просто просмотреть все предложения. Необходимо сузить число вариантов – и быстрее. Лучше всего начать с имен,

которые у вас на слуху: **Yahoo**, **Earthlink**, **Netfirms**, может быть также ваша телефонная или кабельная компания. Если у них нет того, что вам надо, двигайтесь дальше по списку возможных кандидатов.

Часто задаваемые вопросы

Я хочу использовать свой Web-сайт в коммерческих целях. Могу я разместить свой сайт в персональном Web-пространстве, которое предоставляется моим провайдером?

Зачем вам это нужно? Один из контраргументов — поставщик услуг Интернета обычно присваивает вам общий интернет-адрес, который не позволяет выглядеть вашему бизнесу профессионально. Другой контрдовод — для вас нежелательны обычные задержки ответа, которые присущи серверам крупных компаний, поставщиков услуг Интернета. Кроме того, возможно, коммерческие сайты не входят в перечень услуг, предоставляемых вашим провайдером, а вам не нужны лишние хлопоты, связанные с закрытием вашего сайта вместе с учетной записью электронной почты и подключения к Интернету.

Совет

Прежде, чем подписать договор с Web-хостом, проверьте его. Поищите его имя с помощью своей любимой поисковой системы и посмотрите, что получите. Посмотрите также отзывы. У многих хороших Web-хостов существует немало положительных отзывов в прессе. Если результат поиска по выбранному хосту подозрительно пуст, многократно подумайте, прежде чем подписывать с ним договор.

Совет

Если предложение слишком хорошо, чтобы быть реальным, возможно, вас просто хотят обмануть. Покупатель, будь осторожен!

В общем и целом, в случае Web-хостинга объем получаемых услуг прямо пропорционален оплате. Стоящие сделки немногочисленны и находятся где-то в середине ценового диапазона. Помните об этом, когда вам встретятся хосты, предлагающие «бесплатные» услуги. Бесплатные хосты обычно не предоставляют многообразие вариантов учетных записей (или достаточно качественных), как компании, взимающие помесячную оплату, и при этом придется забыть об услугах клиентской службы. Кроме того, вы должны помнить курс экономики, пройденный в институте. Эти люди остаются в бизнесе благодаря получаемому доходу. Они должны где-то брать деньги. И этим «где-то» являетесь вы. Бесплатные Web-хосты обычно загружают ваш сайт с объявлениями, которые вы не можете контролировать, выключать, изменять или перемещать. А электронные сообщения рассерженных посетителей, жалующихся на бесстыдную, вездесущую рекламу, поступают прямо в ваш почтовый ящик. Ваш бесплатный

Web-хост доверяет заниматься обработкой этой почты лично вам. Эти письма все ваши. Конечно, все Web-сайты отличаются друг от друга, и только от вас зависит выбор правильного Web-хоста для своего конкретного сайта. На основании вышесказанного здесь можно привести несколько качеств, которые в общем случае отличают хорошие Web-хосты:

- **Умеренная месячная плата.** Исследуйте рынок и посмотрите, что предлагают конкуренты. За типовой личный или небольшой коммерческий сайт месячная плата может составить от 10 до 25 долларов США. Заплатите меньше и ваш Web-хост возможно компенсирует полученную вами разницу за счет предоставления ужасной клиентской поддержки или путем размещения объявлений на вашем сайте. Заплатите больше, и вы, вероятно, получите услуги и функции, которые вам не будут нужны.

- **Регистрация имени домена (желательно бесплатно).** Многие хорошие Web-хосты выполняют вместо вас регистрацию имени домена для Web-сайта. Некоторые даже требуют вперед ежегодную плату в 35 долларов США за услуги по регистрации. Обратите внимание на такие хосты. Они стоят оплаты, даже если добавляют плату за установку (до тех пор, пока оплата не превышает 35 долларов США).

- **Умеренная (или ее отсутствие) плата за настройку.** Некоторые Web-хосты добавляют специальную одноразовую плату за настройку учетной записи. Другие не берут за это платы. Многие уважаемые и надежные Web-хосты выставляют счет за настройку. Многие этого не делают. Те, кто не берет плату за настройку, часто делают свои деньги на чем-нибудь еще, поэтому читайте положения договора, набранные мелким шрифтом. Если вы обратились в службу, где берут плату за настройку, не платите больше 35 долларов, или из вас, возможно, вытягивают деньги. А если вы смогли напасть на Web-хост, снижающий плату в течение одной из своих рекламных акций, воспользуйтесь этим обстоятельством.

- **Планы по обновлению.** Ваш сайт пользуется успехом. Он расширяется. Вы хотите, чтобы ваш Web-хост расширялся вместе с вами. В идеале ваш хост должен предоставлять несколько уровней услуг по приемлемым ценам. Начните с самого дешевого плана и двигайтесь в направлении услуг верхних уровней.

- **Надежная служба поддержки клиентов (желательно по телефону).** Вам потребуются услуги службы поддержки клиентов. Точка. И не попадайтесь на удочку услуг клиентской службы в формате 24/7 (24 часа 7 дней в неделю). Любой дурак с электронным адресом может рекламировать службу поддержки клиентов в формате 24/7. Но читает ли в действительности этот дурак электронную почту? – ага, попались. Вам необходим бесплатный телефонный номер для поддержки клиентов. Если у вас нет круглосуточного доступа 7 дней в неделю к живому человеку на телефоне, значит, вам не предоставляется поддержка в формате 24/7. Гораздо лучше выбрать Web-хост, который предлагает поддержку по бесплатному телефонному номеру в течение обычного рабочего дня, чем Web-хост, который предлагает поддержку в формате 24/7 по электронной почте.

Предоставляемые возможности

Web-хосты предлагают разнообразные возможности. Трудно узнать, что вам потребуется, а что нет. Прежде, чем заключать договор с Web-хостом, следует рассмотреть некоторые важные факторы, которые описываются ниже:

- **Место на сервере.** Какой точный объем пространства на сервере вам необходим? Это достаточно просто определить. Узнайте общий размер файлов Web-сайта, посмотрев свойства своей локальной корневой папки. Если сайт занимает 5 МБ, вам потребуется, по крайней мерей, 5 МБ пространства на сервере. Обычно не требуется огромное пространство на сервере, если только ваш сайт не содержит больших мультимедийных файлов, например, формата MP3.
- **Ограничения по пропускной способности.** Ваш Web-хост измеряет пропускную способность, или количество данных, которые его компьютеры передают посетителям вашего сайта в течение некоторого периода времени. Обычно, для вас устанавливается ежемесячный предельный объем либо если вы исчерпываете лимит, вам выставляется дополнительная плата. В большинстве случаев для личного или небольшого коммерческого сайта достаточный объем передаваемых ежемесячно данных составляет 1 ГБ. Грубо говоря, один гигабайт переданных данных равен 20 000 просмотров страниц — т.е. это эквивалентно просмотру одним пользователем одной страницы вашего сайта 20 000 раз или просмотру 5000 людей одной страницы вашего сайта по четыре раза, или однократному просмотру страницы сайта 20 000 людей. Трудно узнать, какая точно полоса пропускания потребуется вам, поэтому начните с малых значений. Если вы регулярно превышаете отведенный лимит, рассмотрите возможность изменения своего плана. Учтите: если вы предлагаете на своем сайте файлы MP3 общим объемом 1 ГБ, то вы очень быстро исчерпаете отведенный вам объем передаваемых данных.

Определение

Пропускная способность *(bandwidth) — это количество данных, которые передаются Web-хостом с вашего сайта ежемесячно.*

- **Учетные записи электронной почты по протоколу POP3.** Вам могут потребоваться, а может, и нет, дополнительные учетные записи электронной почты для своего сайта. Но если они вам нужны, обратитесь к Web-хосту, который предоставляет их. Обычно требуется электронная почта по протоколу POP3 вместо электронной почты с Web-интерфейсом. Электронная почта POP3 работает с клиентскими программами, например, с Microsoft Outlook и Eudora, которые позволяют быстро обращаться в Интернет, загружать электронную почту и готовить ответы в автономном режиме. Для работы с электронной почтой через Web-интерфейс требуется активное подключение к Интернету, чтобы набирать и получать сообщения. Многие Web-хосты зачисляют объем электронной почты, посылаемой и получаемой вами через их серверы, в общий лимит передаваемых данных, поэтому следует учитывать этот факт.

- **Потоковое аудио/видео и другие виды мультимедиа.** Если вы хотите держать на сайте файлы потокового аудио или видео, либо если вы хотите показывать Flash-анимацию и другие стандартные типы мультимедиа, Web-хост должен выполнить для этого соответствующую настройку своих служб. Для хостинга Web-изображений форматов **JPEG**, **GIF** и **PNG** не требуется какая-либо специальная служба.
- **Расширения FrontPage.** Если ваш Web-сайт, не дай Бог, использует бестолковые специальные эффекты, которые поставляются с Microsoft FrontPage, вам следует найти Web-хост, который не предлагает расширения **FrontPage**. В таком случае бессмысленные штуки не будут работать, и качество вашего сайта резко улучшится.
- **CGI (Интерфейс компьютерной графики), серверные скрипты и доступ к базам данных.** Если на сайте используется серверная технология для подключения к базе данных, вам потребуется Web-хост, оборудование которого настроено для предоставления таких услуг. Вам не надо беспокоиться о такого рода услуге, если на сайте используются клиентские приложения: HTML, JavaScript и CCS (каскадные таблицы стилей). Все проекты в этой книге относятся к клиентской технологии.
- **Отчеты сайта.** Вам нужен такой хост, который предоставляет доступ к данным, собираемым Web-сервером о посетителях вашего сайта. Вы должны знать, откуда они приходят на сайт, какими браузерами они пользуются, какие страницы посещаются чаще всего и каких разделов на сайте по их мнению не хватает. Вам не нужна эта информация для злонамеренных целей. Она требуется вам для улучшения сайта. Несомненно, следует проверить, какие виды средств отчетности о сайте предлагаются хостом.

Совет

Одна функция, которая определенно не потребуется вам — это приложение «электронная корзина», будь то «бесплатная» или иная. «Электронные корзины», предоставляемые с планами Web-хостинга, страдают обобщенностью, высокой стоимостью, функциональной бедностью, трудны в настройке, и их практически невозможно встроить в сайт. Если вы серьезно задумались о запуске розничного Web-магазина, вам следует нанять команду программистов для разработки приложения «электронной корзины», предназначенного для вашего сайта.

ГЛАВА 85.
Загрузка сайта на сервер

У вас есть Web-сайт. Для вас выделено место на Web-сервере. Теперь все, что необходимо сделать, это загрузить сайт на сервер.

Чтобы сделать это, достаньте бесплатную FTP-программу. **FTP** расшифровывается как **File Transfer Protocol** (Протокол передачи данных) – это метод, используемый компьютерами для обмена файлами данных в Интернете. Программа FTP позволяет подключаться к Web-хосту (удаленному компьютеру) с персональной машины (локального компьютера). После установки подключения достаточно просто указать, какие файлы требуется переместить на сервер, и FTP-программа загрузит их туда. Можно также загрузить файлы с сервера на персональный компьютер. Две наиболее популярные и абсолютно бесплатные FTP-программы – это WS_FTP и CuteFTP, обе можно найти в Интернете. Используя свою любимую поисковую систему, найдите эти программы. Затем загрузите и установите ее, после чего все будет готово к дальнейшей работе.

Определение

***FTP** расшифровывается как **File Transfer Protocol** (Протокол передачи данных) – метод, используемый компьютерами для обмена файлами данных в Интернете.*

Чтобы подключиться к Web-серверу через программу FTP, необходимо знать несколько технических деталей:

1. **Адрес хоста.** Это URL-адрес сервера, на который будет выполняться загрузка. Во многих случаях, но не всегда, адрес хоста совпадает с именем вашего домена. Чтобы точно узнать адрес, обратитесь к Web-хосту.
2. **Ваш идентификатор пользователя (user ID).** Это имя пользователя, которое Web-хост присваивает вам.
3. **Ваш пароль.** Это пароль учетной записи сайта.

В зависимости от вашего Web-хоста может потребоваться предоставить также другие сведения. Если возникают какие-либо сомнения, обратитесь в службу поддержки клиентов.

После первоначального подключения необходимо создать удаленную (remote) **структуру** сайта – иерархию папок, в которых хранятся файлы. Удаленная структура должна как можно точнее соответствовать локальной структуре. Так, например, если в локальной корневой папке сайта находится пять папок, необходимо создать пять идентичных папок на удаленном сайте. В вашей программе FTP есть кнопки или команды для создания папок или **каталогов** (directories), если пользоваться технической терминологией.

После того как создана удаленная структура, начните загрузку файлов. Скопируйте их из локальных папок в соответствующие удаленные папки с помощью программы FTP. Обратите внимание, что при загрузке файлов они не удаляются с персонального компьютера. Вместо этого создаются их идентичные копии на Web-сервере.

Определение

***Структура** (structure) Web-сайта — это иерархия его папок. **Каталог** (directory) — это еще одно название компьютерной папки.*

Совет

*При первоначальной загрузке файлов сайта начинайте загрузку с самого нижнего уровня. Другими словами, первыми следует загружать файлы под-под-подкатегорий и в последнюю очередь должна загружаться домашняя страница, **index.htm**.*

Когда файлы на удаленном сервере соответствуют файлам на локальном компьютере, ваш сайт **синхронизирован**. Но при редактировании локального сайта, добавлении нового содержимого или изменении существующих страниц, сайт становится **рассинхронизированным**. Почему? Потому что больше уже нет соответствия между локальными и удаленными файлами. При сохранении изменений домашней страницы на персональном компьютере, он не обновляет автоматически домашнюю страницу на Web-сервере. Помните, удаленный файл является отдельным документом. Чтобы снова синхронизировать ваш сайт, зайдите на Web-сервер через программу FTP и замените устаревшие удаленные файлы их обновленными локальными аналогами.

Определение

*Web-сайт **синхронизирован**, когда его локальные файлы соответствуют удаленным (remote) файлам. Web-сайт **рассинхронизирован**, когда его локальные файлы не соответствуют удаленным файлам.*

Совет

Всегда следует делать локальную резервную копию файлов своего Web-сайта. Но если вдруг случится невероятное и вы потеряете все свои локальные файлы, скопируйте самые последние версии опубликованных файлов с удаленного сервера. Вы потеряете все изменения, которые внесли с момента последней загрузки файлов на Web-сервер, но, по крайней мере, сможете восстановить версию своего сайта.

Большей частью файлы на удаленном сервере предназначены только для чтения. Обычно вы не можете открывать и редактировать их на удаленном компьютере. В качестве альтернативы загрузите файлы, которые хотите изменить (удостоверьтесь, что загружаете их в папку, отличную от локальной корневой папки, или вы можете перезаписать локальные файлы с таким же именем). Внесите необходимые изменения и загрузите измененные файлы обратно на удаленный компьютер.

ГЛАВА 86.
Проверка правильности исходного кода

Специалисты консорциума W3C (World Wide Web) настолько привержены кодированию, полностью соответствующему спецификациям, что предлагают услуги по бесплатной проверке соответствия Web-страниц правилам кодирования. Если вы хотите проверить, насколько хорошо ваш сайт соответствует текущим спецификациям и рекомендациям, эта услуга может представить для вас определенный интерес. Программа проверки, используемая консорциумом W3C, не только указывает все последние ошибки кодирования или ложные шаги в коде Web-страницы, она выводит длинный отчет с объяснениями и предлагаемыми способами исправления обнаруженных проблем.

Совет

Если вам становится не по себе при получении плохих отметок, просто пошлите программу проверки кода W3C на домашнюю страницу популярного сайта, такого как Amazon или eBay.

Чтобы проверить правильность кода Web-страницы, зайдите на сайт **http://validator.w3.org/**. Введите URL-адрес Web-страницы, код которой требуется проверить или вложите автономный документ и пошлите его им. Желаю удачи! Если ваша страница пройдет проверку успешно, консорциум W3C поощрит вас, разрешив отображение их официальной эмблемы путем добавления следующего кода, обычно внизу Web-страницы:

```
<p>
  <a href="http://validator.w3org/check/referer">
  <img src="http://www.w3.org/Icons/valid-xhtml10" alt="Valid XHTML
1.0!" height="31" width="88" />
  </a>
</p>
```

Чтобы повысить шансы прохождения страницей проверки, попробуйте придерживаться следующих советов:

- Используйте строчные буквы для названий тегов и атрибутов.
- Заключайте в кавычки значения атрибутов.
- Некоторые атрибуты, такие как `checked` и `selected`, не имеют значений. Присваивайте им значения, совпадающие с названием атрибута, как в следующем примере:

```
<input type="checkbox" checked="checked" />
```

- Обращайте внимание на правильность вложения. Закрывайте вложенные теги в порядке, обратном порядку их открытия. Первый открытый тег должен закрываться последним.
- Если у тега нет его закрывающей версии, как у тегов <img>,
 и <hr>, включайте наклонную черту перед закрывающей угловой скобкой, как в следующем примере:

```
<img src="image.jpg" width="200" height="100" alt="An image" />
```

- Выполняйте вложение списков внутри элементов li так, как показано в следующем примере:

```
<ol>
  <li>Элемент 1 списка</li>
  <li>Элемент 2 списка</li>
  <li>
    <ol>
      <li>Элемент 1 вложенного списка</li>
      <li>Элемент 2 вложенного списка</li>
    </ol>
  </li>
</ol>
```

- Не вставляйте скрипты или таблицы стилей. Вместо этого используйте ссылки на внешние файлы JavaScript или CSS.
- Если необходимо вставить скрипты или таблицы стилей, удаляйте все комментарии, которые скрывают код от несовместимых браузеров.

Глоссарий

JavaScript. Клиентская часть технологии добавления логических функций и математических расчетов на Web-страницу.

Web-страница (Web page). Одиночный HTML-документ вместе с изображениями, мультимедийными файлами, сценариями и таблицами стилей.

Абсолютный путь (absolute path). Путь к файлу, предоставляющий полный Web-адрес файла. Абсолютный путь замечателен тем, что он всегда остается одним и тем же, независимо от места появления на сайте. Ссылку с указанием абсолютного пути можно перемещать на любой уровень сайта, не изменяя формата пути. Недостаток абсолютного пути состоит в том, что он не работает в автономном режиме. Для перехода по ссылке, содержащей абсолютный путь, необходимо находиться в режиме онлайн. По этой причине большинство дизайнеров предпочитают применять абсолютные пути только для внешних ссылок, а для внутренних ссылок используют пути, указываемые относительно документа или корневого каталога.

Автоматически изменяющийся макет (expando layout). Макет, размер которого автоматически изменяется под ширину окна браузера. Это то же самое, что и **текучий макет** (liquid layout).

Активная ссылка (active link). Ссылка, которая в текущий момент выделена или на которой щелкает мышью посетитель.

Атрибут (attribute). Параметр тега или свойства каскадных таблиц стилей (CSS), который дополнительно определяет тег или свойство.

Без выключки (ragged). Стиль абзаца, в котором строки текста имеют естественную длину, без дополнительных пустых промежутков между буквами и словами. Сравните с термином **с выключкой** (justified).

Без выключки влево (ragged left). Стиль абзаца, в котором строки текста выравниваются по правому краю, а левый край остается невыровненным.

Без выключки вправо (ragged right). Стиль абзаца, в котором строки текста выравниваются по левому краю, а правый край остается невыровненным.

Без засечек (sans serif). Шрифт, такой, как Arial, который не имеет на концах букв мелких украшений в виде засечек. Сравните с термином **с засечками** (serif).

Блок (block). Прямоугольная область, в которой находится HTML-элемент. Обычно блок прозрачен, хотя его можно закрасить, добавив атрибут цвета в тег блока или в определение стиля.

Брендинг (branding). Метод дизайна, заимствующий элементы из маркетингового материала организации: эмблемы, фирменные цвета, девизы, ассоциативные картинки и так далее.

Буквица (drop cap). Прописная буква завышенного размера, которая обычно отображается в начале главы. Она выступает под базовую линию текста, и окружающие текстовые строки обтекают ее.

В реальном времени (real-time). Обозначение чего-нибудь – обычно информации – что изменяется в точном соответствии со временем.

Визуальный вес (visual weight). Определяет, насколько тяжелым кажется дизайнерский элемент. В общем случае, более крупные элементы выглядят, как правило, тяжелее, чем меньшие по размерам элементы.

Виртуальные переносы (virtual wrapping). Тип переноса слов в текстовой области, где форма передает текст без дополнительных обрывов строк, которые создаются при переносе. Сравните с термином **физический перенос** (physical wrapping).

Вложенная таблица (nested table). Таблица, которая находится внутри ячейки другой таблицы.

Вложенный раздел, вложенный слой (nested div, nested layer). Элемент раздела (div element), который отображается внутри другого элемента раздела (div element).

Вне экрана (below the fold). Области Web-страницы, которые оказываются за границами экрана. Посетитель должен прокрутить окно, чтобы увидеть содержимое вне экрана. Никогда не помещайте основные элементы управления в области вне экрана!

Внедрение (embedding). Включение кода, например, каскадных таблиц стилей или JavaScript, внутрь раздела заголовка HTML-документа между тегами стиля или сценария. Сравните с термином **присоединение** (attaching).

Внешняя ссылка (external link). Ссылка на файл, находящийся вне текущего сайта. Для внешних ссылок следует использовать абсолютные пути. Сравните с термином **внутренняя ссылка** (internal link).

Внутренняя ссылка (internal link). Ссылка на файл, находящийся на текущем Web-сайте. Для внутренних ссылок можно использовать абсолютные пути и пути относительно документа или корневого каталога. Сравните с термином **внешняя ссылка** (external link).

Внутристрочная ссылка (inline link). Ссылка, отображаемая внутри текста на странице.

Внутристрочное изображение (inline image). Изображение, отображаемое внутри блока HTML-текста.

Всплывающее меню (popup menu). Список пунктов меню, который появляется при наведении указателя на элемент в главном меню навигации. Обычно всплывающие меню создаются с помощью элементов разделов (div) языка DHTML.

Выделенное состояние (hover state). Состояние ссылки при наведении на нее указателя мыши. Данное состояние ссылки нельзя получить с помощью обычного HTML-кода. Вместо этого пользуйтесь каскадными таблицами стилей для определения данного выделенного состояния.

Выравнивание по центру (ragged center). Стиль абзаца, в котором браузер центрирует строки текста по горизонтали, оставляя оба края невыровненными.

Гиперкарта ссылок (image map). Изображение, содержащее один или несколько гиперобъектов либо гипероблостей.

Гиперобъект (hotspot). Область на изображении, щелчок на которой вызывает выполнение каких-нибудь действий.

Диапазон (span). Сегмент содержимого, идентифицируемый в HTML-документе тегами диапазона.

Динамический HTML (DHTML – Dynamic HTML). Динамический гипертекстовый язык разметки страниц или комбинация HTML, JavaScript и каскадных таблиц стилей (CSS). Применяется для разработки сложных, интерактивных Web-сайтов.

Дистанционная папка, дистанционная корневая папка (remote folder, remote root folder). Папка (или каталог) на Web-сервере, в которой находятся все файлы сайта.

Дистанционный (remote). Принадлежащий компьютеру, к которому локальный компьютер пользователя подключается через сеть.

Дистанционный файл (remote file). Электронный документ на Web-сервере.

Зазор между ячейками (cellspacing). Величина внешнего зазора между отдельными ячейками таблицы. В макетных таблицах этот атрибут обычно задается равным 0.

Знак конца (end sign). Буллит (черный кружок) или типографский символ, который отображается в конце рассказа или статьи. Также является пророчеством о том, что грядет Армагеддон.

Именованный анкер (named anchor). Местоположение на Web-странице, к которому может привязываться ссылка.

Источник (source). В ссылке это элемент, на котором посетитель щелкает мышью для перехода на другую страницу.

Каскадные таблицы стилей (CSS – Cascading Style Sheets). Технология, используемая на клиентских компьютерах для описания стиля и вида HTML-элементов.

Каскадные таблицы стилей значительно расширяют возможности Web-дизайна, особенно в отношении оформления, но браузеры обеспечивают поддержку лишь части возможностей. Даже самые современные браузеры разбивают каскадные таблицы стилей самыми неожиданными и непредсказуемыми способами, поэтому сайт, использующий каскадные таблицы стилей, необходимо тестировать в разных браузерах.

Класс (class). Селектор специального стиля, который необязательно должен применяться к какому-либо конкретному HTML-тегу или последовательности тегов.

Ключевые слова (keywords). Термины, описывающие содержимое Web-страницы и используемые механизмами поиска. Ключевые слова для страницы помещаются в мета-теги.

Контекстный селектор (contextual selector). Селектор стиля, определяющий все HTML-теги определенного типа, которые располагаются где-либо внутри тега другого типа, например, теги выделения (strong tags), находящиеся внутри упорядоченного списка.

Корень (root). Самый верхний уровень в структуре сайта.

Линейка (rule). Внутренняя граница в таблице. Линейки сохраняют свою стандартную толщину, независимо от размера атрибута границы таблицы.

Локальная папка, локальная корневая папка (local folder, local root folder). Папка на персональном компьютере пользователя, в которой тот хранит все файлы своего сайта.

Локальный (local). Относящийся к персональному компьютеру пользователя.

Локальный файл (local file). Электронный документ, находящийся на персональном компьютере пользователя.

Макет с фиксированной шириной (fixed-width layout). Макет страницы, в котором для определения размера страницы по горизонтали используются абсолютные измерения в пикселах. Страницы, для которых используется макет с фиксированной шириной, всегда отображаются с одинаковой шириной, независимо от размера экрана или окна браузера.

Макетная таблица (layout table). HTML-таблица, которая формирует структуру макета страницы. Спецификации HTML не предусматривают использование макетных таблиц. Сравните с термином **таблица данных** (data table).

Меню переходов (jump menu). Указатель быстрых переходов на наиболее популярные страницы Web-сайта, которые отображаются в открывающемся меню.

Мета-тег (meta tag). Специальный HTML-тег, содержащий общую информацию о Web-странице, например ключевые слова, описание страницы и частоту обновления. Браузер не отображает в окне документа содержимое мета-тегов.

Метка (label) кнопки. Текст на кнопке.

Метка (label) формы. Текст, описывающий функцию элемента управления.

Метка времени (timestamp). Небольшой фрагмент текста, сообщающий о текущем времени.

Мозаичное заполнение (tiling). Повторение изображения для заполнения заданной области.

Моноширинный (monospace). Шрифт, например Courier New, в котором все знаки отображаются с одинаковыми интервалами между ними для создания эффекта печати на пишущей машинке.

Мультимедиа (media). Компоненты Web-сайта, которые не попадают в категории изображений, HTML, CSS (каскадные таблицы стилей), сценарии клиентской или серверной части. Распространенными типами мультимедиа являются звуковые файлы, видеофайлы, Flash-анимации, файлы векторной графики (SVG) и документы программы Acrobat.

Набор полей (fieldset). Группа логически связанных элементов управления.

Навигационная панель (nav bar или navigation bar). Стандартный элемент Web-интерфейса, на котором располагаются ссылки на основные категории содержимого сайта.

Намек (affordance). Визуальный ключ, который подсказывет функцию элемента дизайна. Например, объемный, со скошенными кромками дизайн стандартной HTML-кнопки наводит на мысль о возможности вызова какой-нибудь функции с помощью щелчка на кнопке, поскольку она выделяется на плоском, двухмерном фоне остальной части страницы.

Непосещенная ссылка (unvisited link). Ссылка на страницу, о посещении которой нет сведений в браузере.

Несинхронизированный (out of sync). Состояние Web-сайта, когда его локальные файлы не соответствуют дистанционным файлам.

Нижний выносной элемент (descender). Часть буквы или типографского символа, находящаяся под базовой линией текстовой строки, подобно ножке строчной буквы **р**.

Обратный отступ (hanging indent). Стиль абзаца, в котором первая строка текста выступает влево, а не вправо.

Объединение столбцов (colspan). Набор вертикальных столбцов, которые охватывает ячейка таблицы. Объединения столбцов отлично подходят для таблиц данных, но при создании макетных таблиц их лучше не применять! Вместо объединения столбцов используйте вложенные таблицы. Сравните с термином **объединение строк** (rowspan).

Объединение строк (rowspan). Набор горизонтальных строк, охватываемых ячейкой таблицы. Объединения строк отлично подходят для таблиц данных, но при создании макетных таблиц их лучше не применять! Вместо объединения строк используйте вложенные таблицы. Сравните с термином **объединение столбцов** (colspan).

Определение стиля (style definition). Набор атрибутов и значений каскадных таблиц стилей (CSS), описывающих порядок отображения браузером конкретного элемента.

Оптимизация (optimizing). Уменьшение размера файла изображения для более быстрой загрузки, при этом качество изображения сохраняется без изменений.

Отступы в ячейках (cellpadding). Величина внутреннего зазора между границами ячеек таблицы и содержимым этих ячеек. В макетных таблицах (layout table) этот атрибут обычно задается равным 0.

Палитра (palette). Встроенная в файл изображения таблица цветов. В файлы GIF и PNG входит палитра, содержащая до 256 цветов, тогда как файлы JPEG не имеют палитр и поэтому не ограничиваются 256 цветами.

Поле формы, объект формы (form field, form object). Названия элементов управления (widgets) или частей формы, которые используются для ввода текста в форму или для вызова команд посредством щелчков мышью на этих фрагментах.

Посещенная ссылка (visited link). Ссылка на страницу, о посещении которой в браузере хранятся сведения.

Правило стиля (style rule). Одиночный набор команд представления для браузера в каскадных таблицах стилей. Правило стиля состоит из **селектора стиля** и **определения стиля**.

Представление (presentation). Вид Web-страницы в браузере. Как правило, задачи представления решаются без использования HTML; предполагается, что браузер учитывает этот аспект. Однако традиционно HTML предоставляет **теги форматирования** (formatting tags), которые дают дизайнеру возможность управлять представлением содержимого без идентификации его структуры. Каскадные таблицы стилей (CSS) позволяют HTML снова стать чисто структурным языком.

Присоединение (attaching). Связывание внешнего файла JavaScript или каскадных таблиц стилей (CCS) с HTML-документом. Сравните с термином **внедрение** (embedding).

Проверка достоверности (validating) входных данных формы. Проверка на ошибки данных, вводимых посетителем в форму. Проверка достоверности формы помогает предотвратить повреждение плохими данными серверной части приложения, которое обрабатывает подачу форм.

Проверка достоверности (validating) страницы. Проверка исходного кода Web-страницы на соответствие стандартам.

Протокол передачи данных (FTP – File Transfer Protocol). Используемый компьютерами метод обмена файлами данных в Интернете.

Псевдокласс (pseudo-class). Специальный дополнительно подключаемый модуль для селектора тега-анкера, который сообщает браузеру, к какому состоянию ссылки применяется правило стиля.

Псевдоэлемент (pseudo-element). Специальный дополнительно подключаемый модуль для селектора HTML-тега, который идентифицирует определенный раздел текста внутри элемента, например, первую строку или первую букву.

Пункт назначения (destination). В ссылке это страница, которая загружается, когда посетитель щелкает на элементе источника.

Путь (path). Набор инструкций, которые сообщают браузеру о том, как получить некоторый определенный файл.

Путь относительно документа (document-relative path). Путь к файлу, определяемый на основе текущего местонахождения браузера. Ссылки с путями относительно документа работают в автономном режиме, но их формат полностью зависит от уровня на сайте, где они отображаются. Если переместить ссылку на другой уровень, путь перестанет работать. Пути, указываемые относительно документа, предназначены только для внутренних ссылок. Их не следует использовать для внешних ссылок.

Путь относительно корня (root-relative path). Путь к файлу, находящемуся на текущем Web-сайте. Как и в случае абсолютного пути, путь относительно корня всегда имеет один и тот же формат, независимо от места нахождения на сайте. Как и пути, указываемые относительно документа, «пути относительно корня» используются только для внутренних ссылок, и не применяются для внешних ссылок. Однако «пути относительно корня» работают только на Web-сервере. По этой причине большинство дизайнеров предпочитает использовать для внутренних ссылок пути, указываемые относительно документов.

Разделение на слои (slicing). Метод создания макетных таблиц в графических приложениях, таких, как Fireworks и Photoshop. Вначале разрабатывается макет в графическом редакторе, а затем макет разделяется на разделы или срезы, основанные на разделах страницы. При экспорте срезов в качестве отдельных графических файлов также создается HTML-таблица, с помощью которой отдельные части обратно собираются вместе в Web-браузере.

Разрешение (resolution). Размер пикселов в файле изображения: чем больше разрешение, тем меньше пикселы и резче изображение, но при этом увеличивается размер файла.

Робот (robot). Программное обеспечение механизма поиска, предназначенное для автоматического посещения Web-сайтов с целью каталогизации его содержимого.

Ролловерная графика (rollover graphic). Файл изображения, которое изменяет свой вид при наведении на него указателя мыши. На самом деле функция JavaScript производит замену на другой отдельный файл изображения.

С выключкой (justified). Стиль абзаца, в котором все строки текста имеют одинаковую длину. Чтобы удлинить более короткие строки, браузером вставляются пробелы между словами и символами. Сравните с термином **без выключки** (ragged).

С засечками (serif). Шрифт, такой, как Times New Roman, у которого на концах букв имеются мелкие украшения в виде засечек. Сравните с термином **без засечек** (sans serif).

Сглаживание (antialiasing). Метод создания видимости гладких, плавных кривых посредством вставки слегка размытых пикселов вдоль изогнутого края. Также означает процесс, которому подвергся Ноэль (Noel) из сериала *Фелисити* (Felicity), когда его жена, Дженифер Гарнер (Jennifer Garner), снимавшаяся в сериале «Шпионка» (Alias), бросила его.

Селектор дочернего объекта (child selector). Селектор стиля, определяющий все HTML-теги определенного типа, которые имеют в качестве своего непосредственного родителя тег другого типа, например, все абзацы, которые являются прямыми дочерними объектами ячеек таблицы.

Селектор идентификатора (ID selector). Селектор стиля, который применяется к одному и только к одному элементу на странице.

Селектор стиля (style selector). Индикатор каскадных таблиц стилей, который сообщает браузеру, к каким элементам применяется правило стиля.

Селектор элементов-братьев (sibling selector). Селектор стиля, который идентифицирует все HTML-теги определенного типа, следующие за тегом другого типа, но не являющиеся дочерними элементами данного тега, подобно всем абзацам, следующим за заголовками первого уровня.

Сжатие (compression). Процесс уменьшения размера файла изображения посредством отбрасывания части визуальной информации. При чрезмерно большом сжатии отбрасывается слишком много информации и качество изображения заметно ухудшается.

Синхронизированное (synchronized). Состояние Web-сайта, в котором его локальные файлы полностью соответствуют дистанционным файлам.

Слой (layer). Размещаемый на Web-странице блок содержимого, основанный на каскадных таблицах стилей. Слои не являются плоскими, как ячейки таблицы, поэтому можно накладывать их друг на друга и располагать в любом месте на экране. Еще одно имя для слоя – **элемент раздела** (div element).

Состояние ссылки (link state). Состояние гиперссылки. Обычно существует три состояния ссылки: **непосещенное** (unvisited), **посещенное** (visited) и **активное** (active), каждое окрашивается в свой цвет. Каскадные таблицы стилей добавляют четвертое состояние – **выделенное** (hover).

Состояния кнопки (button states). Различные представления, которые принимает графическое изображение кнопки в зависимости от вида взаимодействия

посетителя с кнопкой. Например, в обычном состоянии ролловера (rollover) имеется два состояния кнопки: состояние по умолчанию и вид кнопки, когда посетитель наводит на нее указатель мыши.

Специальные возможности (accessibility). Отличие хорошего Web-дизайна, когда сайт делается так, чтобы его содержимое было доступно для всех посетителей, включая инвалидов. Чтобы сделать сайт максимально доступным, приводите текстовое описание для всех чисто визуальных элементов, например, изображений, и соблюдайте соглашения о разметке страниц.

Ссылка электронной почты (email link). Ссылка, которая по щелчку на ней открывает пустое окно электронного сообщения.

Строка даты (dateline). Небольшой фрагмент текста, сообщающий текущую дату. Поместите строку даты на Web-страницу, чтобы создать впечатление (порой не соответствующее действительности) о частом обновлении данных.

Строка навигации (хлебные крошки) (breadcrumb trail). Навигационный элемент, который помогает посетителям сайта ориентироваться в иерархии сайта, отображающий связь текущей страницы с остальной окружающей структурой сайта.

Структура (structure) Web-сайта. Способ упорядочивания иерархии категорий и подкатегорий сайта.

Структура (structure) в языке HTML. Техническое описание, устанавливающее соответствие между фрагментами содержимого и отдельными частями Web-страницы. HTML-теги используются для обозначения структуры, которая отображается браузером в соответствии со встроенными стилями или каскадными таблицами стилей, задаваемыми дизайнером.

Таблица данных (data table). HTML-таблица, в которой строки и столбцы данных представляются согласно спецификациям HTML. Сравните с термином **макетная таблица** (layout table).

Таблица стилей (style sheet). Блок кода каскадных таблиц стилей, который находится между тегами стиля на странице HTML или в отдельном файле каскадных таблиц стилей (CSS). Таблица стилей состоит из одного или нескольких **правил стилей**.

Тег (tag). Ключевое слово HTML, которое определяет структуру части содержимого.

Тег форматирования (formatting tag). HTML-тег, описывающий представление блока содержимого без определения его структуры.

Текстовые эквиваленты (text equivalents). Текстовые описания нетекстового содержимого для использования средствами чтения с экрана или другими специальными средствами. Предоставление альтернативного текста для изображений является добавлением текстовых эквивалентов.

Текучий макет (liquid layout). Макет, размеры которого автоматически изменяются в соответствии с шириной окна браузера. Это то же самое, что и **автоматически изменяющийся макет** (expando layout).

Унифицированный указатель ресурса (URL – Universal Resource Locator). Точный адрес файла в Интернете.

Уровень (level). Положение папки или файла в иерархии структуры сайта.

Физический перенос (physical wrapping). Тип переноса слов в текстовой области, в которой браузер представляет текст с дополнительными обрывами строк, создаваемыми при переносе. Сравните с термином **виртуальный перенос** (virtual wrapping).

Хост Web-сайта (Web host). Компания (или частное лицо), предоставляющая сервер для Web-сайта.

Элемент раздела (div element). Еще одно название для **слоя** каскадных таблиц стилей. Название элемента говорит само за себя – это раздел Web-страницы.

Элемент управления (widget). Один из многих элементов, на которых щелкают кнопкой мыши или в которые вводят текст на форме. Элементы управления также называют **полями формы** и **объектами формы**.

Элемент фразы (phrase element). HTML-элемент, определяющий структуру сегмента текста внутри более крупного элемента, например, абзаца. Среди прочих, элементами фразы являются теги `em`, `strong`, `cite` и `dfn`.

Язык гипертекстовой разметки (HTML – Hypertext Markup Language). Компьютерный язык, служащий для описания содержимого Web-страниц.

Марк Кемпбел

Строим Web-сайты

ДИЗАЙН • HTML • CSS

Перевод с английского языка

Отдел распространения издательской группы «ТРИУМФ»:

- ✔ «Издательство Триумф»
- ✔ «Лучшие книги»
- ✔ «Технический бестселлер»
- ✔ «Только для взрослых»
- ✔ «Технолоджи – 3000»
- ✔ «25 КАДР»

Телефон: (095) 720-07-65, (095) 772-19-56.
E-mail: opt@triumph.ru

Интернет-магазин: www.3st.ru

КНИГА-ПОЧТОЙ: 125438, г.Москва, а/я 18 «Триумф».
E-mail: post@triumph.ru

Перевод *А.Г. Горлач, А.Ю. Климович*
Главный редактор *А.Г. Жадаев*
Научный редактор *С.В. Корсаков*
Ответственный редактор *К.В. Хомякова*
Выпускающий редактор *И.Г. Колмыкова*
Корректор *А.Н. Левина*
Верстка *В.Г. Баукова*
Дизайн обложки *И.Г. Колмыкова*

Подписано в печать с оригинал-макета 23.08.2005 г.
Формат $70\times100^{1}/_{16}$. Печать офсетная. Печ.л. 30. Заказ № 191.
Тираж 3 500 экз.

ООО «Издательство ТРИУМФ».
Россия, 125438, г. Москва, а/я 18.

Отпечатано в ОАО «Можайский полиграфический комбинат»
143200, г. Можайск, ул. Мира, 93