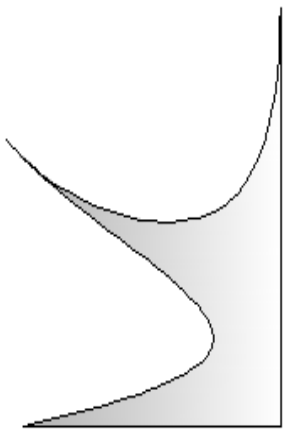
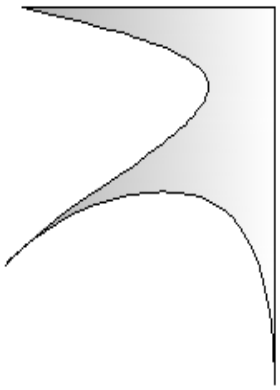


О.М. Шикула

Вступ до сучасного Web-дизайну: HTML5+CSS3



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

ІНСТИТУТ ПІСЛЯДИПЛОМНОЇ ОСВІТИ

О.М. Шикула

Вступ до сучасного Web-дизайну: HTML5+CSS3

Дисципліна: «Використання сервісів Google у роботі наукових працівників»

Навчальний посібник

Київ 2019

БКК 32.97
УДК 631.1301

Шикула О.М. **Вступ до сучасного Web-дизайну: HTML5+CSS3**: Навчальний посібник. / О.М. Шикула – К. ІПДО, 2019. – 240 с.

Анотація

В навчальному посібнику наведено практичну інструкцію по створенню сучасних Web-сайтів. Викладено мови HTML 5 и CSS 3, що використовуються відповідно для наповнення і форматування Web-сторінок.

Навчальний посібник розроблено на кафедрі інформатики та обчислювальної техніки Інституту післядипломної освіти Національного університету харчових технологій.

Призначено для широкого кола науковців, аспірантів, викладачів, науково-технічних працівників, професійна діяльність яких пов’язана з Web-дизайном.

Автор: О.М. Шикула, доктор фізико-математичних наук, професор

Редактор: Н.Я.КОСТІНА

РОЗДІЛ 1. ВМІСТ WEB-СТОРІНОК. МОВА HTML 5

ТЕМА 1. ВСТУП ДО СУЧАСНОГО WEB-ДИЗАЙНУ. WEB 2.0. СТВОРЕННЯ WEB-СТОРІНОК

***Основна мета*:** одержати представлення про можливості сучасного Web-дизайну, вибрати для роботи Браузер і Web-сервер, почати вивчати мову HTML, створити Web-сторінку.

Вміст Web-сторінок створюється за допомогою мови HTML, а зовнішній вигляд їх елементів визначається стилями, які описуються мовою CSS. Існує також можливість написати невеликі програми мовою Javascript, які вбудовуються в саму Web-сторінку й змінюють її вміст у відповідь на дії відвідувача, — Web-сценарії.

Всі ці мови і технології були створені досить давно, і до певного часу в них мало що мінялося (а в мові HTML не мінялося взагалі нічого), тобто в Web-дизайні не було ніяких революцій — тільки невеликі еволюційні зміни.

Але вже впроваджені нові стандарти, які описують чергові версії цих мов: HTML 5 і CSS 3. Вони принесли в Web-дизайн багато нового.

- Спрощену вставку на Web-сторінку аудіо- і відеоматеріалів.

- Можливості малювання на Web-сторінках.
- Багатоколоночну верстку тексту.
- Підтримку роботи в оффлайновому режимі (при відключенні від Інтернету).
- Додаткову підтримку мобільних пристроїв.
- Підтримку спеціальних Браузерів для осіб з обмеженими фізичними можливостями.
- І багато - багато чого іншого.

ВСТУП ДО СУЧАСНОГО WEB-ДИЗАЙНУ ТА МОВИ ОПИСУ ГІПЕРТЕКСТІВ (HTML)

World Wide Web та її призначення

World Wide Web (*WWW, "Всесвітня павутина"*) – найвідоміша і найпопулярніша служба Інтернету. Часто її ототожнюють з усією мережею Internet. Проте, Internet має кілька служб, зокрема, електронну пошту, телеконференції (дискусійні групи), протокол передачі файлів, тощо. Word Wide Web – глобальна розподілена інформаційна гіпертекстова система, яка дозволяє з'єднувати в одну систему розрізнену інформацію, що зберігається на різних комп'ютерах. WWW призначена для інтерактивного пошуку різнотипної інформації за певним способом обміну інформацією, чи протоколом. "Всесвітня павутина" побудована на основі протоколу передачі гіпертексту **http** (hypertext transfer protocol), який дозволяє передавати документи формату HTML з Web-серверів до Web-клієнтів.

Принципи сучасного Web-сайту

При створенні Web-сайту необхідно дотримувати трьох нескладні правил:

- При створенні Web-сторінок слід дотримуватися сучасних інтернет-стандартів. При цьому потрібно повністю відмовитися від застарілих і закритих інтернет-технологій, що не відповідають сучасній парадигмі Web-дизайну і найчастіше не підтримуються всіма Браузерами.
- Особливу увагу потрібно звернути на структуру і наповнення Web-сторінок. Структура Web-сторінок повинна бути максимально простою, а наповнення — достатнє багатим, щоб відвідувач швидко знайшов

потрібну йому інформацію. Крім того, необхідно створювати Web-сторінки так, щоб дизайн не заважав сприйняттю інформації.

- Web-сторінки обов'язково слід робити максимально доступними на будь-яких пристроях. Web-сторінки повинні швидко завантажуватися та виводитися на екран. Також Web-сторінки не повинні потрібувати для відображення ніякого додаткового програмного забезпечення.

Фактично тут ми навели постулати так званої концепції *Web 2.0.* Це список правил, яким повинен задовольняти будь-який Web-сайт, що претендує на звання сучасного.

Також концепція Web 2.0 передбачає чотири принципи, що є "переднім краєм" Web-дизайну.

Принцип перший — розділення вмісту, представлення і поведінки Web-сторінки. Тут *вміст* — це інформація, яка виводиться на Web-сторінці, *представлення* описує формат виводу цієї інформації, а *поведінка* — реакцію Web-сторінки або окремих її елементів на дії відвідувача. Завдяки цьому розділенню ми зможемо правити, скажемо, вміст, не торкаючись представлення і поведінки, або доручати створення вмісту, представлення і поведінки різним людям.

Принцип другий — *вміст, що підвантажується.* Замість того щоб обновляти всю Web-сторінку у відповідь на клацання на гіперпосиланні, ми можемо довантажувати тільки її частину, що містить необхідну інформацію. Це сильно зменшить об'єм переданої по мережі інформації (мережевний трафік) і дозволить виконувати які-небудь дії з даними після їх підвантаження.

Принцип третій — *вміст, що генерується.* Якась частина Web-сторінки може не завантажуватися по мережі, а генеруватися прямо на місці, у Браузері. Так ми ще сильніше скоротимо мережевний трафік.

Принцип четвертий — *семантична розмітка* даних. Вона дозволить нам зв'язати виведені на Web-сторінку дані згідно з якими-небудь правилами. Наприклад, ми можемо семантично зв'язати сторінки довідника по HTML, і відвідувач, завантаживши яку-небудь сторінку, зможе відразу ж перейти на пов'язані з нею сторінки, що містять додаткові або споріднені відомості.

Клієнти і сервери Інтернету. Інтернет-адреси

Як все це працює? Звідки Браузер отримує потрібну Web-сторінку? Хто відповідає за роботу складного механізму за назвою Всесвітня павутина?

Візьмемо для прикладу головну Web-сторінку Web-сайту, який ми відкрили. Вона повинна десь зберігатися. Але де? На диску іншого

комп'ютера, підключеного до мережі (у цьому випадку — до мережі Інтернет), який може належати як авторові Web-сайту, так і сторонній організації, що надає доступ в Інтернет *(інтернет-провайдеру)*. І зберігається вона у вигляді файлу або набору файлів, таких же, які удосталь "водяться" на нашому власному комп'ютері.

Але як ми змогли отримати і переглянути вміст цього файлу? По-перше, за допомогою самої мережі — вона зв'язала комп'ютер, що зберігає файл, з нашим. По-друге, за допомогою особливих програм, які, власне, і виконали передачу файлу. Ці програми діляться на дві групи.

Програми першої групи взаємодіють безпосередньо з користувачем: приймають від нього запити на інформацію, яка зберігається десь у мережі, отримують її, виводять на екран і, можливо, дозволяють її правити і відправляти назад. Такі програми називають *клієнтами.*

Для перегляду Web-сторінок ми користуємося Браузером. Це програма-клієнт; вона приймає від нас інтернет-адреси Web-сторінок, отримує файли, що зберігають їхній вміст, і виводить цей вміст на екран. Програма поштового клієнта дозволяє як добувати з поштової скриньки отримані листи, так і створювати нові. Існують також клієнти чата, систем миттєвих повідомлень та ін.

Але клієнти не мають прямого доступу до, що зберігається на інших комп'ютерах інформації. Вони не можуть просто "залізти" на жорсткий диск віддаленого комп'ютера і прочитати звідти файл. Так зроблене з міркувань безпеки. Замість цього вони відправляють запити програмам другої групи — серверам.

Сервери працюють на комп'ютерах, що зберігають інформацію, яка повинна бути доступна в мережі. Вони приймають запити від клієнтів, добувають необхідну інформацію з файлів і відправляють їм. Також вони можуть отримувати введену користувачами інформацію від клієнтів і зберігати їх у файлах, при цьому, можливо, якось обробивши. Можна сказати, що сервери виступають посередниками між клієнтами та запитуваної ними інформацією.

Для керування Web-сайтами використовуються *Web-сервери,* які приймають запити від клієнтів і відправляють їм вміст необхідних файлів. Для керування поштовими службами застосовуються сервери електронної пошти; вони зберігають отримані листи у файлах, видають їх поштовим клієнтам по запиту, приймають від клієнтів нові повідомлення та відправляють їх по зазначеній адресі — взагалі, працюють як поштова відділення. Служби чатів і миттєвих повідомлень також мають свої сервери.

Але як вказати, яка інформація і з якого сервера нам потрібна? За допомогою певним чином складеної інтернет-адреси.

Кожна одиниця інформації — файл, ящик електронної пошти, канал чату, — доступна в мережі, однозначно ідентифікується інтернет-адресою, яка являє собою рядок з букв, цифр і деяких інших символів.

Інтернет-адреса містить у собі дві частини:

- інтернет-адреса програми-сервера, що працює на комп'ютері;
- вказівник на потрібну одиницю інформації, наприклад, шлях до файлу, ім'я ящика електронної пошти, ім'я каналу чата та ін. (може бути відсутньою).

Розглянемо кілька прикладів інтернет-адрес.

В інтернет-адресі **http://www.somesite.ua/folder1/file1.htm** присутні обидві частини. Тут **http://www.somesite.ua** — інтернет-адреса програми-сервера, в цьому випадку — Web-сервера, а **/folder1/file1.htm** — шлях до запитуваного файлу.

В інтернет-адресі **http://www.othersite.ua** присутня тільки інтернет-адреса Web-серверу. Яка інформація в цьому випадку буде відправлена клієнтові (Браузеру), ми довідаємося потім.

А в адресі **user@mail.someserver.ua** ми бачимо інтернет-адресу сервера електронної пошти **(mail.someserver.ua)** і ім'я поштової скриньки **(user).**

Web-сайти та Web-сервери

Усі інтернет-програми діляться на клієнти і сервери. Клієнти працюють на боці користувача, отримують від нього інтернет-адреси і виводять йому отриману із цих адрес інформацію. Сервери приймають запити від клієнтів, знаходять запитувану ними інформацію на дисках серверних комп'ютерів і відправляють її клієнтам.

У Всесвітній павутині WWW як клієнти використовуються Браузери, а як сервери — Web-сервери. Це ми теж знаємо.

Будь-яка інформація на дисках комп'ютера зберігається у файлах. Web-сторінки також зберігаються у файлах з розширенням htm або html. Одна Web-сторінка займає один або більше файлів.

Web-сайт — це сукупність множини Web-сторінок, об'єднаних загальною темою та зв'язаних одна з одною за допомогою гіперпосилань. Отже, Web-сайт — це також набір файлів, що, можливо, зберігаються в різних папках, — так ними зручніше управляти.

А тепер розглянемо деякі подробиці роботи Web-серверів.

Насамперед, для зберігання всіх файлів, що складають Web-сайт, на диску серверного комп'ютера виділяється особлива папка, називана *кореневий папкою Web-сайту*. Шлях до цієї папки вказується в налаштуваннях Web-серверу, щоб він зміг її "знайти".

Всі файли, що складають Web-сайт, повинні зберігатися в кореневій папці або в папках, вкладених у неї. Файли, розташовані поза кореневою папкою, з точки зору Web-сервера не існують. Так зроблено для безпеки, щоб зловмисник не зміг отримати доступ до дисків серверного комп'ютера.

Коли в інтернет-адресі вказується шлях до запитуваного файлу, Web-сервер відраховує його відносно кореневої папки. Це найпростіше показати на прикладах.

- **http://www.somesite.ua/page1.htm** — у відповідь буде відправлений файл **page1.htm**, що зберігається в кореневій папці Web-сайту.
- **http://www.somesite.ua/chapter2/page6.htm** — у відповідь буде відправлений файл **page6.htm**, що зберігається в папці chapter2, яка вкладена в кореневу папку Web-сайту.
- **http://www.somesite.ua/downloads/others/archive.zip** — у відповідь буде відправлений файл **archive.zip**, що зберігається в папці others, вкладеної в папку **downloads**, яка, у свою чергу, вкладена в кореневу папку Web-сайту.

Але ж ми нечасто набираємо інтернет-адресу, що включає шлях до запитуваного файлу. Набагато частіше інтернет-адреси включають тільки адресу програми-сервера, наприклад, **http://www.somesite.ua.** Що в такому випадку робить Web-сервер? Який файл він відправляє у відповідь?

Спеціально для цього передбачені так звані *Web-сторінки за замовчуванням*. Така Web-сторінка видається клієнтові, якщо він указав в інтернет-адресі тільки шлях до файлу, але не його ім'я. Звичайно файл Web-сторінки за замовчуванням має ім'я **default.htm[l]** або **index.htm[l],** хоча його можна змінити в налаштуваннях Web-сервера.

Так, якщо ми наберемо інтернет-адресу **http://www.somesite.ua,** Web-сервер поверне нам файл Web-сторінки за замовчуванням, що зберігається в кореневій папці Web-сайту. Практично завжди це буде *головна Web-сторінка* — та, з якої починається "подорож" по Web-сайту.

Ми можемо набрати і інтернет-адресу виду **http://www.somesite.ua/chapter2/.** Тоді Web-сервер відправить нам файл Web-сторінки за замовчуванням, що зберігається в папці **chapter2**, вкладеної в кореневу папку Web-сайту.

ОСНОВНІ ПРИНЦИПИ СТВОРЕННЯ WEB-СТОРІНОК. МОВА HTML5

Web-сторінки найчастіше мають дуже строкатий вигляд: різнокаліброві шматки тексту, таблиці, картинки, врізання, виноски і навіть фільми. Але описується все це у вигляді звичайного тексту. Web-сторінки — це текстові файли, які можна створити за допомогою текстового редактора Блокнот, що поставляється в складі Windows! (Зрозуміло, підійде будь-який аналогічний текстовий редактор, наприклад Notepad++).

Для форматування вмісту Web-сторінок застосовується особлива мова — *HTML* (Hypertext Markup Language, мова гіпертекстової розмітки). За допомогою команд — *тегів* — цієї мови створюють і абзаци тексту, і заголовки, і врізання, і навіть таблиці.

Перша версія мови HTML з'явилася дуже давно, ще в 1992 році. В теперішній час діє нова версія HTML5.

Мова HTML і її теги

Вивчати HTML найкраще на прикладі. Так що давайте відразу ж створимо нашу першу Web-сторінку. *Windows* уже містить необхідний для цього інструмент — *Блокнот.*

Примітка. Взагалі, для створення Web-сторінок існує багато спеціальних програм — Web-редакторів. Вони дозволяють працювати з Web-сторінками, навіть не знаючи HTML, — як з документами Microsoft Word, просто набираючи текст і форматувати його. Також вони стежать за правильністю розміщення тегів, допоможуть швидко створити складний елемент Web-сторінки і навіть опублікувати готовий Web-сайт у мережі. До таких програм належить, зокрема, відомий Web-Редактор Adobe Dreamweaver.

Однак ми поки що будемо користуватися найпростішим текстовим редактором Блокнот. Це дозволить нам краще познайомитися з HTML.

Відкриємо Блокнот і наберемо в ньому текст (код), наведений у лістингу 1.1.

Лістинг 1.1

```
<!DOCTYPE html>
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html;
    charset=utf-8">
    <TITLE>Приклад Web-сторінки</TITLE>
  </HEAD>
```

```
  <BODY>
      < H1>Довідник по HTML</H1>
      </P> Вітаємо на нашому Web-сайті всіх, хто
      займається Web-дизайном! Тут ви зможете знайти
      інформацію про всі інтернет-технології,
      застосовуваних при створенні Web-сторінок. Зокрема,
      про мову <STRONG>HTML</STRONG></P>
  </BODY>
</HTML>
```

Перевіримо набраний код на помилки і збережемо у файл із ім'ям 1.1.htm. При цьому:

1. Виберемо в списку, що розкривається, *Тип файла* пункт *Все файлы*. Інакше Блокнот додасть до нього розширення txt, і наш файл одержить ім'я 1.1.htm.txt.
2. Збережемо HTML-код у кодуванні UTF-8. Для цього в діалоговому вікні збереження файлу Блокнота знайдемо список, що розкривається, *Кодировка* і виберемо в ньому пункт *UTF-8***.**

Все, наша перша Web-сторінка готова! Тепер залишилося відкрити її в Браузері і подивитися на результат.

HTML 5 підтримують останні версії Mozilla Firefox, Opera, Apple Safari і Google Chrome, тому переважніше яка-небудь із цих програм.

Відкриємо ж Web-сторінку в вибраному Браузері (Firefox) і подивимося на неї (рис. 1.1).

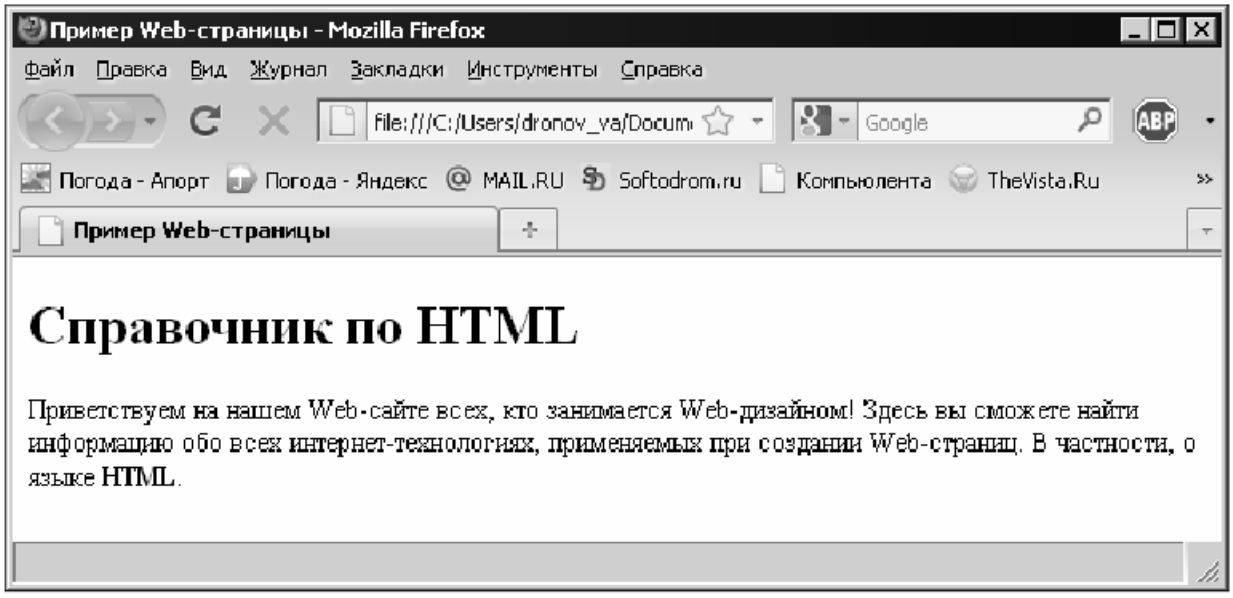


Рис. 1.1. **Перша Web-сторінка**

Ми створили Web-сторінку, що містить великий "кричущий" заголовок, абзац тексту, який автоматично розбивається на рядки, а також фрагмент тексту, виділений напівжирним шрифтом (абревіатура "HTML").

В частині лістингу, що вкладена між **<BODY>** і **</BODY>** ми бачимо текст заголовка і абзацу. І слова, узяті в кутові дужки — символи < і >. Це і є теги HTML. Вони перетворюють той або інший фрагмент HTML-коду в визначений елемент Web-сторінки: абзац, заголовок або текст, виділений напівжирним шрифтом.

Почнемо з тегів **<H1>** і **</H1>.** Ці теги перетворюють фрагмент тексту, що знаходиться між ними, у заголовок. Тег **<H1>** позначає початок фрагмента, на який поширюється дія тегу, і називається *відкриваючим.* А тег **</H1>** встановлює кінець "охоплюваного" фрагмента і називається *закриваючим.* Що стосується самого фрагмента, вкладеного між відкриваючим і закриваючим тегами, то він називається *вмістом тегу.* Саме до вмісту застосовується дія тегу.

Усі теги HTML являють собою символи < і >, всередині яких знаходиться *ім'я тегу,* що визначає призначення тегу. Закриваючий тег повинен мати те ж ім'я, що і відкриваючий; єдина відмінність закриваючого тегу — символ /, який ставиться між символом < і ім'ям тегу.

Розглянуті нами теги **<H1>** і **</H1>** в HTML фактично вважаються одним тегом **<H1>.** Такий тег називається *парним.*

Далі. Парний тег **<P>** створює на Web-сторінці абзац; вміст тегу стане текстом цього абзацу. Такий абзац буде відображатися з відступами зверху і знизу. Якщо він повністю поміщається по ширині у вікні Браузера, то відобразиться в один рядок; а якщо ні, то сам Браузер розіб'є його на кілька більш коротких рядків. (Те ж справедливо і для заголовка.)

Парний тег **<STRONG>** виводить свій вміст напівжирним шрифтом. Як бачимо, тег **<STRONG>** вкладений всередину вмісту тегу **<P>.** Це значить, що вміст тегу **<STRONG>** буде відображатися як частина абзацу (тегу **<P>**) **.**

Давайте заради інтересу виділимо слова " Web-дизайном" курсивом. Для цього помістимо відповідний фрагмент тексту абзацу в парний тег **<EM>:**

```
<P>Вітаємо на нашім Web-сайті всіх, хто займається <EM>
Web-дизайном</EM>! Тут ви зможете знайти інформацію про
всі інтернет-технології, застосовувані при створенні Web-
сторінок. Зокрема, про мову <STRONG>HTML</STRONG></P>
```

Збережемо виправлену Web-сторінку і обновимо вміст вікна Браузера, натиснувши клавішу **<F5>.**

Розглянемо найважливіші правила, згідно з якими пишеться HTML-код.

- Імена тегів можна писати як прописними (великими), так і рядковими (малими) буквами. Традиційно в мові HTML імена тегів пишуть прописними буквами.
- Між символами <, >, / і іменами тегів, а також усередині імен тегів не допускаються пробіли і перенесення рядків.
- В звичайному тексті, що не є тегом, не повинні бути присутнім символи < і >. (Ці символи називають *неприпустимими*.) А якщо ні, то Браузер визнає фрагмент тексту, де зустрічається один із цих символів, тегом і відобразить Web-сторінку некоректно.

Вкладеність тегів

Якщо ми знову подивимося на приведений у лістингу фрагмент HTML-коду, то помітимо, що одні теги вкладені в інші. Так, тег **<STRONG>** вкладено в тег **<P>,** являючись частиною його вмісту. Тег **<P>**, в свою чергу, вкладено у тег **<BODY>,** а той — в "глобальний" тег **<HTML>.** Така *вкладеність тегів* в HTML — звичайне явище.

Коли Браузер зустрічає тег, вкладений в інший тег, він як би накладає дію "внутрішнього" тегу на ефект "зовнішнього". Так, дію тегу **<STRONG>** буде накладено на дію тегу **<P>,** і фрагмент абзацу буде виділеним напівжирним шрифтом, при цьому залишаючись частиною цього абзацу.

Давайте для прикладу текст " Web-дизайн", який ми недавно помістили в тег **<EM>,** вкладемо ще й у тег **<STRONG>.** От так:

```
<P>Вітаємо на нашому Web-сайті всіх, хто займається
<EM><STRONG> Web-дизайном</STRONG></EM>! Тут ви зможете
знайти інформацію про всі... інтернет-технології,
застосовувані при створенні Web-сторінок. Зокрема, про мову
<STRONG>HTML</STRONG></P>
```

У цьому випадку даний текст буде виділений напівжирним курсивом. Іншими словами, дія тегу **<STRONG>** буде накладено на дію тегу **<EM>.**

Порядок проходження закриваючих тегів повинен бути зворотним тому, у якім випливають теги відкриваючі. Говорячи інакше, теги з усім їхнім умістом повинні повністю вкладатися в інші теги.

Якщо ж ми порушимо це правило й напишемо такий HTML-код (зверніть увагу на спеціально переплутаний порядок проходження відкриваючих тегів):

```
<P>Вітаємо на нашім Web-Сайті всіх, хто займається
<EM><STRONG> Web-Дизайном</EM></STRONG>! Тут ви зможете
знайти інформацію про всі інтернет-технологіях,
застосовуваних при створенні Web-Сторінок. Зокрема, про
мову <STRONG>HTML</STRONG></P>
```

Браузер може відобразити нашу Web-сторінку неправильно.

Залишилося вивчити кілька нових термінів. Тег, в який безпосередньо вкладено даний тег, називається *батьківським,* або *батьком.* В свою чергу, тег, вкладений в даний тег, називається *дочірнім,* або *нащадком.* Так, для тегу **<EM>** в наведеному далі прикладі тег **<P>** — батьківський, а тег **<STRONG>** — дочірній. Будь-який тег може мати скільки завгодно дочірніх тегів, але тільки один батьківський (що, втім, зрозуміло — не може ж він бути безпосередньо вкладений одночасно у два теги).

Елемент Web-сторінки, в який вкладено елемент, створюваний даним тегом, називається *батьківським,* або *батьком.* А елемент Web-сторінки, який вкладено в даний елемент, — *дочірнім,* або *нащадком.* Те ж саме, що і в випадку тегів.

Рівень вкладеності того або іншого тегу показує кількість тегів, в які він послідовно вкладений. Якщо прийняти за точку відліку тег **<P>,** то тег **<EM>** буде мати перший рівень вкладеності, тому що він вкладений безпосередньо в тег **<P>.** Тег **<STRONG>** же буде мати другий рівень вкладеності, оскільки він вкладений в тег **<EM>,** а той, в свою чергу, — в тег **<P>.** В складних ж Web-сторінках рівень вкладеності інших тегів може складати кілька десятків.

Рівень вкладеності тегів в HTML-коді позначають за допомогою відступів, які ставлять ліворуч від відповідного тегу і створюють за допомогою пробілів (лістинг 1.1). На відображення Web-сторінки вони ніяк не впливають.

Секції Web-сторінки

Знову повернемося до HTML-коду нашої Web-сторінки. Подумки видалимо з нього вже розглянутий фрагмент і отримаємо лістинг 1.2.

Лістинг 1.2

```
<!DOCTYPE html>
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html;
    charset=utf-8">
```

```
    <TITLE>Приклад Web-сторінки</TITLE>
  </HEAD>
  <BODY>
     ..............................................
  </BODY>
</HTML>
```

Тут застосовані кілька тегів — так звані *невидимі теги* — теги, вміст яких ніяк не відображається Браузером. Вони зберігають відомості про параметри самої Web-сторінки і поділяють її на дві *секції,* що мають принципово різне призначення.

Секція тіла Web-сторінки знаходиться всередині парного тегу **<BODY>.** Вона описує сам вміст Web-сторінки, те, що буде виведено на екран. Саме секцію тіла ми розглядали раніше.

А в парному тегу **<HEAD>** знаходиться *секція заголовка* Web-сторінки. (Не плутати із заголовком, який створюється за допомогою тегу **<H1>**!) У цю секцію поміщають відомості про параметри Web-сторінки, не відображувані на екрані і призначені винятково для Браузера.

І заголовок, і тіло Web-сторінки знаходяться всередині парного тегу **<HTML>,** який розташований на самому вищому (нульовому) рівні вкладеності і не має батька.

Будь-яка Web-сторінка повинна бути правильно відформатована: мати секції заголовка і тіла та всі відповідні їм теги. Тільки в такому випадку вона буде вважатися коректної з погляду стандартів HTML.

Метадані та тип Web-сторінки

Повернемося до відомостей про параметри Web-сторінки, які знаходяться у секції її заголовка.

Параметри Web-сторінки, які не відображаються на екрані та призначені для Браузера, назвемо *метаданими.* Це свого роду дані, що описують інші дані, в нашім випадку — Web-сторінку. А HTML-теги, які задають метадані, називаються *метатегами.*

Насамперед, в метадані входить *назва* Web-сторінки. Вона відображається в заголовку вікна Браузера, де виводиться вміст даної Web-сторінки, і зберігається в "історії" (списку Web-сторінок, що відвідано до теперішнього часу). Назва поміщається в парний тег **<TITLE>** і розташовується в секції заголовка Web-сторінки:

```
<HEAD>
   <TITLE>Приклад Web-Сторінки</TITLE>
</HEAD>
```

Далі, звичайно в секції заголовка розташований особливий метатег, що задає кодування, у якому збережена Web-сторінка. Цей метатег має "красномовне" ім'я **<META>**:

```
<HEAD>
  <META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=utf-8">
</HEAD>
```

Наведений тег задає кодування UTF-8, у якому ми зберегли нашу Web-сторінку. Існують аналогічні теги, що задають кодування 1251, КОИ-8, кодування західноєвропейських і східноазійських мов та ін.

Примітка. Кодування *UTF-8* — це різновид кодування Unicode, призначене для Web-дизайну. Кодування Unicode (а виходить, і UTF-8) може закодувати всі символи всіх мов, наявних на Землі. Саме воне в цей час найчастіше застосовується для створення Web-сторінок.

У тега **<META>** немає ні вмісту, ні пари, що закриває. Це так званий *одинарний* тег, який має тільки відкриваючу частину. Такий тег діє в тій точці HTML-коду, де він сам знаходиться, і або задає метадані, або поміщає у відповідне місце Web-сторінки який-небудь елемент, що не відноситься до тексту.

Тепер залишилося розглянути останній тег, що знаходиться на самому початку HTML-коду нашої Web-сторінки. Цей тег знаходиться навіть поза "всеосяжним" тегом **<HTML>.**

```
<!DOCTYPE html>
```

Метатег **<!DOCTYPE>** задає, по-перше, версію мови HTML, на якій написана Web-сторінка, а по-друге, різновид даної версії. Так, існують позначка-теги **<!DOCTYPE>,** що вказують на HTML 5, HTML 4.01 (це попередня версія мови HTML, що ще діє на даний момент) і мова XHTML (відгалуження HTML, що має трохи інший синтаксис).

Метатег **<!DOCTYPE html>,** який ми поставили в початок нашої Web-сторінки, вказує на HTML 5.

Атрибути HTML-тегів

Останнє важливе питання — атрибути HTML-тегів.

Подивимося на тег **<META>**, що задає кодування Web-сторінки:

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
```

Тут ми бачимо, що між символами < і >, крім імені тегу, присутні ще якісь дані. Це *атрибути тегу, що* задають його параметри. Зокрема, два атрибути даного тегу **<META>** вказують, що документ являє собою Web-сторінку, і задають її кодування.

Кожний атрибут тегу має *ім'я,* за яким ставиться знак рівності, і *значення* даного атрибута, взяте в подвійні лапки. Так, атрибут з ім'ям **HTTP-EQUIV** має значення **"Content-Type",** що говорить про те, що даний метатег задає тип документа. А атрибут з ім'ям **CONTENT** має значення **"text/html; charset=utf-8",** що показує, що даний документ є Web-сторінкою, та вказує, що вона набрана в кодуванні UTF-8.

Атрибути тегів бувають обов'язковими та необов'язковими. *Обов'язкові* атрибути повинні бути присутні в тегу в обов'язковому порядку. *Необов'язкові* ж атрибути можуть бути опущені; в такому випадку тег поводиться так, начебто відповідному атрибуту привласнено значення за замовчуванням.

Атрибути **HTTP-EQUIV** і **CONTENT** тегу **<META>** обов'язкові. А от атрибут ID, підтримуваний практично всіма тегами HTML, необов'язковий, він використовується тільки в особливих випадках:

```
<H1 ID="header1">Довідник по HTML</H1>
```

Раніше ми вивчили три правила написання HTML-коду. Додамо до них ще шість.

- Імена атрибутів тегів можуть бути написані як прописними (великими), так і рядковими (малими) буквами. Традиційно в мові HTML імена атрибутів тегів пишуть прописними буквами, а їх значення — рядковими, якщо, звісно, значення не чутливе до регістру букв.
- Імена атрибутів тегів пишуть між символами < і > після імені тегу і відокремлюють від нього пробілом або розривом рядка. Якщо в тегу присутні кілька атрибутів, їх відокремлюють друг від друга також пробілами або розривами рядка.
- Всередині імен атрибутів не повинні бути присутні пробіли, інакше Браузер буде вважати, що це не один атрибут, а декілька.

- Значення атрибута тегу пишуть після його імені і поміщають у подвійні лапки. Між іменем атрибута тегу і його значенням ставлять знак рівності.
- Між іменем атрибута тегу, знаком рівності і відкриваючими лапками можуть бути присутнім пробіли або розриви рядків.
- Символи подвійних лапок неприпустимі і не повинні бути присутніми у звичайному тексті, інакше Браузер буде вважати наступний за ними текст значенням атрибута тегу.

Програми, якими ми будемо користуватися

Не враховуючи Блокнота (або аналогічного текстового редактора), будемо працювати із двома програмами: Браузер і Web-сервер.

Браузер

Браузери: Microsoft Internet Explorer, Mozilla Firefox, Opera, Google Chrome і Apple Safari.

Зрозуміло, усі ці програми підтримують мови HTML, CSS (мова опису каскадних таблиць стилів) і Javascript (мова, на якій пишуться Web-сценарії). Більшість цих програм підтримує деякий набір можливостей HTML 5 і CSS 3:

- Mozilla Firefox — починаючи з версії 3.5;
- Opera — починаючи з версії 9.5;
- Google Chrome — починаючи з версії 2.0.158.0;
- Apple Safari — починаючи з версії 4.0.

Web-сервер

Коли ми тестували нашу першу Web-сторінку, то прекрасно обійшлися без Web-сервера, відкривши її прямо в Браузері. Але надалі, особливо коли ми почнемо реалізовувати підвантаження вмісту, Web-сервер все-таки нам знадобиться. Багато Web-сценаріїв нормально працюють тільки в тому випадку, якщо Web-сторінка завантажена з Web-сервера. Це зроблене заради безпеки.

У складі Windows XP/2003/Vista/Se7en поставляється цілком серйозний Web-сервер Microsoft Internet Information Services.

Тепер випробуємо Web-сервер у дії. Для цього нам знадобиться будь-яка програма керування файлами (наприклад, що поставляється в складі Windows Провідник). Відкриємо в ній кореневу папку створеного при установці Web-сайту (у випадку Internet Information Services це папка C:\Inetpub\wwwroot).

Видалимо з кореневої папки всі наявні там файли, щоб вони нам не заважали. І скопіюємо туди нашу Web-сторінку **l.l.htm**.

Тепер відкриємо вибраний раніше Браузер і наберемо в ньому інтернет-адресу **http://localhost/1.1.htm.** Інтернет-адреса **http://localhost** ідентифікує наш власний комп'ютер *(локальний хост),* а **/l.l.htm** — вказівка на файл **l.l.htm**, що зберігається в кореневій папці Web-сайту.

Якщо ми все зробили правильно, Браузер відобразить нашу Web-сторінку. Виходить, Web-сервер працює.

КОНТРОЛЬНІ ПИТАННЯ

1. World Wide Web та її призначення .
2. Принципи сучасного Web-сайту.
3. Клієнти і сервери Інтернету. Інтернет-адреси.
4. Web-сайти і Web-сервери.
5. Мова HTML та її теги.
6. Вкладеність тегів.
7. Секції Web-сторінки.
8. Метадані та тип Web-сторінки.
9. Атрибути HTML-тегів.
10. Які програми ми вибрали в якості Браузера і Web-сервера?

ТЕМА 2. СТРУКТУРУВАННЯ ТЕКСТУ

***Основна мета*:** навчитися структурувати текст для його зручного читання: поділяти текст Web-Сторінки на логічні "шматки" — абзаци, створювати заголовки, списки, цитати, текст фіксованого формату та горизонтальні лінії.

СТРУКТУРУВАННЯ ТЕКСТУ

Абзаци
Заголовки
Списки
Цитати
Текст фіксованого формату
Горизонтальні лінії
Адреси
Дата
Коментарі

КОНТРОЛЬНІ ПИТАННЯ

СТРУКТУРУВАННЯ ТЕКСТУ

Продовжимо вивчення мови HTML і розглянемо засоби для структурування тексту — розбивки його на окремі значущі фрагменти, що мають різне призначення та несуть різний смисл. Такими фрагментами є абзаци, заголовки, списки і цитати.

Абзаци

Якщо оформити текст у вигляді великого монолітного "шматка", його навряд хтось буде читати. Саме тому текст завжди розбивають на абзаци. Мова HTML для створення абзацу використовує парний тег **<P>.** Вміст цього тегу стає текстом абзацу:

```
<P>Я — зовсім короткий абзац</P>
<P>А я — уже більш довгий абзац. Можливо, Браузер розіб'є
мене на два рядки</P>
```

Абзац HTML відокремлюється невеликим відступом від попереднього та наступного елементів сторінки. Якщо абзац повністю поміщається по

ширині в батьківський елемент Web-сторінки, він буде виведений в один рядок; а якщо ні, то Браузер розіб'є його текст на декілька більш коротких рядків.

Абзац — це незалежний елемент Web-сторінки, який відображається окремо від інших елементів. Такі елементи Web-сторінки називаються *блоковими,* або *блоками.*

Правила відображення тексту абзацу Браузером:

- два і більш пробілів, що йдуть один за одним, вважаються за один пробіл;
- перенесення рядка вважається за пробіл;
- пробіли і перенесення рядка між тегами, що створюють блокові елементи, ніяк не відображаються на Web-сторінці. (Завдяки цьому ми можемо форматувати HTML-код для більш зручного читання, у тому числі ставити відступи для позначення вкладеності тегів.).

  Ці ж правила слушні для інших блокових елементів.

Давайте створимо головну Web-сторінку нашого першого Web-сайту — довідника по HTML і CSS. Відкриємо Блокнот і наберемо в ньому HTML-код, наведений в лістингу 2.1.

Лістинг 2.1

```
<!DOCTYPE html>
<HTML>
  <HEAD>
    <META   HTTP-EQUIV="Content-Type"   CONTENT="text/html;
    charset=utf-8">
    <TITLE>Довідник по HTML і CSS</TITLE>
  </HEAD>
   <BODY>
    <P>Довідник по HTML і CSS</P>
    <P>Вітаємо на нашім Web-сайті всіх, хто займається
    Web-дизайном! Тут ви зможете знайти інформацію про
    всі інтернет-технології, що застосовуються при
    створенні Web-сторінок. А саме, про мови HTML і
    CSS.</P>
    <P>Вікіпедія визначає термін HTML так: </P>
    <P>HTML (від англ. Hypertext Markup Language — мова
    розмітки гіпертексту) — стандартна мова розмітки
    документів у Всесвітній павутині</P>
```

```
    <P>HTML дозволяє формувати на Web-сторінках наступні
    елементи:</P>
    <P>абзаци;</P>
    <P>заголовки;</P>
    <P>цитати;</P>
    <P>списки;</P>
    <P>таблиці; </P>
    <P>графічні зображення;</P>
    <P>аудіо- і відеоролики</P>
    <P>осНовНі принципи HTML</P>
     <P>. .,</P>
    <P>Теги HTML</P>
    <P>!DOCTYPE, BODY, EM, HEAD, HTML, META, P, STRONG,
    TITLE</P>
  </BODY>
</HTML>
```

Поки що це тільки заготовка для головної Web-сторінки. Пізніше ми будемо доповнювати та правити її.

Створимо папку, куди будемо "складати" файли, що складають наш Web-сайт. І збережемо в цій папці набраний HTML-код, давши файлу ім'я **index.htm.** Файл головної Web-сторінки Web-сайту повинен мати ім'я **in dex.htm[l]** (або **default.htm[l],** але це зустрічається рідше).

Відразу ж відкриємо створену Web-сторінку в Браузері — так ми відразу зможемо визначити, чи є помилки. Якщо помилки все є, виправимо їх.

Поки що наша Web-сторінка містить одні абзаци. Перше, що ми повинні в неї додати, — це...

Заголовки

Крім абзаців, великий текст для зручності читання і пошуку в ньому потрібного фрагмента звичайно ділять на більші частини: параграфи, глави, розділи. HTML не надає засобів для такого структурування тексту. Але він дозволяє створити заголовки, які ділять текст на частини, принаймні, візуально. Як в звичайній "паперовій" книзі.

Насамперед, усвідомимо, що в HTML є поняття *рівня заголовка,* яке вказує, наскільки велику частину тексту відкриває даний заголовок. Усього таких рівнів шість, і позначаються вони числами від 1 до 6.

- Заголовок першого рівня (1) відкриває найбільшу частину тексту. Як правило, це заголовок усієї Web-сторінки. Браузер виводить заголовок першого рівня самим великим шрифтом.
- Заголовок другого рівня (2) відкриває більш дрібну частину тексту. Звичайно це великий розділ. Браузер виводить заголовок другого рівня меншим шрифтом, чим заголовок першого рівня.
- Заголовок третього рівня (3) відкриває ще більш дрібну частину тексту; звичайно главу в розділі. Браузер виводить такий заголовок ще меншим шрифтом.
- Заголовки четвертого, п'ятого і шостого рівнів (4-6) відкривають окремі параграфи, великі, більш дрібні так самі дрібні відповідно. Браузер виводить заголовки четвертого і п'ятого рівня ще меншим шрифтом, а заголовок шостого рівня — тим же шрифтом, що і звичайні абзаци, тільки напівжирним.

На Web-сторінках невеликого та середнього розміру звичайно застосовують заголовки першого, другого і третього рівня. Менші рівні використовуються тільки на дуже більших Web-сторінках, що містять складно структурований текст.

Для створення заголовка HTML надає парний тег **<H*n*>**, де ***n*** — рівень заголовка. Вміст цього тегу стане текстом заголовка (лістинг 2.2).

Лістинг 2.2

```
<H1>Я — заголовок Web-сторінки, найголовніший</H1>
<H2>Я — заголовок розділа</H2>
<H3>Я — заголовок глави</H3>
<H4>Я — заголовок великого параграфа</H4>
<H5>Я — заголовок меншого параграфа</H5>
<H6>А я — заголовок маленького параграфа</H6>
```

Заголовок також належить до блокових елементів Web-сторінки. При його виведенні на екран Браузер використовує ті же правила, що і для абзацу.

Заголовки — це те, чого не вистачає нашій Web-сторінці **index.htm**. Давайте їх додамо (лістинг 2.3).

Лістинг 2.3

```
<H1>Довідник по HTML і CSS</H1>
.....
<H2 >Основні принципи HTML</H2>
.....
```

```
<H2>Теги HTML</H2>
```

Ми просто замінили теги <**P**> у відповідних фрагментах HTML-коду на теги **<H1>** і **<H2>.** Тепер можемо відкрити Web- сторінку в Браузері та подивитися на результат.

Списки

Списки використовуються для того, щоб представити читачеві перелік яких-небудь позицій, пронумерованих або відзначених маркером, — пунктів списку. Список із пронумерованими пунктами так і називається — *нумерованим,* а з відзначеними маркером — *маркіруванимм.* У маркірованих списках пункти позначаються особливим значком — *маркером,* який ставиться лівіше пункту списку.

Маркіровані списки звичайно використовуються для простого перерахування яких-небудь позицій, порядок проходження яких не важливий. Якщо ж слід звернути увагу читача на те, що позиції повинні іти одна за одною саме в тому порядку, у якому вони перераховані, слід застосувати нумерований список.

Браузер виводить список з відступом ліворуч. Відстань між пунктами списку він робить меншими, чим у випадку абзаців або заголовків. Також він сам розставляє необхідні маркери або нумерацію.

Будь-який список в HTML створюється у два етапи. Спочатку пишуть рядки, які стануть пунктами списку, і кожен із цих рядків поміщають всередину парного тегу **<LI>.** Потім всі ці пункти поміщають всередину парного тегу **<UL>** (якщо створюється маркірований список) або **<OL>** (при створенні нумерованого списку) — ці теги визначають сам список (лістинг 2.4).

Лістинг 2.4

```
<UL>
    <LI>Я — перший пункт маркірованого списку</LI>
    <LI>Я — другий пункт маркірованого списку</LI>
     <LI>Я — третій пункт маркірованого списку</LI>
</UL>
<OL>
    <LI>Я — першим пункт нумерованого списку</LI>
     <LI>Я — другий пункт нумерованого списку</LI>
    <LI>Я — третій пункт нумерованого списку</LI>
```

```
</OL>
```

Списки можна поміщати друг у друга, створюючи *вкладені списки.* Робиться це в такий спосіб. Спочатку в "зовнішньому" списку (в який повинен бути поміщений вкладений) відшукують пункт, після якого повинен знаходитися вкладений список. Потім HTML-код, що створює вкладений список, поміщають у розрив між текстом цього пункту і його закриваючим тегом **</LI>.** Якщо ж вкладений список повинен міститися на початку "зовнішнього" списку, його слід вставити між відкриваючим тегом **<LI>** першого пункту "зовнішнього" списку і його текстом. Що, втім, логічно.

В лістингу 2.5 представлений HTML-код, що створює два списки, один з яких вкладений усередину іншого.

Лістинг 2.5

```
<UL>
    <LI>Я — першим пункт зовнішнього списку. </LI>
    <LI>Я — другий пункт зовнішнього списку.
      <UL>
        <LI>Я — першим пункт вкладеного списку. </LI>
        <LI>Я — другий пункт вкладеного списку. </LI>
        <LI>Я — третій пункт вкладеного списку. </LI>
      </UL>
    </LI>
    <LI>Я — третій пункт зовнішнього списку</LI>
</UL>
```

HTML дозволяє вкладати нумерований список всередину маркірованого і навпаки. Глибина вкладення списків не обмежена.

Ще HTML дозволяє створити так званий *список визначень, що* представляє собою перелік термінів і їх роз'яснень. Такий список створюють за допомогою парного тегу **<DL>.** Всередині нього поміщають пари "термін — роз'яснення", причому терміни розміщають у парний тег <DT>, а роз'яснення — у парний тег **<DD>** (лістинг 2.6).

Лістинг 2.6

```
<DL>
    <DT>Вміст</DT>
    <DD>Інформація, відображувана на Web-сторінці</DD>
    <DT>Представлення</DT>
```

```
    <DD>Набір правил, що визначають формат відображення
    вмісту</DD>
    <DT>Поведінка</DT>
    <DD>Набір правил, що визначають реакцію Web-сторінки
    або її елементів на дії відвідувача</DD>
</DL>
```

Залишилося сказати, що списки і їх пункти належать до блокових елементів Web-сторінки, і при їхньому виведенні на екран Браузер керується тими ж правилами, що й при виведенні абзаців і заголовків.

На нашій Web-сторінці є кілька абзаців, що перераховують основні можливості HTML. Перетворимо їх у маркірований список (лістинг 2.7).

Лістинг 2.7

```
<UL>
    <LI>абзаци;</LI>
    <LI>заголовки;</LI>
    <LI>цитати;</LI>
    <LI>списки;</LI>
    <LI>таблиці;</LI>
     <LI>графічні зображення;</LI>
    <LI>аудіо- і відеоролики</LI>
 </UL>
```

Цитати

У тексті Web-сторінки, яку ми створюємо, присутня велика цитата з Вікіпедії. Давайте її якось виділимо.

HTML уже приготував для нас вихід із становища — парний тег **<BLOCKQUOTE>,** всередині якого розміщається HTML-код, що створює цитату (лістинг 2.8). Браузер виводить цитату з відступом ліворуч.

Лістинг 2.8

```
<BLOCKQUOTE>
   <P> Я- початок великої цитати</P>
   <P>Я — продовження великої цитати</P>
</BLOCKQUOTE>
```

Як бачимо, в тег **<BLOCKQUOTE>** можна помістити кілька абзаців. Там також можуть бути заголовки і списки (якщо вже виникне така необхідність).

Велика цитата HTML також належить до блокових елементів.

Залишилося тільки зробити те, що було задумано, — оформити цитату (лістинг 2.9).

Лістинг 2.9

```
<BLOCKQUOTE>
   <P>HTML (від англ. Hypertext Markup Language — мова
   розмітки гіпертексту) — стандартна мова розмітки
   документів у Всесвітній павутині,</P>
 </BLOCKQUOTE>
```

Текст фіксованого формату

Давайте зробимо нову Web-сторінку. Нова Web-сторінка (лістинг 2.10) буде присвячена тегу **<TITLE>.**

Лістинг 2.10

```
<!DOCTYPE html>
<HTML>
   <HEAD>
      <META   HTTP-EQUIV="Content-Type"   CONTENT="text/html;
      charset=utf-8">
      <TITLE>Тег TITLE</TITLE>
   </HEAD>
   <BODY>
      <H1>Тег TITLE</H1>
      <P>Тег TITLE служить для зазначення назви Web-
        сторінки. Він ставиться в її секції заголовка</P>
      <H6>Приклад:</H6>
      <P>!HEAD!
        !TITLE!Я — заголовок Web-сторінки!/TITLE!
      !HEAD!</P>
   </BODY>
</HTML>
```

Тут ми помістили короткий опис тегу **<TITLE>** і код прикладу, що має такий вигляд:

```
!HEAD!
    !TITLE!Я — заголовок Web-сторінки!/TITLE!
!HEAD!
```

Замість символів **<** і **>**, які неприпустимі в звичайному тексті, ми поставили знак оклику. Надалі ми дізнаємося, як все- таки вставити в текст неприпустимі символи, і замінимо їх.

Збережемо набраний код у файлі з ім'ям **t_title.htm** і відкриємо його в Браузері.

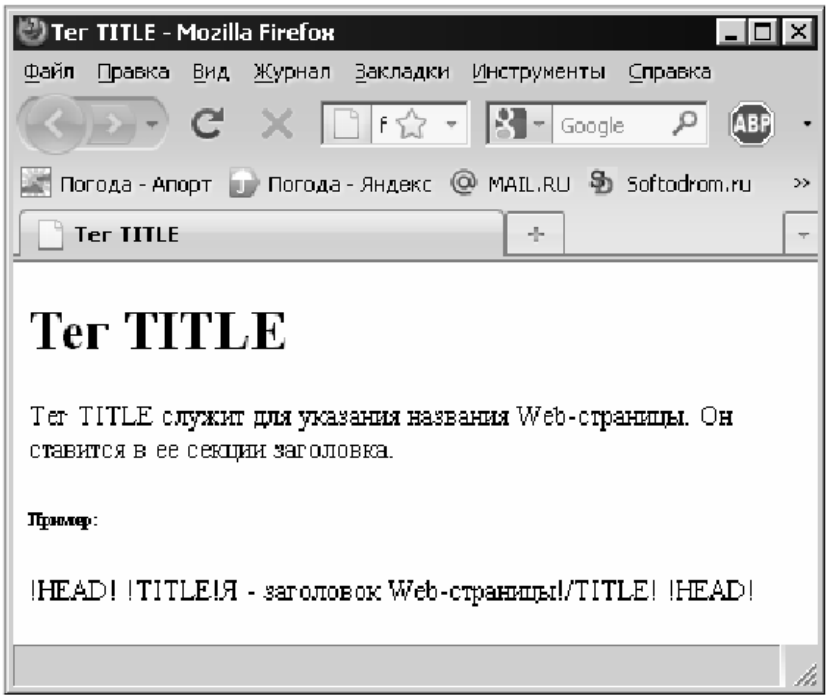


Рис. 2.1. **Web-сторінка t_title.htm в Браузері**

Як бачимо на рис. 2.1, Браузер вивів код прикладу в один рядок і без всяких відступів. Але ж ми розбили його на три рядки і створили відступи за допомогою пробілів, щоб HTML-код краще читався й відразу була видна вкладеність тегів! Що трапилося?

Згадаємо два правила, якими керується Браузер при виведенні тексту блокового елемента і які були наведені в параграфі, присвяченому абзацам. Ці правила говорять, що два або більше пробілів або перенесень рядка, що йдуть один за одним, перетворяться в один пробіл, і перенесення рядка вважається за пробіл. Так Браузер і зробив: перетворив перенесення рядка в пробіли, а послідовні пробіли — в один пробіл. І вивів код прикладу у вигляді звичайного абзацу, який у нього помістився в один рядок.

Спеціально для випадків, коли текст повинен бути виведений саме так, як набраний, зі збереженням усіх пробілів і перенесень рядків, мова HTML

передбачає парний тег <**PRE**>. Виведений текст поміщають усередині цього тегу (лістинг 2.11).

Лістинг 2.11

```
<PRE>Я — текст,
який буде виведений
на Web-сторінку
з усіма
                    відступами
і
перенесеннями рядків</PRE>
```

Такий текст називається текстом *фіксованого формату.* Правила відображення тексту фіксованого формату:

- для виведення використовується моноширинний шрифт (у *моноширинного шрифта* всі символи мають однакову ширину, на відміну від *пропорційних,* у яких ширина символів різна);
- всі пробіли і перенесення рядків зберігаються при виведенні;
- якщо рядок тексту фіксованого формату не поміщається у вікні Браузера по ширині, він в жодному разі не буде переноситися. Через це може виникнути необхідність в горизонтальному прокручуванні Web-сторінки. (що, взагалі, поганий стиль Web-дизайну...);
- можлива наявність HTML-тегів для виділення тексту і гіперпосилань.

Текст фіксованого формату також є блоковим елементом.

Виправимо HTML-код Web-сторінки **t_title.htm** так, щоб приклад виводився у вигляді тексту фіксованого формату (лістинг 2.12).

Лістинг 2.12

```
<!DOCTYPE html>
<HTML>
   <HEAD>
      <META HTTP-EQUIV="Content-Type" CONTENT="text/html;
      charset=utf-8">
      <TITLE>Тег TITLE</TITLE>
   </HEAD>
   <BODY>
      <H1>Тег TITLE</H1>
```

```
      <P>Тег TITLE використовується для зазначення назви
        Web-сторінки. Він ставиться в її секції
        заголовка</P>
      <H6>Приклад:</H6>
      <PRE>!HEAD!
        !TITLE!Я — заголовок Web-сторінки!/TITLE!
      !HEAD!</PRE>
  </BODY>
</HTML>
```

Відкриємо виправлену Web-сторінку в Браузері і переконаємося, що він виводиться правильно (рис. 2.2).

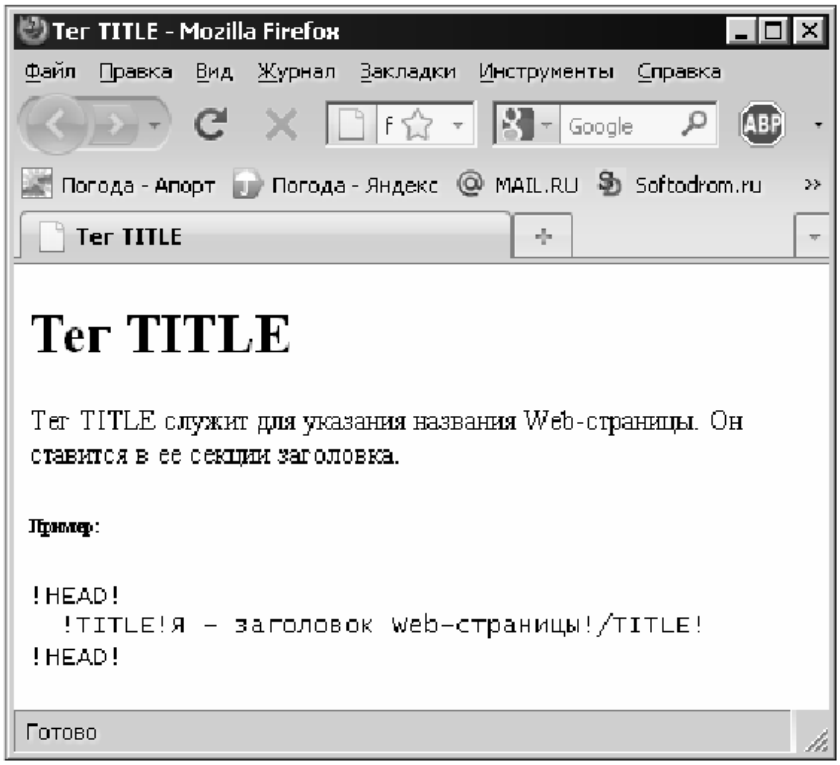


Рис. 2.2. **Виправлена Web-сторінка t_title.htm у Браузері. Код прикладу оформлений як текст фіксованого формату**

Як правило, текст фіксованого формату використовується для виведення вхідних текстів програм і швидкої публікації в Мережі документів, набраних звичайним текстом.

Горизонтальні лінії

Для виділення можна зробити горизонтальну лінію, яка створюється за допомогою одинарного тегу <**HR**>:

```
<P>Я буду відділений від наступного абзацу горизонтальною лінією</P> <HR>
<P>Я відділений від попереднього абзацу горизонтальною лінією</P>
```

Горизонтальна лінія HTML розтягується по горизонталі на всю ширину Web-сторінки, має один-два пиксела в товщину і опуклий або втиснений вигляд (конкретні параметри залежать від Браузера). Вона застосовується для відділення однієї частини вмісту Web-сторінки від іншою і просто "для краси". Однак потрібно пам'ятати, що занадто велике число горизонтальних ліній — дурний тон Web-дизайну.

Внесемо в Html-Код сторінки **index.htm** необхідні виправлення (лістинг 2.13).

Лістинг 2.13

```
 <P>Вітаємо на нашому Web-сайті всіх, хто займається Web-
  дизайном! Тут ви зможете знайти інформацію про всі
  інтернет-технології, що застосовуються при створенні
  Web-сторінок. А саме, про мови HTML і CSS.</P>
<HR>
<P> Вікіпедія визначає термін HTML так: </P>
.....
<HR>
<P>HTML дозволяє формувати на Web-сторінках наступні
  елементи:</P>
```

Результат показано на рис. 2.3.

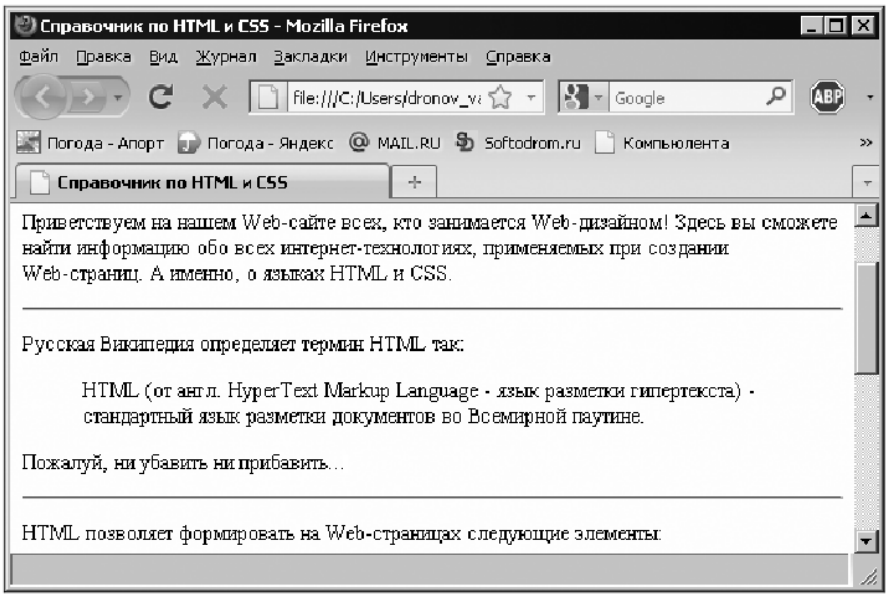


Рис. 2.3. **Web-сторінка index.htm з горизонтальними лініями**

Адреси

Часто на Web-сторінках вказують контактні дані їхніх творців (поштові і електронні адреси, телефони, факси та ін.). Для цього HTML передбачає особливий тег **<ADDRESS>.** Він поводиться так само, як тег абзацу **<P>,** але його вміст виводиться курсивом:

```
<ADDRESS>Я — адреса: поштова, електронна, телефони і факси</ADDRESS>
```

Дата

Для того, щоб додати дату є спеціальний елемент HTML5. Елемент **<TIME>** дозволяє оголосити дату в машинному форматі, а також додати читабельний текст із вказівкою дати і часу.

Лістинг 2.14. Відображення дати за допомогою елемента **<TIME>**

```
<ARTICLE>
    <HEADER>
        <H1>ЗАГОЛОВОК СТАТТІ L</H1>
        <TIME DATETIME="2011-10-12" PUBDATE>опубліковано
        10.12.2011</TIME>
    </HEADER>
    Це текст статті
</ARTICLE>
```

Значення атрибута **DATETIME** є часовою позначкою в машинному форматі, яке повинно відповідати наступному шаблону: 2011-10-12T12:10:45. Ми також додали атрибут **PUBDATE**, який лише повідомляє, що значення атрибута **DATETIME** визначає дату публікації.

Коментарі

Розглянемо одну дуже важливу можливість HTML, яка, хоч і не стосується прямо Web-дизайну, але сильно допоможе забудькуватим Web-дизайнерам.

Коментар — це фрагмент HTML-коду, який не виводиться на Web-сторінку і взагалі не обробляється Браузером. Він служить для того, щоб Web-дизайнер зміг залишити текстові замітки для себе або своїх колег.

Текст коментаря поміщають між відкриваючим тегом **<!--** і закриваючим тегом **-->** і обов'язково відокремлюють від цих тегів пробілами. Як бачимо, теги коментаря не вкладаються в основне правило HTML: закриваючий тег повинен мати те ж ім'я, що і відкриваючий. Відкриваючий і закриваючий теги коментаря — різні.

Приклад:

```
<!-- Я — коментар. Мене не видно на Web-сторінці. -->
```

Ми можемо створити в HTML-коді наших Web-сторінок коментарі, що вказують, що нам слід доробити. Хоча б просто для практики:

```
<!-- Виділити важливі фрагменти тексту і доробити код прикладів. -->
 <!-- Створити Web-сторінки, присвячені іншим тегам HTML. -->
```

КОНТРОЛЬНІ ПИТАННЯ

1. Що таке структурування тексту? Із чого воно полягає?
2. Як розбити текст на абзаци?
3. Як створити заголовки різного рівня?
4. Які бувають типи списків? Як їх створити?
5. Що таке «Текст фіксованого формату»? Як його вставити в документ і як він відобразиться?
6. Як вставити в документ цитату і як вона відобразиться?
7. Як вставити в текст горизонтальну лінію?
8. Як вставити в документ коментар і як він відобразиться?
9. Як вставити в документ адресу і як вона відобразиться?
10. Як вставити в документ дату і як вона відобразиться?

ТЕМА 3. ОФОРМЛЕННЯ ТЕКСТУ

***Основна мета*:** навчитися оформляти текст: виділяти фрагменти абзаців і заголовків, щоб привернути до них увагу відвідувачів, а також. вставляти в текст неприпустимі символи, зокрема, **<** і **>**.

ОФОРМЛЕННЯ ТЕКСТУ

Оформлення тексту

Ми навчилися структурувати текст Web-сторінки, розбиваючи його на логічні "шматки". Також ми довідалися про коментарі HTML, які дозволяють створювати невеликі замітки прямо в коді Web-сторінки. І створили дві Web-сторінки Web-сайту.

В цьому параграфі ми продовжимо працювати з текстом. Ми навчимося оформляти його, виділяючи окремі фрагменти тексту, щоб привернути до них увагу відвідувача. А ще ми навчимося вставляти в текст неприпустимі символи.

Виділення фрагментів тексту

Власне, засоби HTML для оформлення тексту ми вже почали вивчати. Це парні теги **<STRONG>** і **<EM>**, які виділяють свій вміст напівжирним і курсивним шрифтом відповідно.

Однак насправді теги **<STRONG>** і **<EM>** — це щось більше, ніж просте виділення тексту. Вони дають фрагменту текста, що є їхнім вмістом, особливе значення з погляду БРаузеРа. Вони говорять, що даний фрагмент тексту є важливим, і на нього слід звернути увагу відвідувача. Тег **<STRONG>** робить фрагмент тексту дуже важливим, а тег **<EM>** — менш важливим (лістинг 3.1).

Лістинг 3.1

```
<P><STRONG> — дуже важливий текст і тому набраний напівжирним шрифтом!</STRONG> Прочитайте мене в першу чергу! .</P>
 <P><EM>А я — менш важливий текст і набраний курсивом. </EM> Не забудьте прочитати мене, але можете зробити це потім.</P>
```

HTML передбачає для виділення тексту досить багато тегів (табл. 3.1), що мають дві особливості:

- всі вони парні;
- служать для виділення частин тексту блокових елементів (абзаців, заголовків, пунктів списків, тексту фіксованого форматування).

Табл. 3.1. **Теги HTML, призначені для виділення фрагментів тексту**

Тег	Призначення	Відображення Браузером
<ABBR>	Абревіатура	Підкресленим
<ACRONYM>	Абревіатура. Фактично те ж саме, що і тег **<ABBR>**	Підкресленим
<CITE>	Невелика цитата	Курсивом
<CODE>	Фрагмент вхідного коду програми	Моношириним шрифтом
<DEL>	Текст, видалений з Web-сторінки	Закресленим
<DFN>	Новий термін	Курсивом
<EM>	Менш важливий текст	Курсивом
<INS>	Текст, знову поміщений на Web-Сторінку	Підкресленим
<KBD>	Дані, що вводяться користувачем в яку-небудь програму	МоноширРиним шрифтом
<Q>	Невелика цитата. Фактично те ж саме, що й тег **<CITE>**	Звичайним шрифтом
<SAMP>	Дані, що виведені якою-небудь програмою	Моношириним шрифтом
<STRONG>	Дуже важливий текст	Напівжирним шрифтом
<TT>	Дані, що виведені якою-небудь програмою. Фактично те ж саме, що і тег **<SAMP>**	Моношириним шрифтом
<VAR>	Ім'я змінної у вхідному коді програми	Курсивом
<MARK>	Текст, що має особливе значення при обставинах, що склалися, наприклад, відображення результатів пошуку (підсвечуються слова, що збігаються з рядком пошуку).	Підсвічуванням

Як уже говорилося раніше, всі ці теги служать для виділення фрагментів тексту, що є частиною блокових елементів, скажемо, абзаців (лістинг 3.2). Елементи Web-сторінки, які вони створюють, не є незалежними і не відображаються окремо від них "сусідів", а належать іншим елементам — блоковим. Такі елементи Web-сторінки називаються *вбудованими.*

Лістинг 3.2

```
<P>Теги HTML служать для створення елементів Web-сторінок.
<STRONG> Зберігайте порядок вкладеності тегів!</STRONG></P>
<P>Тег<CODE>P</CODE> служить для створення <DFN>абзацу</DFN> HTML.</P>
<P>Мова <ABBR>HTML</ABBR> служить для створення <INS>вмісту</INS> Web-сторінок</P>
<P>Наберіть у Браузері інтернет-адресу <KBD>http://www.w3.org</KBD></P>
```

Із усіх розглянутих нами тегів найчастіше зустрічаються **<STRONG>** і **<EM>.** Інші теги так і не здобули великої популярності серед Web-дизайнерів.

Для практики давайте відкриємо Web-сторінку **index.htm** і виділимо деякі фрагменти її тексту за допомогою тегів, перерахованих в табл. 3.1 (лістинг 3.3).

Лістинг 3.3

```
<P><Вітаємо на нашім Web-сайті всіх, хто займається Web-дизайном! Тут ви зможете знайти інформацію про всі інтернет-технології, застосовувані при створенні Web-сторінок. А саме, про мови <DFN>HTML</DFN> і <DFN>CSS</DFN></P>
.....
<P><CODE>!DOCTYPE</CODE>, <CODE>BODY</CODE>, <CODE>EM</CODE>, <CODE>HEAD</CODE>, <CODE>HTML</CODE>, <CODE>META</CODE>, <CODE>P</CODE>, <CODE>STRONG</CODE>, <CODE>TITLE</CODE></P>
```

Виведення додаткової інформації,дрібним шрифтом

Нова специфіка HTML проявляється особливо наочно на таких елементах, як **<SMALL>.** Раніше даний елемент було призначено для

виведення тексту шрифтом меншого розміру. Це ключове слово відносилося тільки до розміру тексту, без врахування його смислу. В HTML5 в елемента **<SMALL>** з'явилося нове призначення — виведення додаткової інформації, яка звичайно пишеться дрібним шрифтом, наприклад юридична інформація, відмова від відповідальності і т.п. (лістинг 3.4).

Лістинг 3.4. Юридична інформація, оформлена за допомогою елемента **<SMALL>**

```
<SMALL>Copyright &copy; 2811 Minkbooks</SMALL>
```

Розрив рядків

Давайте помістимо відомості про авторські права в самий низ Web-сторінок за допомогою тегу **<ADDRESS>:**

```
<ADDRESS>Всі права захищені. Студенти, 2019 рік.</ADDRESS>
```

Звичайне попередження про те, що авторські права захищені, і ім'я або назва розроблювача розносять на різні рядки. Можна, звичайно, використовувати два теги **<ADDRESS>:** один — для попередження, а іншій— для імені розроблювача. Але тоді рядки будуть розділені досить великою відстанню. (Тег **<ADDRESS>** поводиться як абзац, тобто відділяється від сусідніх абзаців відступом.) А це буде виглядати некрасиво.

Вихід — додати *розрив рядків* HTML. Він виконує безумовне перенесення рядка, в якому він присутній, в тому місці, де проставлений. Розрив рядка визначається одиночним тегом **
:**

```
<P> Цей абзац буде розірваний на два рядки в цьому<BR>місці</P>
```

Розрив рядка також належить до вбудованих елементів Web-сторінки.

Давайте вставимо розрив рядка в текст відомості про авторські права, між крапкою в першому реченні та початком другого речення. Пробіл між ними можна прибрати — він там зовсім не потрібний:

```
<ADDRESS>Всі права захищені.<BR>Студенти, 2019 рік</ADDRESS>
```

Відкриємо виправлену Web-сторінку в Браузері та подивимося на результат. Бачимо, що текст відомостей про авторські права розділений на рядки в тому самому місці, куди ми вставили тег **
.**

Розрив рядків слід застосовувати ощадливо, тільки в тих випадках, коли потрібно розділити один абзац на декілька логічно зв'язаних частин. Раніше недосвідчені Web-дизайнери з його допомогою часто розбивали текст на абзаци, але зараз це дурний тон Web-дизайну.

Вставка неприпустимих символів. Літерали

Давайте відкриємо Web-сторінку **t_title.htm** і подивимося на код наведеного там приклада використання тегу **<TITLE>.** Чого там не вистачає? Символів **<** і **>**, за допомогою яких і створюється тег HTML. Ці символи є неприпустимими і не повинні зустрічатися в звичайному тексті. Ми замінили їх на знак оклику.

Так чи є спосіб все-таки помістити у звичайний текст неприпустимі символи? Творці HTML вирішили, що, якщо вже безпосереднє ці символи вставити в текст неможливо, то їх потрібно замінити на особливу послідовність символів, що називається *літералом*. Зустрівши літерал, Браузер "зрозуміє", що тут повинен бути присутнім відповідний неприпустимий символ, і виведе його на Web-сторінку.

Літерали HTML починаються із символу **&** і закінчуються символом **;** (крапка з комою). Між ними міститься певна послідовність букв. Так, символ < визначається літералом **<**, а символ > — літералом **>**.

Відразу ж виправимо код прикладу (лістинг 3.5).

Лістинг 3.5

```
<!DOCTYPE html>
<HTML>
   <HEAD>
      <META HTTP-EQUIV="Content-Type" CONTENT="text/html;
      charset=utf-8">
      <TITLE>Тег TITLE</TITLE>
   </HEAD>
   <BODY>
      <H1>Тег TITLE</H1>
      <P>Тег TITLE служить для вказівки назви Web-
        сторінки. Він ставиться в її секції заголовка</P>
      <H6>Приклад:</H6>
```

```
      <PRE>&lt;HEAD&gt;
        &lt;TITLE&gt;     Я     —     заголовок     Web-
        сторінки>&lt;/TITLE&gt;
       &lt;HEAD&gt;</PRE>
   </BODY>
</HTML>
```

Відкриємо виправлену Web-сторінку в Браузері. От тепер теги в прикладі відображаються з усіма призначеними символами **<** і **>**!

Літералів в HTML досить багато. Самі часто застосовувані з них наведено в табл. 3.2.

Табл.3.2. **Деякі літерали мови HTML**

Неприпустимий символ	Літерал HTML
— (довге тире)	`&mdash;`
- (коротке тире)	`&ndash;`
"	`&quot;`
&	`&`
<	`&lt ;`
>	`>`
©	`&copy;`
®	`&reg;`
Ліві подвійні лапки	`&ldquo;`
Ліві кутові лапки	`&laquo;`
Лівий апостроф	`&lsquo;`
Три крапки	`&hellip;`
Нерозривний пробіл	` `
Праві подвійні лапки	`&rdquo;`
Праві кутові лапки	`&raquo;`
Правий апостроф	`&rsquo;`
Символ євро	`&euro;`

Серед наведених в табл. 3.2 літералів і позначуваних ними неприпустимих символів особливо виділяється один. Це *нерозривний пробіл*, позначуваний літералом **` `**. По цьому пробілу Браузер ніколи не буде виконувати перенесення рядків.

Нерозривний пробіл необхідний, якщо в якомусь місці речення перенесення рядків ніколи не повинно виконуватися. Так, правила правопису

не допускають перенесення рядків перед довгим тире. Тому дуже рекомендується відокремлювати довге тире від попереднього слова нерозривним пробілом:

```
<P>Нерозривний пробіл &mdash; дуже важливий літерал<P>
```

Тут літерал ** ** створює нерозривний пробіл, а літерал **—** — довге тире.

До речі, ми можемо у відомостях про авторські права вставити символ ©:

```
<ADDRESS>Всі права захищені<BR>&copy; Студенти, 2019 рік</ADDRESS>
```

HTML також дозволяє вставити в текст будь-який символ, підтримуваний кодуванням Unicode, просто вказавши його код. Для цього передбачений літерал виду

&#<десятковий код символу>;.

Довідатися код потрібного символу допоможе утиліта *Таблиця символів*, що поставляється в складі Windows. Давайте запустимо її та подивимося на її вікно (рис. 3.1).

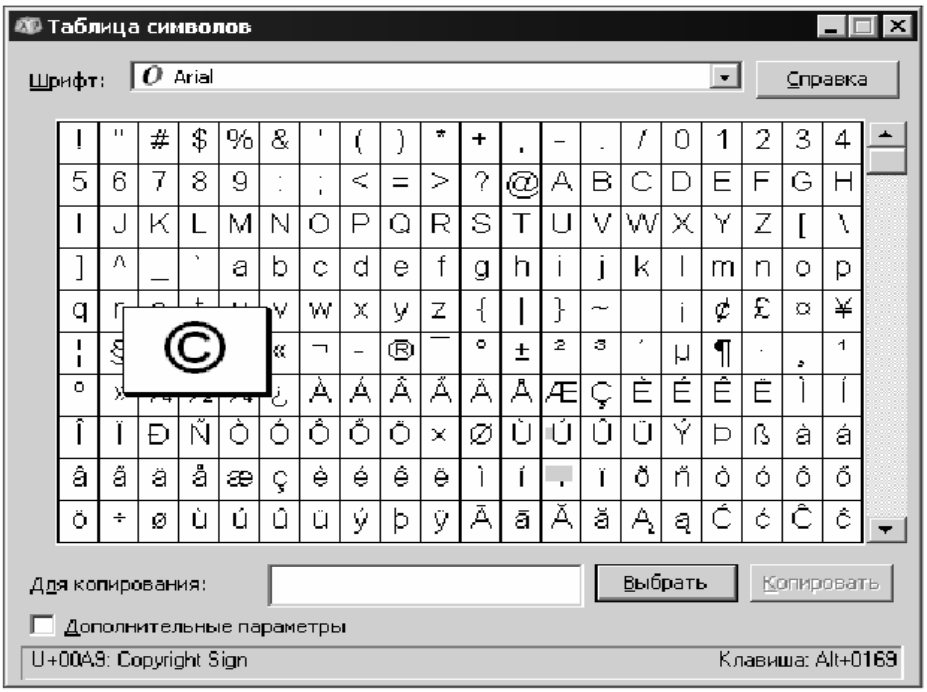


Рис. 3.1. **Вікно утиліти *Таблиця символів* (вибраний символ ©)**

У великому списку символів, що займає майже все вікно цієї утиліти, виберемо потрібний нам символ. Після цього подивимося на рядок стану,

розташований уздовж нижнього краю вікна. В правій його частині знаходиться напис вигляду **Клавіша: Alt+<*десятковий код символу*>.** Ось цей код нам і потрібен/

Примітка. Напис **Клавіша: Alt+<*десятковий код символу*>** з'являється в рядку стану вікна **Таблиця символів** тільки при виборі символів, які не можна ввести безпосередньо із клавіатури.

Так, ми можемо вставити у відомості про авторські права символ ©, використавши літерал **`&#0169;`**, де **`0169`** — десятковий код даного символу (рисунок 3.1):

```
<ADDRESS>Всі права захищені<BR>&#0169;Студенти, 2019 рік</ADDRESS>
```

КОНТРОЛЬНІ ПИТАННЯ

1. Що можна віднести до оформлення тексту?
2. Як можна виділити фрагменти тексту?
3. Назвіть основні теги HTML, призначені для виділення фрагментів тексту.
4. Які теги найчастіше використовуються для оформлення тексту?
5. Як організувати розриви рядків?
6. Чим розрив рядків в абзаці відрізняється від оформлення тексту за допомогою абзаців?
7. Виведення додаткової інформації дрібним шрифтом.
8. Для чого використовується контейнер **`<SMALL>`**…**`</SMALL>`**?
9. Вставка неприпустимих символів.
10. Літерали. Наведіть приклади часто вживаних літералів.

ТЕМА 4. СТВОРЕННЯ ТАБЛИЦЬ

***Основна мета*:** навчитися створювати таблиці, у тому числі таблиці складної структури та вставляти їх у текст

СТВОРЕННЯ ТАБЛИЦЬ

Таблиці

Крім тексту і картинок таблиці часто зустрічаються в друкованих виданнях.. Таблиці — кращий спосіб умістити множину відомостей на обмеженій площі сторінки.

HTML надає засоби для створення таблиць.

Створення таблиць HTML

Таблиці HTML створюються в чотири етапи.

На першому етапі в HTML-коді за допомогою парного тегу **<TABLE>** формують саму таблицю:

```
<TABLE>
</TABLE>
```

Таблиця HTML являє собою блоковий елемент Web-сторінки. Це значить, що вона розміщається окремо від усіх інших блокових елементів: абзаців, заголовків, великих цитат, аудіо- і відеороликів. Так що вставити таблицю в абзац ми не зможемо.

На другому етапі формують рядки таблиці. Для цього засосовуються парні теги **<TR>;** кожний такий тег створює окремий рядок. Теги **<TR>** поміщають всередину тегу **<TABLE>** (лістинг 4.1).

Лістинг 4.1

```
<TABLE>
    <TR>
    </TR>
    <TR>
    </TR>
    <TR>
    </TR>
</TABLE>
```

На третьому етапі створюють комірки таблиці, для чого використовують парні теги **<TD>** і **<TH>.** Тег **<TD>** створює звичайну комірку, а тег **<TH>** — *комірку заголовка,* у якій буде розміщатися "шапка" відповідної колонки таблиці. Теги **<TD>** і **<TH>** поміщають в теги **<TR>**, що створюють рядки таблиці, в яких повинні знаходитися ці комірки (лістинг 4.2).

Лістинг 4.2

```
<TABLE>
    <TR>
       <TH></TH>
       <TH></TH>
       <TH></TH>
    </TR>
    <TR>
       <TD></TD>
       <TD></TD>
       <TD></TD>
    </TR>
    <TR>
       <TD></TD>
       <TD></TD>
       <TD></TD>
    </TR>
</TABLE>
```

На четвертому, останньому, етапі вказують вміст комірок, який поміщають у відповідні теги **<TD>** і **<TH>** (лістинг 4.3).

Лістинг 4.3

```
<TABLE>
    <TR>
       <TH> Колонка 1</TH>
       <TH> Колонка 2</TH>
       <TH> Колонка 3</TH>
    </TR>
    <TR>
       <TD> Комірка 1.1</TD>
       <TD> Комірка 1.2</TD>
       <TD> Комірка 1.3</TD>
    </TR>
    <TR>
       <TD> Комірка 2.1</TD>
       <TD> Комірка 2.2</TD>
       <TD> Комірка 2.3</TD>
    </TR>
</TABLE>
```

Якщо нам потрібно помістити в комірку таблиці простий текст, ми можемо просто вставити його у відповідний тег **<TD>** або **<TH>** (як показано в лістингу 4.3). При цьому поміщати його в теги, що створюють блокові елементи, необов'язково.

Якщо нам буде потрібно якось оформити вміст комірок, ми застосуємо вивчені раніше теги. Наприклад, ми можемо додати номерам комірок особливу важливість, скориставшись тегом **<EM>;** в результаті вони будуть виведені курсивом (лістинг 4.4).

Лістинг 4.4

```
<TABLE>
    ……..
    <TR>
       <TD> Комірка <EM>1.1</EM></TD>
       <TD> Комірка <EM>1.2</EM></TD>
       <TD> Комірка <EM>1.3</EM></TD>
    </TR>
    ……..
</TABLE>
```

Але часто буває необхідно помістити в комірку таблиці великий текст, що іноді складається з декількох абзаців. В такому випадку застосовуються теги, що створюють блокові елементи сторінки. Теги **<TD>** і **<TH>** це дозволяють (лістинг 4.5).

Лістинг 4.5

```
<TD>
    <H4>Це великий текст</H4>
    <P>Цей початок великого тексту, що представляє собою
    вміст комірки таблиці</P>
    <P>Це продовження великого тексту, що представляє
    собою вміст комірки таблиці</P>
    <P><IMG SRC="picture.jpg"> ALT=”Ілюстрація до
    великого тексту"></P>
    <P>А це <STRONG>довгоочікуване закінчення</STRONG>
    великого тексту</P>
</TD>
```

Даний HTML-код поміщає в комірку таблиці заголовок і чотири абзаци. Причому один із цих абзаців містить графічне зображення, а частина іншого позначена як дуже важлива (і буде набрана напівжирним шрифтом).

HTML-код, що створює таблиці, може здатися трохи громіздким. Але це плата за виняткову гнучкість таблиць HTML. Ми можемо помістити в таблицю будь-який вміст: абзаци, заголовки, зображення, аудіо- і відеоролики та навіть інші таблиці.

Тепер настав час розглянути правила, якими керуються Браузери при виведенні таблиць на екран.

- Таблиця є блоковим елементом Web-сторінки.
- Розміри таблиці і її комірок робляться такими, щоб повністю вмістити їхній вміст.
- Між границями окремих комірок і між границею кожного комірки і її вмістом робиться невеликий відступ.
- Текст комірок заголовка виводиться напівжирним шрифтом і вирівнюється по центру.
- Рамки навколо всієї таблиці і навколо окремих її комірок не рисуються.

Таблиця — усього лише вміст Web-сторінки, а за її виведення "відповідає" представлення. Якщо нам потрібно, наприклад, вивести навколо таблиці рамку, ми зможемо створити відповідне представлення.

І ще кілька правил, відповідно до яких створюється HTML-код таблиць. Якщо їх порушити, Браузер відобразить таблицю некоректно або не виведе її взагалі.

- Тег **<TR>** може знаходитися тільки всередині тегу **<TABLE>.** Будь-який інший вміст тегу **<TABLE>** (крім заголовка і секцій таблиці, розмова про яких піде далі) буде зігноровано.
- Теги **<TD>** і **<TH>** можуть знаходитися тільки усередині тегу **<TR>.** Будь-який інший вміст тегу **<TR>** буде зігноровано.
- Вміст таблиці може знаходитися тільки в тегах **<TD>** і **<TH>.**
- Комірки таблиці повинні мати хоч якісь вміст, інакше Браузер може їх взагалі не відобразити. Якщо ж якась комірка повинна бути порожньою, в неї слід помістити нерозривний пробіл (HTML-літерал ** **).

Всеі, з теорією покінчене. Настала пора практики. Давайте помістимо на Web-сторінку **index.htm** таблицю, що перераховує всі версії мови HTML із вказівкою року виходу. Вставимо її після цитати з Вікіпедії та горизонтальної лінії, що відокремлює її.

Лістинг 4.6 містить фрагмент HTML-коду Web-сторінки **index.htm**, що створює таку таблицю.

Лістинг 4.6

```
<HR>
<P>Список версій HTML:</P>
<TABLE>
    <TR>
        <TH>Версія HTML</TH>
        <TH>Рік виходу</TH>
        <TH>Особливості</TH>
    </TR>
     <TR>
        <TD>1.0</TD>
        <TD>1992</TD>
        <TD>Офіційно не була стандартизована</TD>
    </TR>
     <TR>
        <TD>2.0</TD>
```

```
        <TD>1995</TD>
        <TD>Перша стандартизована версія</TD>
    </TR>
    <TR>
        <TD>3.2</TD>
         <TD>1997</TD>
        <TD>Підтримка таблиць і графіки</TD>
    </TR>
    <TR>
        <TD>4.0</TD>
         <TD>1997</TD>
        <TD>&quot;Очищений&quot;      від      застарілих
        тегів</TD>
    </TR>
    <TR>
        <TD>4.01</TD>
         <TD>1999</TD>
        <TD>В основному виправлення помилок</TD>
     </TR>
     <TR>
        <TD>5.0</TD>
        <TD>2014</TD>
        <TD>Корінні зміни</TD>
     </TR>
</TABLE>
<P>HTML дозволяє формувати на Web-Сторінках наступні
елементи1:</P>
```

Збережемо Web-сторінку та відкриємо в Браузері (рис. 4.1)

Версия HTML	Год выхода	Особенности
1.0	1992	Официально не была стандартизована
2.0	1995	Первая стандартизованная версия
3.2	1997	Поддержка таблиц и графики
4.0	1997	"Очищен" от устаревших тегов
4.01	1999	В основном, исправление ошибок
5.0	2014	Коренные изменения

Рис. 4.1. **Таблиця — список версій HTML на Web-сторінці index.htm**

Як бачимо, наша таблиця не дуже презентабельна. Браузер зробив її стислої, без рамок, з маленькими відступами між коміркими. Ну так ця справа поправна — надалі ми зможемо оформити таблицю (і інші елементи Web-сторінки) як побажаємо.

Заголовок і секції таблиці

Тепер розглянемо додаткові можливості HTML по створенню таблиць. На практиці вони застосовуються нечасто, але іноді можуть пригодитися.

Насамперед, за допомогою парного тегу **<CAPTION>** ми можемо дати таблиці заголовок. Текст заголовка поміщають усередину цього тегу, який, в свою чергу, знаходиться всередині тегу **<TABLE>** (лістинг 4.7).

Лістинг 4.7

```
<TABLE>
    < CAPTION>Это таблиця</CAPTION>
     <TR>
          <TH>Колонка 1</TH>
          <TH>Колонка 2</TH>
          <TH>Колонка 3</TH>
    </TR>
.....
</TABLE>
```

Заголовок таблиці виводиться над нею, а текст його вирівнюється по центру таблиці. При бажанні ми можемо його якось оформити, скориставшись такими тегами:

```
<CAPTION><STRONG>Это таблиця <STRONG></CAPTION>
```

Звичайно тег **<CAPTION>** поміщається відразу після відкриваючого тегу **<TABLE>** — так логічніше. Але не має значення, у якому місці HTML-коду таблиці він присутній — заголовок однаково буде поміщений Браузером над таблицею.

Крім того, ми можемо логічно розбити таблицю HTML на три значущі частини — *секції таблиці*:

- *секцію заголовка,* в якій знаходиться рядок із коміркими заголовка, що формує її "шапку";
- *секцію тіла,* де знаходяться рядки таблиці, що містять основні дані;

- *секцію завершення* з рядками, що формують "піддон" таблиці (звичайно в "піддоні" розташовують підсумкові дані та різні примітки).

Секцію заголовка таблиці формує тег **<THEAD>,** секцію тіла — **<TBODY>,** а секцію завершення — **<TFOOT>.** Усі ці теги парні, поміщаються безпосередньо в тег **<TABLE>** і містять теги **<TR>**, що формують рядки таблиці, які входять у відповідну секцію (лістинг 4.8).

Лістинг 4.8

```
<TABLE>
   <THEAD>
       <TR>
          <TH>Колонка 1</TH>
           <TH>Колонка 2</TH>
          <TH>Колонка 3</TH>
       </TR>
    </THEAD>
    <TBODY>
       <TR>
          <TD>Комірка 1.1</TD>
          <TD>Комірка 1.2</TD>
          <TD>Комірка 1.3</TD>
       </TR>
       <TR>
          <TD>Комірка 2.1</TD>
          <TD>Комірка 2.2</TD>
          <TD>Комірка 2.3</TD>
      </TR>
    </TBODY>
   <TFOOT>
      <TR>
          <TD>Підсумок по комірці 2.1</TD>
          <TD>Підсумок по комірці 2.2</TD>
          <TD>Підсумок по комірці 2.3</TD>
      </TR>
   </TFOOT>
</TABLE>
```

Секції таблиці Браузер ніяк не відображає і не виділяє на Web-сторінці. Вони просто поділяють таблицю на три логічні частини. Однак ми можемо

задати для тегів, що формують секції таблиці, якесь представлення, яка буде управляти їхнім відображенням. Докладніше про це ми довідаємося далі.

Примітка. Тег **<TABLE>** підтримує необов'язковий атрибут **SUMMARY**, за допомогою якого ми можемо задати примітку до таблиці. Ця примітка взагалі не показується на екрані, однак може використовуватися іншими засобами виведення Web-сторінок, наприклад, Браузерами для людей з вадами зору, що читають вміст Web-сторінок. Так чи інакше, примітка до таблиць практично ніколи не задається.

Застосуємо отримані знання до таблиці — списку версій HTML, який ми недавно створили на Web-сторінці **index.htm**. Над цією таблицею ми вставили абзац із текстом "Список версій HTML:", який так і проситься в заголовки. Заодно розділимо таблицю на секції заголовка і тіла. ("Піддона" наша таблиця не має, так що обійдемося без секції завершення.)

У лістингу 4.9 представлений виправлений фрагмент HTML-коду Web-сторінки **index.htm**, який створює цю таблицю.

Лістинг 4.9

```
<HR>
<TABLE>
  <CAPTION>Список версій HTML:</CAPTION>
  <THEAD>
     <TR>
       <TH>Версія HTML</TH>
       <TH>Рік виходу</TH>
       <TH>Особливості</TH>
     </TR>
  </THEAD>
   <TBODY>
      <TR>
       <TD>1.0</TD>
       <TD>1992</TD>
       <TD>Офіційно не була стандартизована</TD>
     </TR>
      <TR>
       <TD>2.0</TD>
       <TD>1995</TD>
       <TD>Перша стандартизована версія</TD>
     </TR>
     <TR>
```

```
        <TD>3.2</TD>
         <TD>1997</TD>
        <TD>Підтримка таблиць і графіки</TD>
      </TR>
      <TR>
        <TD>4.0</TD>
         <TD>1997</TD>
        <TD>&quot;Очищений&quot; від застарілих
        тегів</TD>
      </TR>
      <TR>
        <TD>4.01</TD>
         <TD>1999</TD>
        <TD>В основному виправлення помилок</TD>
       </TR>
       <TR>
        <TD>5.0</TD>
        <TD>2014</TD>
        <TD>Корінні зміни</TD>
       </TR>
  </TBODY>
</TABLE>
<P>HTML дозволяє формувати на Web-сторінках наступні
елементи:</P>
```

Збережемо виправлену Web-сторінку і відкриємо її в Браузері. І відразу побачимо, що текст "Список версій HTML" тепер вирівняний по центру таблиці. Це й не дивно — адже ми перетворили його в заголовок таблиці. Сама ж таблиця нітрохи не змінилася (що теж зрозуміло — адже її секції Браузер ніяк не виділяє).

Об'єднання комірок таблиць

Залишилося поговорити про одну цікаву особливість мови HTML. Це так зване *об'єднання комірок* таблиць. Найкраще розглянути приклад — просту таблицю, HTML-код якої наведено в лістингу 4.10.

Лістинг 4.10

```
<TABLE>
    <TR>
```

```
        <TD>1</TD>
        <TD>2</TD>
        <TD>3</TD>
         <TD>4</TD>
         <TD>5</TD>
     </TR>
    <TR>
        <TD>6</TD>
        <TD>7</TD>
        <TD>8</TD>
        <TD>9</TD>
        <TD>10</TD>
     </TR>
    <TR>
        <TD>11</TD>
        <TD>12</TD>
        <TD>13</TD>
         <TD>14</TD>
        <TD>15</TD>
    </TR> <TR>
        <TD>16</TD>
        <TD>17</TD>
        <TD>18</TD>
        <TD>19</TD>
        <TD>20</TD>
    </TR>
</TABLE>
```

Це звичайна таблиця, комірки якої пронумеровані — так нам буде простіше надалі. На рис. 4.2 показаний її вид у вікні Браузера.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

Рис. 4.2. **Початкова таблиця, комірки якої будуть об'єднані**

А тепер розглянемо таблицю на рис. 4.3.

<table>
<tr><td rowspan="2">1+6</td><td colspan="2">2+3</td><td colspan="2">4+5</td></tr>
<tr><td>7</td><td>8</td><td>9</td><td>10</td></tr>
<tr><td>11</td><td colspan="4">12+13+14+15</td></tr>
<tr><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td></tr>
</table>

Рис. 4.3. **Таблиця, показана на рис. 4.2, після об'єднання деяких комірок (об'єднані комірки позначені додаванням їх номерів)**

Тут виконане об'єднання деяких комірок. Бачимо, що об'єднані комірки немов злилися в одну. Як це зробити?

Спеціально для цього теги **<TD>** і **<TH>** підтримують два досить важливі необов'язкові атрибути. Перший — **COLSPAN** — об’єднує комірки по горизонталі, другий — **ROWSPAN** — по вертикалі.

Щоб ***об'єднати кілька комірок по горизонталі*** в одну, потрібно виконати наступні кроки.

1. Знайти в коді HTML тег **<TD> (<TH>),** що відповідає першої з поєднуваних комірок (якщо лічити комірки зліва направо).
2. Вписати в нього атрибут **COLSPAN** і присвоїти йому кількість об’єднуваних комірок, лічачи і найпершу з них.
3. Видалити теги **<TD> (<TH>),** що створюють інші об’єднувані комірки даного рядка.

Давайте об'єднаємо комірки 2 і 3 таблиці (див. лістинг 4.8). Виправлений фрагмент коду, що створює перший рядок цієї таблиці, наведено в лістингу 4.11.

Лістинг 4.11

```
<TR>
   <TD>1</TD>
   <TD COLSPAN="2">2 + 3</TD>
    <TD>4</TD>
    <TD>5</TD>
 </TR>
```

Точно так само створимо об'єднані комірки 4 + 5 і 12 + 13 + 14 + 15.

Об'єднати комірки по вертикалі трохи складніше. От кроки, які потрібно для цього виконати.

1. Знайти в коді HTML рядок (тег **<TR>**), в якому знаходиться перша з об'єднуваних комірок (якщо лічити рядки зверху вниз).
2. Знайти в коді цього рядка тег **<TD> (<TH>),** що відповідає першої з об'єднуваних комірок.
3. Вписати в нього атрибут **ROWSPAN** і присвоїти йому кількість об'єднуваних комірок, лічачи і найпершу з них.
4. Переглянути наступні рядки та вилучити з них теги **<TD> (<TH>),** що створюють інші об'єднувані комірки.

Нам залишилося об'єднати комірки 1 і 6 нашої таблиці. Лістинг 4.12 містить виправлений фрагмент її HTML-коду (виправлення стосуються першого і другого рядків).

Лістинг 4.12

```
<TR>
    <TD ROWSPAN=”2”>1+6</TD>
    <TD COLSPAN=”2”>2+3</TD>
    <TD COLSPAN=”2”>4+5</TD>
</TR>
<TR>
    <TD>7</TD>
    <TD>8</TD>
     <TD>9</TD>
     <TD>10</TD>
</TR>
```

Звернемо увагу, що ми видалили із другого рядка тег **<TD>**, що створює шосту комірку, оскільки вона об'єдналася з першою коміркою.

КОНТРОЛЬНІ ПИТАННЯ

11. Як розбити текст на абзаци?
12. Як створити заголовки різного рівня?
1. Таблиці. Як створювати таблиці?

2. Назвіть етапи створення таблиці.
3. Правила, згідно з якими створюється HTML-код таблиць.
4. Правила, якими керуються Браузери при виведенні таблиць на екран.
5. Що таке заголовок і секції таблиці? Як їх створювати?
6. Як створюються рядки і комірки таблиці?
7. В чому відмінність парних тегів **<TD>** і **<TH>?** Для чого вони використовуються?
8. Чи можна оформляти текст в комірках? Якщо да, то за допомогою яких тегів це можна зробити?
9. Як об'єднати комірки в таблиці по горизонталі?
10. Як об'єднати комірки в таблиці по вертикалі?

ТЕМА 5. ГРАФІКА І МУЛЬТИМЕДІА

***Основна мета*:** навчитися поміщати на Web-сторінки графічні зображення, аудіо- і відеоролики; почати вивчати нові можливості HTML5.

Web-сторінки можуть містити також графіку і мультимедійні дані (аудіо- і відеоролики). Уміло застосовані, вони здатні значно оживити Web-сайт. І це не говорячи вже про випадки, коли без графіки і мультимедіа просто не обійтися. Дійсно, Web-сайт, присвячений музиці, обов'язково містить на Web-сторінках музичні композиції.

Зараз ми будемо працювати із графічними зображеннями та мультимедійними даними. І почнемо вивчення нових можливостей HTML 5, які значно спрощують роботу.

ВБУДОВАНІ ЕЛЕМЕНТИ WEB-СТОРІНОК

Прямо в HTML-коді неможливо описувати графічне зображення, аудіо- або відеокліп, оскільки графіка та мультимедійні дані мають принципово іншу природу, ніж текст. Через це об'єднати їх в одному файлі неможливо.

Розроблювачі HTML знайшли оригінальний вихід з положення. Насамперед, графічні зображення та мультимедійні дані зберігаються в

окремих файлах. А в HTML-коді Web-сторінок за допомогою особливих тегів прописують посилання на ці файли, фактично — їх інтернет-адреси. Зустрівши такий тег-посилання, Браузер запитує у Web-сервера відповідний файл із зображенням, аудіо- або відеороликом і виводить його на Web-сторінку в те місце, де зустрівся даний тег.

Графічні зображення, аудіо- і відеоролики та взагалі будь-які дані, що зберігаються в окремих від Web-сторінки файлах, називаються *вбудованими* елементами Web-сторінок.

Примітка Раніше ми говорили, що Web-сторінка може зберігатися в декількох файлах. Web-сторінка із вбудованими елементами — тому приклад.

Графіка

Графіка на Web-сторінках з'явилася досить давно. Призначений для цього тег з'явився ще у версії 3.2 мови HTML, яка вийшла в 1997 році. Як уже говорилося, графічні зображення — це вбудовані елементи Web-сторінок. Це значить, що вони зберігаються в окремих від самої Web-сторінки файлах.

Формати інтернет-графіки

На даний момент існує кілька десятків форматів зберігання графіки у файлах. Але Браузери підтримують далеко не всі. В WWW зараз використовуються всього три формати, які ми далі розглянемо.

Потрібно відзначити, що всі три формати підтримують стиснення графічної інформації. Завдяки стисненню розмір графічного файла сильно зменшується, і тому він передається по мережі швидше, чим незжатий файл.

Формат GIF

Формат *GIF* (Graphics Interchange Format, формат обміну графікою) — старожил серед "мережевних" форматів графіки. Він був розроблений ще в 1987 році та модернізований в 1989 році. На даний момент він вважається застарілим, але усе ще широко поширений.

Переваг у нього досить багато. По-перше, GIF дозволяє задати для зображення "прозорий" колір; зафарбовані цим кольором області зображення стануть свого роду "дірками", крізь які будуть "просвічувати" фон батьківського елемента. По-друге, в одному GIF-файлі можна зберігати відразу кілька зображень, фактично — справжній фільм *(анімований GIF)*. По-третє, через особливості застосовуваного в ньому стиснення він відмінно

підходить для зберігання штрихових зображень (карт, схем, малюнків олівцем та ін.).

Недолік у форматі GIF усього один — він зовсім не годиться для зберігання півтонових зображень (фотографій, картин і т.п.). Це обумовлено тим, що GIF-зображення можуть включати всього 256 кольорів, і втратами якості при стисненні.

GIF використовується для зберігання елементів оформлення Web-сторінок (усіляких лінійок, маркерів списків і т.п.) і штрихових зображень.

Формат JPEG

Формат *JPEG* (Joint Photographic Experts Group, Об'єднана група експертів по фотографії) був розроблений в 1993 році спеціально для зберігання півтонових зображень. Для цього активно застосовується дотепер.

JPEG, на відміну від GIF, не обмежує кількість кольорів у зображення, а реалізоване в ньому стиснення найкраще підходить для півтонових зображень. Однак воно погано справляється із штриховою графікою, не підтримує "прозорий" колір і анімацію.

Формат PNG

Формат *PNG* (Portable Network Graphics, переміщувана мережевна графіка) з'явився на світ в 1996 році. Він розроблявся як заміна застарілому і не дуже зручному GIF, а також, до деякої міри, JPEG. В цей час він послідовно відвойовує "життєвий простір" у GIF.

До переваг формату PNG можна віднести можливість зберігання як штрихових, так і півтонових зображень та підтримку напівпрозорості. Недолік усього один і некритичний — неможливість зберігання анімації.

Залишилося назвати розширення, під якими зберігаються файли того або іншого формату. Файли GIF і PNG мають розширення **gif** і **png**, а файли JPEG — **jpg**, **jpe** або **jpeg**.

Вставка графічних зображень

Додати на Web-сторінку графічне зображення дозволяє одинарний тег **<IMG>.** Браузер помістить зображення в тому місці Web-сторінки, у якому зустрівся тег **<IMG>.**

Обов'язковий атрибут тегу SRC служить для вказівки інтернет-адреси файлу із зображенням.

Приклад:

```
<IMG SRC="image.gif">
```

Цей тег поміщає на Web-сторінку зображення, що зберігається у файлі **image.gif**, який знаходиться в тій же папці, що і файл самої Web-сторінки.

Приклад:

```
<IMG SRC="/images/picture.jpg">
```

Даний тег поміщає на Web-сторінку зображення, що зберігається у файлі **picture.jpg**, який знаходиться в папці **images**, вкладеної в кореневу папку Web- сайту.

Приклад:

```
<IMG SRC="http://www.othersite.ua/book12/author.jpg">
```

А цей тег поміщає на Web-сторінку зображення, що зберігається у файлі з інтернет-адресою **http://www.othersite.ua/book12/author.jpg**, тобто зображення з іншого Web-сайту.

Примітка. Принципи формування інтернет-адрес файлів, застосовувані в WWW, ми докладно розглянемо далі.

Ми вже знаємо про те, що елементи Web-сторінки можуть бути блоковими і вбудованими. Отож, зображення, поміщене на Web-сторінку за допомогою тегу **<IMG>**, — вбудований елемент. Це значить, що він повинен бути частиною блокового елемента, наприклад, абзацу.

Із цього випливають два важливі висновки.

По-перше, ми можемо вставити графічне зображення прямо в абзац:

```
<P>Подивітеься картинку — <IMG SRC="image.gif"></P>
```

По-друге, якщо нам знадобиться відобразити на Web-сторінці окреме, не зв'язане ні з яким абзацом графічне зображення, нам доведеться помістити його в спеціально створений абзац:

```
<P><IMG SRC="/images/picture.jpg"></P>
```

Помістимо зображення значка @ у папку, де зберігаються файли нашого Web-сайту. Впишемо в розділ тегів Web-сторінки **index.htm** тег **<IMG>** і створимо Web-сторінку, що його описує. Це буде третя Web-сторінка нашого Web-сайту. Її HTML-код наведено в лістингу 5.1.

Лістинг 5.1

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
     <META    HTTP-EQUIV="Content-Type"    CONTENT="text/HTML;
     charset=utf-8">
     <TITLE>Тег IMG</TITLE>
  </HEAD>
  <BODY>
     <H1>Тег IMG</H1>
     <P>Тег IMG служить для вставки на Web-сторінку
     графічного зображення</P>
     <H6>Приклад:</H6>
     <PRE>&lt;P&gt;&lt;IMG
     SRC=&quot;image.gif&quot;&gt;&lt;/P&gt;</PRE>
     <H6>Результат:</H6>
     <P><IMG SRC="image.gif"></P>
  </BODY>
</HTML>
```

Збережемо нову Web-сторінку в файлі **t_img.htm** і відразу ж відкриємо в Браузері (рис. 5.1). На цій Web-сторінці ми побачимо код прикладу вигляду

```
<P><IMG SRC="image.gif"></P>
```

а трохи нижче — результат його виконання.

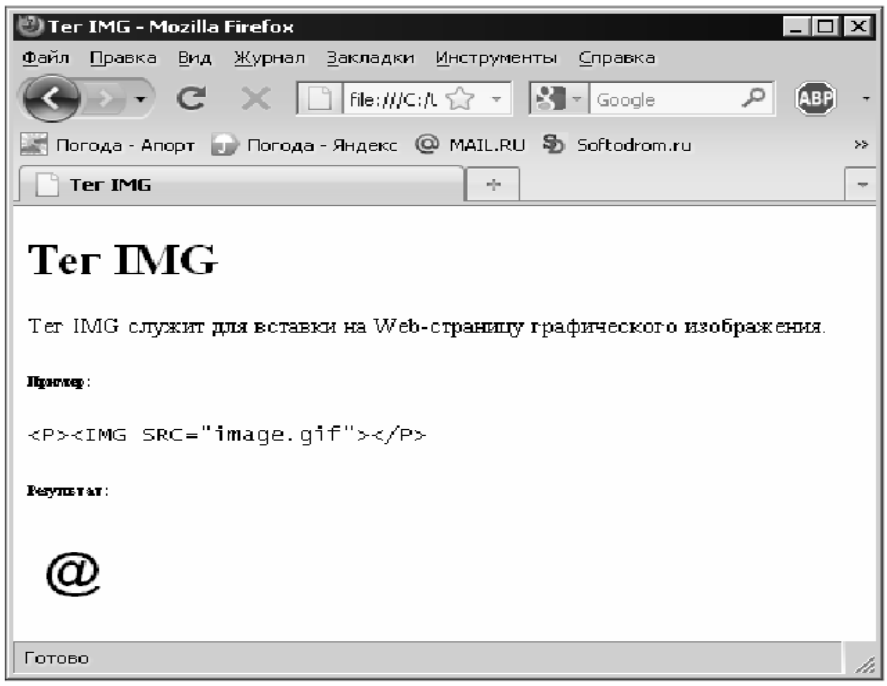


Рис. 5.1. **Web-сторінка t_img.htm у Браузері**

А тепер розглянемо ще один атрибут тегу **<IMG>,** який може нам знадобитися надалі.

Оскільки зображення зберігається в окремому від Web-сторінки файлі, Браузеру потрібно послати Web-серверу ще один запит на його одержання. Web-серверу потрібно знайти цей файл і відправити його Браузеру. Файл повинен завантажитися по мережі.

На все це потрібен час. Якщо зображень на Web-сторінці багато, всі вони великі по розміру, а канал зв'язку повільний, знадобиться значний час. Може трапитися так, що сама Web-сторінка буде успішно завантажена і відображена на екрані, а зображення — ще ні. І Браузер замість незавантаженого ще зображення виведе на екран порожній прямокутник.

Виникають дві проблеми. По-перше, порожні прямокутники замість зображень виглядають некрасиво. По-друге, відвідувач не зможе зрозуміти, що за зображення повинно знаходитися замість того або іншого прямокутника, і чи має сенс чекати закінчення його завантаження.

І якщо з першою проблемою впоратися практично неможливо, то другу ми цілком здатні розв'язати. Для цього тег **<IMG>** підтримує необов'язковий атрибут **ALT,** за допомогою якого вказується так званий *текст заміни.* Він буде відображатися в порожньому прямокутнику, що позначає незавантажене зображення, поки це зображення не завантажиться:

```
<P><IMG SRC="image.gif" ALT="Приклад зображення"></P>
```

Тут ми задали для зображення з Web-сторінки **t_img.htm** текст заміни "Приклад зображення".

Примітка. Гарним тоном Web-дизайну вважається вказівка тексту заміни тільки у значущих зображень. У зображень, що є елементами оформлення Web-сторінки, текст заміни звичайно не вказують.

МУЛЬТИМЕДІА

Мультимедіа — це, в першу чергу, аудіо і відео. Мультимедіа в відношенні до Web-дизайну — це аудіо- і відеоролики, розміщені на Web-сторінках.

До недавнього часу розмістити на Web-сторінці аудіо- або відеоролик можна було тільки за допомогою величезного HTML-коду і додаткових програм. Але зараз, з появою HTML 5 і підтримуючих його Браузерів, буде потрібно всього один тег.

Примітка. Буде розглядатися робота з мультимедіа винятково засобами HTML 5. Застарілі способи, зокрема, тег **<OBJECT>,** не описані.

Формати файлів і формати кодування

Форматів мультимедійних файлів існує не менше, чим форматів файлів графічних. Як і у випадку з інтернет-графікою, Браузери підтримують далеко не всі мультимедійні формати, а тільки деякі.

Але Браузеру мало підтримувати тільки сам формат мультимедійних файлів. Він повинен бути "знайомим" і з форматом кодування записаної в ньому аудіо- і (або) відеоінформації. В світі мультимедіа так — різні файли одного формату можуть зберігати інформацію, закодовану різними форматами. Більше того, аудіо- і відеодоріжки мультимедійного файлу практично завжди кодуються різними форматами.

Практично всі формати кодування мультимедійних даних підтримують їхнє стиснення. Завдяки цьому розмір мультимедійних файлів значно (іноді на кілька порядків) зменшується, що благотворно позначається на швидкості їх передачі по мережі.

Перелічимо та коротко опишемо всі формати мультимедійних файлів, використовувані в Web-дизайні і підтримувані Браузерами.

- *WAV* (Wave, хвиля) — "старожил" серед мультимедійних форматів. Був розроблений Microsoft на самому початку 90-х років минулого століття для зберігання аудіоданих і застосовується для цієї мети дотепер. Файли такого формату мають розширення **wav.**
- *OGG* — більш новий формат. Був розроблений близько десяти років тому некомерційною організацією Xiph.org для зберігання аудіо- і відеоінформації. Файли цього формату мають розширення **ogg** (універсальне розширення), **oga** (аудіофайли) і **ogv** (відеофайли); останні два розширення зустрічаються рідко.
- *MP4* — також "новачок". Був розроблений організацією Motion Picture Expert Group (Експертна група з питань зображення, що рухається; також відома як MPEG) в 1998 році для зберігання аудіо- і відеоданих. Файли цього формату мають розширення **mp4.**
- *Quicktime* — формат дуже старий, він старше навіть WAV. Був розроблений Apple в 1989 році для зберігання аудіо- і відеоданих. Файли такого формату мають розширення **mov.**

Тепер розглянемо формати кодування аудіо і відео, підтримувані сучасними Браузерами.

- *PCM* (Pulse-Coded Modulation, імпульсно-кодова модуляція) — найпростіший і самий старий формат кодування. Він навіть не підтримує стиснення інформації. Служить для кодування аудіоданих.
- *Vorbis* — більш сучасний формат кодування. Був представлений організацією Xiph.org (розроблювачем формату файлу OGG) в 2002 році. Використовується для кодування аудіоданих.
- *AAC* (Advanced Audio Coding, розвинене кодування аудіо) — не дуже новий формат кодування. Був розроблений організацією Motion Picture Expert Group в 1997 році. Застосовується для кодування аудіоданих.
- *Theora* — мабуть, самий "молодий" формат кодування. Він також був розроблений організацією Xiph.org кілька років назад. Використовується для кодування відеоданих.
- *H.264* — теж дуже "молодий". Був представлений організаціями Motion Picture Expert Group і Video Coding Experts Group (Група експертів по кодуванню відео) в 2003 році. Призначений для кодування відеоданих.

Майже всі ці формати є відкритими. Виключення — формат файлів Quicktime, що належить Apple, і формат кодування H.264, захищений більш ніж сотнею патентів.

Залишилося з'ясувати, які комбінації форматів файлів і форматів кодування використовуються в Web-дизайні і які Браузери їх підтримують. Ці дані представлено в табл. 5.1.

Табл. 5.1. **Комбінації формату мультимедійних файлів і форматів кодування аудіо і відео, використовувані в Web-дизайні, і підтримка їх сучасними Браузерами**

Аудіо	Відео	Firefox	Opera	Safari	Chrome
Файли, що містять тільки аудіо					
WAV-PCM		*	*		*
Ogg-vorbis		*	*		*
MOV-AAC				*	
Файли, що містять аудіо і відео					
Ogg-vorbis	Ogg-theora	*	*		*
MOV-AAC	MOV-H.264			*	
MP4-AAC	MP4-H.264			*	

Як бачимо, різні Браузери підтримують різні формати. Через це в Web-дизайнерів можуть бути проблеми...

Типи MIME

По мережі передаються самі різні дані: Web-сторінки, графічні зображення, аудіо- і відеофайли, архіви, файли, що виконуються, та ін. Ці дані призначені різним програмам. До того ж, з різними даними програма, що прийняла їх, може вступитиодитися по-різному. Так, Браузер при отримаанні Web-сторінки або графічного зображення відобразить їх на екрані, а при отримаанні архіву або файлу, що виконується, — відкриє або збереже його на диску.

Всім переданим по мережі даним присвоюється особливе позначення, що однозначно вказує на їхню природу, — тип *MIME* (Multipurpose Internet Mail Extensions, багатоцільові розширення пошти Інтернету). Тип MIME присвоює даним програма, що відправляє їх,, наприклад, Web-сервер. А програма, що приймає (той же Браузер) по типу MIME прийнятих даних визначає, чи підтримує вона ці дані, і, якщо підтримує, що з ними робити.

Web-сторінка має тип MIME **`text/HTML`**. Графічне зображення формату GIF має тип MIME **`image/gif`**. Тип MIME файлу, що виконується, **— `application/x-msdownload`**, а архіву ZIP **— `application/x-zip-compressed`**. Свої типи MIME мають і мультимедійні файли.

От про мультимедійні файли і їх типи MIME ми і поговоримо.

Раніше було сказано, що сучасні Браузери працюють із дуже обмеженим набором форматів мультимедійних файлів з декількох десятків існуючих. Більше того, різні Браузери підтримують різні формати. Тому Браузер повинен визначити, чи підтримує він формат отриманого файлу, тобто чи варто його взагалі завантажувати. Як це зробити, ми вже знаємо — по типу MIME цього файлу.

У табл. 5.2 перераховані типи MIME форматів мультимедійних файлів, підтримуваних Браузерами на даний момент.

Табл. 5.2. **Типи MIME форматів мультимедійних файлів, підтримуваних сучасними Браузерами**

Формат файлів	Тип MIME
WAV	**`audio/wave`** **`audio/wav`** **`audio/x-wav`** **`audio/x-pn-wav`**
OGG	**`audio/ogg`** **(для аудіофайлов)** **`video/ogg`** **(для відеофайлів)** **`application/ogg`**
MP4	**`video/mp4`**
Quicktime	**`video/quicktime`**

Як бачимо, один формат файлів може мати кілька типів MIME. Звичайно вибирається найперший зі списку як самий кращий.

З'ясуємо, як HTML 5 дозволить нам помістити аудіо або відео на Web-сторінку.

Вставка аудіоролика

Для вставки на Web-сторінку аудіоролика мова HTML 5 використовує парний тег **<AUDIO>.** Інтернет-адресу файлу, в якому зберігається цей аудіоролик, вказують за допомогою атрибута **SRC** цього тегу:

```
<AUDIO SRC="sound.wav"></AUDIO>
```

Зустрівши тег **<AUDIO>**, Браузер може відразу завантажити і відтворити аудіофайл, тільки завантажити його без відтворення або взагалі нічого не робити. Також він може вивести на Web-сторінку елементи керування, за допомогою яких відвідувач запускає відтворення аудіофайла, припиняє його, прокручує вперед або назад і регулює гучність. Все це настроюється за допомогою різних атрибутів тегу **<AUDIO>**, які ми скоро розглянемо.

Тег **<AUDIO>** створює блоковий елемент Web-сторінки. Так що ми не зможемо вставити аудіоролик на Web-сторінку як частину абзацу. Зате, щоб помістити його в окремий абзац, нам не потрібно робити ніяких додаткових дій (на відміну від зображення).

За замовчуванням Браузер не буде відтворювати аудіоролик. Щоб він це зробив, в тегу **<AUDIO>** потрібно вказати особливий атрибут **AUTOPLAY.** Це дійсно особливий атрибут: він не має значення — досить однієї його присутності в тегу, щоб він почав діяти *(атрибут тегу без значення):*

```
<P>Зараз ви почуєте звук!</P>
<AUDIO SRC="sound.ogg" AUTOPLAY></AUDIO>
```

За замовчуванням аудіоролик ніяк не відображається на Web-Сторінці (що, втім, зрозуміло — аудіо потрібно не дивитися, а слухати). Але якщо в тегу **<AUDIO>** поставити атрибут без значення **CONTROLS**, Браузер виведе в тому місці Web- сторінки, де проставлений тег **<AUDIO>,** елементи керування відтворенням аудіоролика. Вони включають кнопку запуску й припинення відтворення, шкалу відтворення і регулятор гучності:

```
<P>Натисніть кнопку відтворення, щоб почути звук!</P>
<AUDIO SRC="sound.ogg" CONTROLS></AUDIO>
```

Атрибут без значення **AUTOBUFFER** має сенс указувати в тегу **<AUDIO>** тільки в тому випадку, якщо там відсутній атрибут **AUTOPLAY**. Якщо він зазначений, Браузер відразу після завантаження Web-Сторінки почне завантажувати файл аудіоролика — це дозволить виключити затримку файлу перед початком його відтворення.

Щоб перевірити отримані знання в дії, нам знадобиться аудіоролик підтримуваного Браузерем формату, наприклад формату WAV-PCM по імені **sound.wav**. Ви можете використовувати будь-який інший аудіоролик, але, зрозуміло, в HTML-Коді лістингу 4.2 прийде вказати ім'я файлу, у якім він зберігається.

Відкриємо Web-Сторінку index.htm і впишемо в розділ тегів тег **<AUDIO>.** Створимо, що описує цей тег Web-Сторінку, HTML-Код якої наведено в лістингу 5.2.

Лістинг 5.2

```
<!DOCTYPE HTML>
<HTML>
   <HEAD>
      <META HTTP-EQUIV="Content-Type"
       CONTENT="text/HTML; charset=utf-8">
      <TITLE>Тег AUDIO</TITLE>
   </HEAD>
   <BODY>
      <H1>Тег AUDIO</H1>
      <P>Тег AUDIO служить для вставки на Web-сторінку
       аудіоролика.</P>
      <H6>Приклад : </H6>
      <PRE>&lt;AUDIO            SRC=&quot;sound.wav&quot;
        CONTROLS&gt;&lt;/AUDIO&gt;</PRE>
      <H6>Результат:</H6>
      <AUDIO SRC="sound.wav" CONTROLS></AUDIO>
   </BODY>
</HTML>
```

Збережемо Web-сторінку у файлі з іменем **t_audio.htm**. Помістимо вибраний аудіофайл (**sound.wav**) у папку, де знаходяться усі файли нашого

Web-сайту (і **t_audio.htm** у тому числі). І відразу ж відкриємо тільки що створену Web- сторінку в Браузері (рис. 5.2).

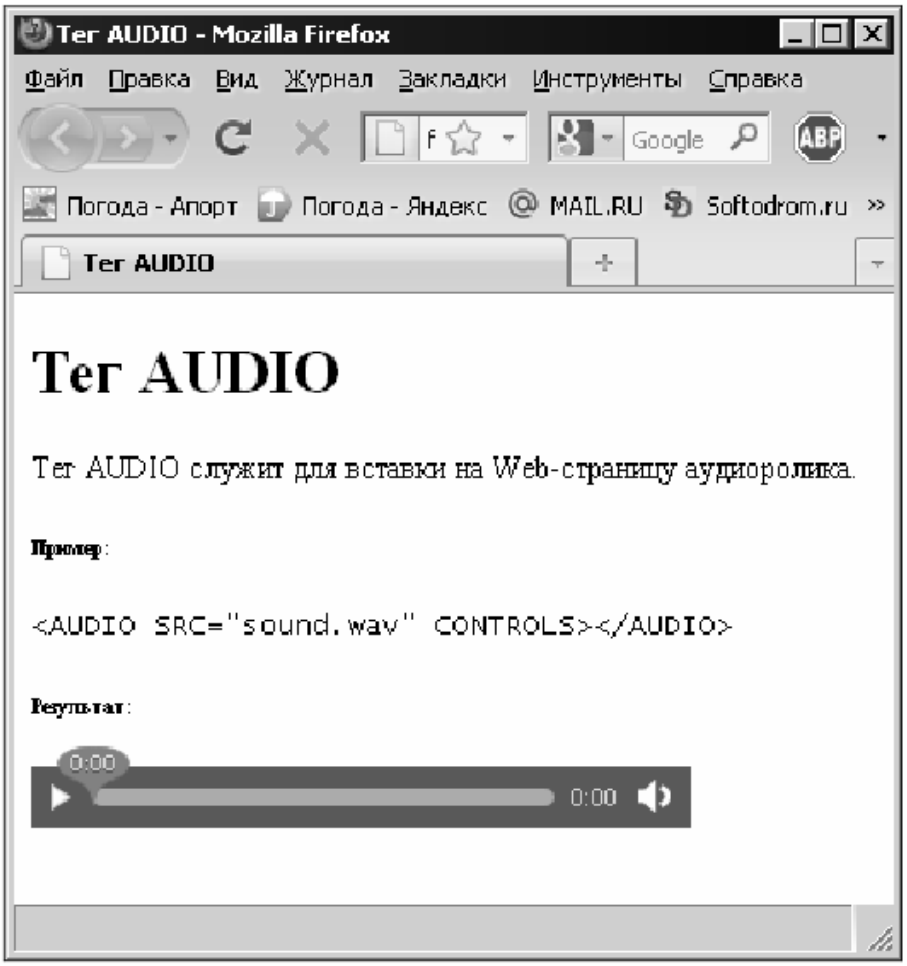


Рис. 5.2. **Web-сторінка t_audio.htm у Браузері**

Ми бачимо код прикладу та, нижче, результат його виконання -елементи для керування відтворенням аудіороліка. Ми можемо натиснути кнопку відтворення і прослухати його.

Вставка відеоролика

Для вставки на Web-сторінку відеоролика використовується парний тег **<VIDEO>.** Інтернет-адреса відеофайлу вказується за допомогою знайомого нам атрибута **SRC** цього тегу:

```
<VIDEO SRC="film.ogg"></VIDEO>
```

Зустрівши цей тег, Браузер виведе в тому місці Web-сторінки, де він проставлений, панель перегляду вмісту відеоролика. Залежно від вказаних нами в тегу **<VIDEO>** атрибутів, він може відразу завантажити та відтворити аудіофайл, тільки завантажити його без відтворення або взагалі нічого не

робити. Також він може вивести на Web-сторінку елементи керування відтворенням відеоролика.

Як і тег **<AUDIO>**, тег **<VIDEO>** створює блоковий елемент Web-сторінки і підтримує атрибути **AUTOPLAY, CONTROLS** і **AUTOBUFFER:**

```
<VIDEO SRC="film.ogg" AUTOPLAY CONTROLS></VIDEO>
```

Якщо відтворення відеоролика ще не запущене, у панелі перегляду буде виведений перший його кадр або взагалі нічого (конкретна поведінка залежить від Браузера). Але ми можемо вказати графічне зображення, яке буде туди виведене як заставка. Для цього використовується атрибут **POSTER** тегу **<VIDEO>;** його значення вказує інтернет-адресу потрібного графічного файлу:

```
<VIDEO SRC="film.ogg" CONTROLSPOSTER="filmposter.jpg">
</VIDEO>
```

Тут ми вказали для відеоролика зображення-заставку, яке буде виведено в панелі перегляду перед початком його відтворення і яке зберігається у файлі **filmposter.jpg.**

Виберемо відеоролик формату OGG і по імені **film.ogg**.

Примітка. Якщо ви не знайдете відеоролика відповідного формату, то можете самі створити його, перекодувавши відеоролик, збережений в іншому форматі. Для цього підійде утиліта **SUPER ©**, яку можна знайти за інтернет-адресою **http://www.erightsoft.com/SUPER.html .** Вона підтримує дуже багато мультимедійних форматів, у тому числі й OGG.

Відкриємо Web-сторінку **index.htm** і впишемо в розділ тегів тег **<VIDEO>.** Створимо Web-сторінку, що описує цей тег, HTML-код якої наведено в лістингу 5.3.

Лістинг 5.3

```
<!DOCTYPE HTML>
<HTML>
   <HEAD>
      <META HTTP-EQUIV="Content-Type"
       CONTENT="text/HTML; charset=utf-8">
      <TITLE>Ter VIDEO</TITLE>
   </HEAD>
   <BODY>
      <H1>Ter VIDEO</H1>
```

```
      <P>Тег VIDEO служить для вставки на Web-сторінку
       відеоролика</P>
      <H6>Приклад:</H6>
      <PRE>&lt;VIDEO SRC=&quot;film.ogg&quot;
       CONTROLS&gt;&lt;/VIDEO&gt;</PRE>
      <H6>Результат:</H6>
      <VIDEO SRC="film.ogg" CONTROLS ></VIDEO>
   </BODY>
</HTML>
```

Збережемо Web-сторінку у файлі з іменем **t_video.htm**. Помістимо вибраний відеофайл (**film.ogg**) у папку, де знаходяться усі файли нашого Web-сайту (і **t_video.htm** у тому числі). І відкриємо готову Web-сторінку **t_video.htm** в Браузері (рис. 5.3).

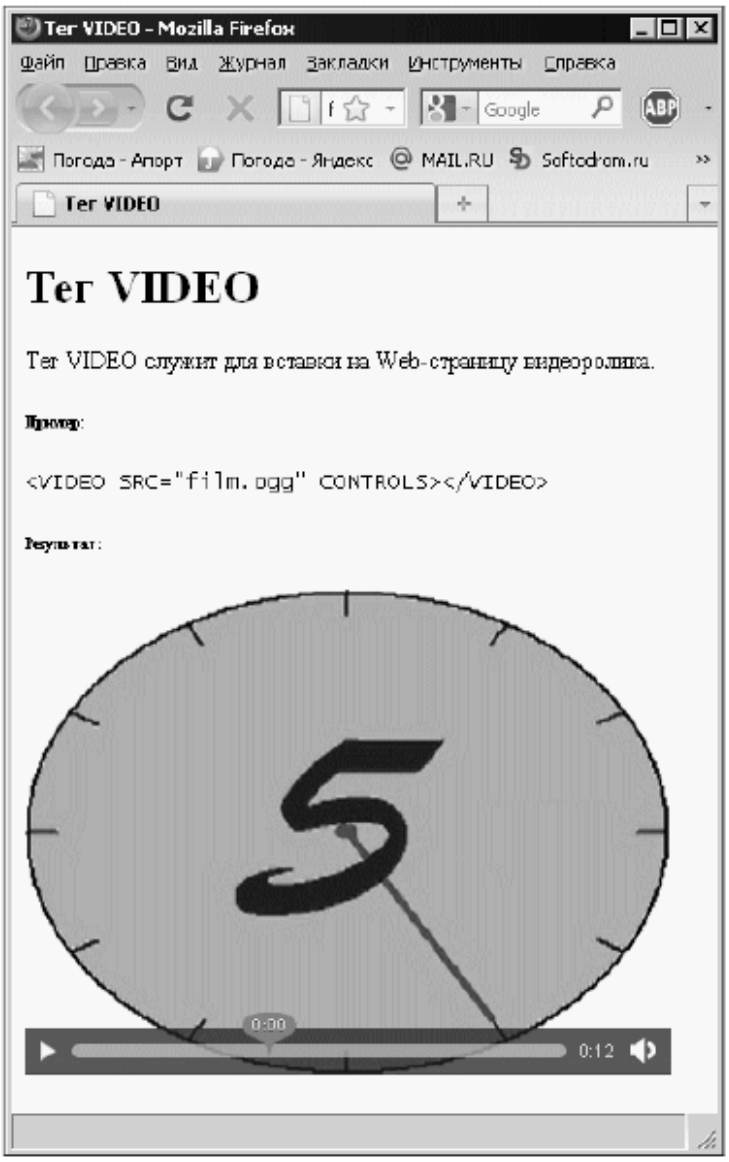


Рис. 5.3. **Web-Сторінка t_video.htm в Браузері**

Нижче коду приклада ми бачимо результат його виконання — панель перегляду та елементи для керування відтворенням. Натиснемо кнопку відтворення і подивимося "кіно". Відзначимо, що після початку відтворення елементи керування пропадуть; щоб отримати до них доступ, слід навести на панель перегляду курсор миші. Як тільки відтворення відеоролика закінчиться, ці елементи керування знову з'являться на екрані.

Додаткові можливості тегів <AUDIO> і <VIDEO>

Раніше ми довідалися, що набір підтримуваних мультимедійних форматів у різних Браузерів різниться. І може трапитися так, що аудіо- або відеоролик, який ми помістили на Web-сторінку, виявиться якомусь Браузері не "по зубах". Як бути?

Спеціально для таких випадків HTML 5 передбачає можливість вказати в одному тегу **<AUDIO>** або **<VIDEO>** відразу кілька мультимедійних файлів. Браузер сам вибере з них той, формат якого він підтримує.

Якщо ми збираємося вказати відразу кілька мультимедійних файлів в одному тегу **<AUDIO>** або **<VIDEO>**, то повинні зробити дві речі.

1. Видалити з тегу **<AUDIO>** або **<VIDEO>** вказівку на мультимедійний файл, тобто атрибут **SRC** і його значення.
2. Помістити всередині тегу **<AUDIO>** або **<VIDEO>** набір тегів **<SOURCE>**, кожний з яких буде визначати інтернет-адресу одного мультимедійного файлу.

Одинарний тег **<SOURCE>** вказує як інтернет-адресу мультимедійного файлу, так і його тип MIME. Для цього призначені атрибути **SRC** і **TYPE** даного тегу відповідно:

```
<VIDEO>
   <SOURCE SRC="film.ogg" TYPE="video/ogg">
   <SOURCE SRC="film.mov" TYPE="video/quicktime">
</VIDEO>
```

Даний тег **<VIDEO>** визначає відразу два відеофайли, що зберігають той самий фільм: один — формату OGG, інший — формату Quicktime. Браузер, що підтримує формат OGG, завантажить і відтворить перший файл, а Браузер, що підтримує Quicktime, — другий файл.

Відзначимо, що тип MIME відеофайлу (і, відповідно, атрибут **TYPE** тегу **<SOURCE>**) можна видалити. Але тоді Браузеру прийдеться завантажити початок файлу, щоб з'ясувати, чи підтримує він формат цього файлу. А це

зайве і зовсім непотрібне навантаження на мережу. Так що тип MIME краще вказувати завжди.

А якщо Браузер взагалі не підтримує теги **<AUDIO>** і **<VIDEO>**? В такому випадку він їх проігнорує s нічого на Web-сторінку не виведе. Однак ми можемо вказати текст заміни, у якому описати виниклу проблему та запропонувати який-небудь шлях її розв'язку (наприклад, перемінити Браузер). Такий текст заміни ставлять всередині тегу **<AUDIO>** або **<VIDEO>** після всіх тегів **<SOURCE>** (якщо вони є), наприклад, як в лістингу 5.4.

Лістинг 5.4

```
<VIDEO>
    <SOURCE SRC="film.ogg" TYPE="video/ogg">
    <SOURCE SRC="film.mov" TYPE="video/quicktime">
    Ваш Браузер не підтримує виведення мультимедійних
    даних засобами HTML 5. Спробуйте інший.
</VIDEO>
```

Відзначимо, що ми не вказали в тексті заміни теги, що створюють абзац. Теги <**<AUDIO>** і **<VIDEO>** самі по собі — блокові елементи, так що в цьому немає потреби.

КОНТРОЛЬНІ ПИТАННЯ

1. Що таке «вбудоване зображення»?
2. Які формати інтернет-графіки ви знаєте?
3. Розкажіть про формат *GIF*.
4. Розкажіть про формат *JPEG*.
5. Розкажіть про формат *PNG*.
6. Як вставити графічне зображення на Web-сторінку?
7. Розкажіть про формати файлів і формати кодування.
8. Що таке типи MIME?
9. Вставка аудіороліка на Web-сторінку.
10. Вставка відеороліка на Web-сторінку.
11. Як вказати в одному тегу **<AUDIO>** або **<VIDEO>** відразу кілька мультимедійних файлів?
12. Як вказати текст заміни, якщо Браузер взагалі не підтримує теги **<AUDIO>** і **<VIDEO>?**

ТЕМА 6. СТВОРЕННЯ WEB-ФОРМ І ЕЛЕМЕНТІВ КЕРУВАННЯ

Основна мета **:** познайомитися з Web-формами і елементами керування, тегами HTML для їхнього створення.

ПРИЗНАЧЕННЯ WEB-ФОРМИ І ЕЛЕМЕНТІВ КЕРУВАННЯ

Форми є одним з важливих елементів будь-якого сайту і призначені для обміну даними між користувачем та сервером.

Зараз практично жоден сайт не обходиться без елементів інтерфейсу типа полів введення тексту, кнопок, перемикачів і прапорців. Вони необхідні для взаємодії з користувачем, щоб він міг шукати на сайті по ключових словах, писати коментарі, відповідати на опитування, прикріплювати фотографії та робити багато інших подібних речей. Саме форми і забезпечують одержання даних від користувача та передачу їх на сервер, де вони вже піддаються аналізу і обробці. Тому якщо ви плануєте зробити щось подібне на сайті, без форм не вдасться це реалізувати.

Мова HTML надає набір тегів для створення різноманітних елементів керування. Ці елементи керування вже "уміють" відгукуватися на дії відвідувача: поля введення — приймати введені символи, прапорці —

встановлюватися і скидатися, перемикачі — перемикатися, списки — прокручиватися, виділяти пункти, розвертатися і згортатися, а кнопки — натискатися. Всім цим буде займатися Браузер.

Набір елементів керування, підтримуваний HTML, невеликий. Він включає поля введення різного призначення, область редагування, звичайний список та список, що розкривається, прапорець, перемикач, звичайну і графічну кнопку. Більш складні елементи керування (таблиці, "блокноти" із вкладками, панелі інструментів та ін.) так просто створити не вийде. Хоча, як правило, для створення простих Web-додатків перерахованого обмеженого набору цілком достатньо.

Стандарт HTML вимагає, щоб всі елементи керування перебували всередині Web-форми. ***Web-форма*** — це особливий елемент Web-сторінки, що служить "вмістищем" для елементів керування. На Web-сторінці вона ніяк не відображається (якщо, звичайно, ми не задамо для неї якого-небудь представлення); в цьому Web-форма схожа із блоковим контейнером. Для створення Web-форми HTML передбачає особливий тег.

Стандарт HTML вимагає, щоб кожний елемент керування мав унікальне в межах Web-форми ім'я. Це потрібно для успішного формування пар даних перед відправленням їх серверному додатку.

СТВОРЕННЯ WEB-ФОРМ І ЕЛЕМЕНТІВ КЕРУВАННЯ

Розглянемо засоби мови HTML, призначені для створення Web-форм і елементів керування.

Створення Web-форм

Для створення Web-форми застосовується парний тег **<FORM>**, всередині якого поміщають теги, які формують елементи керування, що входять в цю Web-форму:

```
<FORM>
    <теги, що формують елементи керування>
</FORM>
```

Web-форма поводиться як блоковий елемент Web-сторінки.

Тег **<FORM>** підтримує обов'язковий атрибут **ACTION**, який вказує інтернет-адресу серверного додатка. Якщо Web-форма служить для введення даних, призначених для обробки Web-сценарієм, значенням цього атрибута тега вказують "пусту" інтернет-адресу **#**:

<FORM ACTION="#">

</FORM>

Створення елементів керування

Більшість елементів керування HTML створюють за допомогою одинарного тегу **<INPUT>.** Який саме елемент керування слід створити, вказують за допомогою необов'язкового атрибута **TYPE** цього тегу. Деякі елементи керування, такі як область редагування і списки, створюють за допомогою інших тегів. Ми їх розглянемо.

На основі даних, введених в елементи керування, Web-форма, у якій ці елементи керування знаходяться, сформує пари вигляду **<ім'я елемента керування>=<введені в нього дані>**, які відправить серверному додатку. Ім'я елемента керування визначається атрибутом **NAME** тега **<INPUT>** або іншим тегом, припустимим у формі, а значення вводиться користувачем або встановлюється в поле форми за замовчуванням. Атрибут тега **NAME** -це обов'язковий атрибут.

Всі елементи керування HTML є вбудованими елементами Web-сторінки.

Тепер розглянемо всі елементи керування HTML і особливості їх створення.

Поле введення

Поле введення — найпоширеніший елемент керування в Web-формах — створюється за допомогою одинарного тегу **<INPUT>:**

<INPUT [TYPE="text"] [VALUE="*<початкове значення>*"] [SIZE="*<розмір>*"] [MAXLENGTH="*<максимальна кількість символів>*"] [DISABLED] [TABINDEX="*<номер в порядку обходу>*"] [ACCESSKEY="*<швидка клавіша>"]* [READONLY] [AUTOFOCUS]>

Атрибут тегу **TYPE**, як вже було сказано, задає тип елемента керування. Значення "**text**" вказує Браузеру створити саме поле введення. Поле введення також створюється, якщо атрибут тегу **TYPE** не вказаний (як вже зазначалося, він необов'язковий).

Необов'язковий атрибут тегу **VALUE** задає значення, яке повинне бути присутнім в поле введення спочатку. Якщо цей атрибут не вказаний, поле введення не буде містити нічого.

Необов'язковий атрибут тегу **SIZE** задає довжину поля введення в символах. Якщо він не вказаний, довжина поля введення буде залежати від Браузера.

Необов'язковий атрибут тегу **MAXLENGTH** задає максимальний розмір рядка, який можна ввести в це поле введення, в символах. Якщо цей атрибут тегу не вказаний, в поле введення можна буде ввести рядок необмеженого розміру.

Необов'язкові атрибути тегу **TABINDEX** і **ACCESSKEY** задають, відповідно, номер в порядку обходу і "гарячу" клавішу для доступу до елемента керування.

Атрибут тегу без значення **DISABLED** дозволяє зробити поле введення недоступним для відвідувача; воно буде відображатися сірим кольором, і відвідувач не зможе навіть його активізувати. Якщо цей атрибут присутній в тегу, поле введення недоступно, якщо відсутній — доступно.

Атрибут тегу без значення **READONLY** дозволяє зробити поле введення доступним тільки для читання; при цьому відвідувач все-таки зможе активізувати це поле, виділити текст, що міститься в ньому. і скопіювати його в Буфер обміну. Якщо цей атрибут тегу присутній, поле введення буде доступно тільки для читання, якщо відсутній — доступно і для читання, і для введення.

Якщо атрибут тегу без значення **AUTOFOCUS** є присутнім, дане поле введення буде автоматично активізоване при відкритті Web-сторінки. Якщо ж він відсутній, поле введення активізоване не буде, і відвідувачеві доведеться його активізувати клацанням мишею або клавішами <**Tab**> або <**Shift**>+<**Tab**>**.**

Помітимо, що атрибут тегу **AUTOFOCUS** можна вказувати тільки для одного елемента керування на всій Web-сторінці.

Лістинг 6.1

```
<FORM ACTION="#">
  <P>Ім'я:  <INPUT  TYPE="text"  NAME="name1"  SIZE="20"
  AUTOFOCUS></P>
  <P>Прізвище: <INPUT TYPE="text" NAME="name2" SIZE="30"></P>
</FORM>
```

В лістингу 6.1 ми створюємо Web-форму із двома полями введення: **name1** довжиною 20 символів, що автоматично активізується при відкритті Web-сторінки, і **name2** довжиною 30 символів. Обидва поля введення мають написи, що є звичайним текстом і розташовані перед ними.

Зверніть увагу, що для розміщення елементів керування в Web-формі ми використовували абзаци. Взагалі, для цього можна застосовувати будь-які елементи Web-сторінок із уже знайомих нам: списки, таблиці та ін.

Поле введення пароля

Поле введення пароля нічим не відрізняється від звичайного поля введення за тим виключенням, що замість символів, які вводяться, в ньому відображаються крапки. Такі поля введення широко застосовують для запиту паролів і інших конфіденційних даних.

Поле введення пароля також створюється за допомогою одинарного тегу **<INPUT>:**

<INPUT TYPE="password" [VALUE="*<початкове значення>"]* [SIZE="<розмір>"] [MAXLENGTH="*<максимальна кількість символів>"]* [TABINDEX="*<номер в порядку обходу>"]* [ACCESSKEY="*<швидка клавіша>"]* [DISABLED] [READONLY] [AUTOFOCUS]>

Значення "**password**" атрибута тегу **TYPE** вказує Браузеру створити поле введення пароля. Інші атрибути вже знайомі по звичайному полю введення.

Лістинг 6.2

```
<FORM ACTION="#">
   <P>Ім'я: <INPUT TYPE="text" NAME="login" SIZE="20"
   AUTOFOCUS></P>
   <P>Пароль: <INPUT TYPE="password" NAME="password" SIZE="20"></P>
</FORM>
```

В лістингу 6.2 ми створюємо Web-форму зі звичайним полем введення і полем введення пароля. Перше —довжиною 20 символів, буде автоматично активізуватися при відкритті Web-сторінки. Друге —довжиною також 20 символів.

Поле введення значення для пошуку

Поле введення значення для пошуку з'явилося в HTML 5. Воно нічим не відрізняється від звичайного поля введення за тим виключенням, що з введеного в нього значення автоматично видаляються перенесення рядків.

Поле введення значення для пошуку також створюється за допомогою одинарного тегу **<INPUT>:**

```
<INPUT TYPE="search" [VALUE="<початкове значення>"]
[SIZE="<розмір>"] [MAXLENGTH="<максимальна кількість
символів>"] [TABINDEX="<номер в порядку обходу>"]
[ACCESSKEY="<швидка клавіша>"] [DISABLED] [READONLY]
[AUTOFOCUS]>
```

Значення "**search**" атрибута тегу **TYPE** вказує Браузеру створити поле введення значення для пошуку. Інші атрибути нам уже знайомі по звичайному полю введення (лістинг 6.3).

Лістинг 6.3

```
<FORM ACTION="#">
   <P>Знайти: <INPUT TYPE="search" NAME="keyword" SIZE="40"></P>
</FORM>
```

Область редагування

Область редагування створюється парним тегом **<TEXTAREA>:**

```
<TEXTAREA [ROWS="<висота>"] [COLS="<ширина>"]
[WRAP="off|soft|hard"] [TABINDEX="<номер в порядку
обходу>"] [ACCESSKEY="<швидка клавіша>"] [DISABLED]
[READONLY] [AUTOFOCUS]>
    <початкове значення>
</TEXTAREA>
```

Значення, яке повинне спочатку бути присутнім в області редагування, міститься всередину тегу **<TEXTAREA>**. Це повинен бути текст без всяких HTML-тегів.

Необов'язковий атрибут тегу **ROWS** задає висоту області редагування в рядках. Якщо він не вказаний, висота області редагування буде залежати від Браузера.

Необов'язковий атрибут тегу **COLS** задає ширину області редагування в символах. Якщо він не вказаний, висота області редагування буде залежати від Браузера.

Необов'язковий атрибут тегу **WRAP** дозволяє управляти переносом рядків в області редагування. Атрибут **WRAP** може приймати два значення:

- "**soft**" — область редагування буде автоматично виконувати перенесення занадто довгих рядків. При цьому в саме значення, введене в область редагування, символи перенесення рядків вставлятися не будуть.
- "**hard**" — область редагування буде автоматично виконувати перенесення занадто довгих рядків. При цьому у відповідні місця значення, введеного в область редагування, будуть вставлені символи перенесення рядків.

Якщо атрибут тегу **WRAP** не вказаний, область редагування буде поводитися так, немов задане значення "**soft**".

Інші атрибути, підтримувані тегом **<TEXTAREA>,** нам уже знайомі (лістинг 6.4).

Лістинг 6.4

```
<FORM ACTION="#">
   <P>
   Введіть сюди ваш відгук про Web-сайт:<BR>
   <TEXTAREA NAME="opinion" COLS="60" ROWS="10">
   Відмінний Web-сайт!
   </TEXTAREA>
   </P>
</FORM>
```

Кнопка

Кнопка при натисканні запускає на виконання яку-небудь дію. Вона створюється за допомогою тегу **<INPUT>:**

```
<INPUT TYPE="button" VALUE="<напис>"
[TABINDEX="<номер у порядку обходу>"]
[ACCESSKEY="<швидка клавіша>"] [DISABLED] [AUTOFOCUS]>
```

Значення "button" атрибута тегу **TYPE** вказує Браузеру створити звичайну кнопку. Атрибут тегу **VALUE**, що задає напис для кнопки, в цьому випадку є обов'язковим. Інші атрибути тегу нам уже знайомі (лістинг 6.5).

Лістинг 6.5

```
<FORM ACTION="#">
   <P>
      Знайти:
      <INPUT TYPE="search" NAME="keyword" SIZE="40">
      <INPUT TYPE="button" NAME="find" VALUE="Шукати!">
   </P>
</FORM>
```

Прапорець

Прапорці зустрічаються в Web-формах нечасто, у випадках, коли потрібно дати відвідувачеві можливість вибрати або не вибрати якусь опцію. Для створення прапорців застосовується тег **<INPUT>:**

```
<INPUT TYPE="checkbox" [CHECKED]
[TAB INDEX="<номер в порядку обходу>"]
[ACCESSKEY="<швидка клавіша>"] [DISABLED] [AUTOFOCUS]>
```

Значення "**checkbox**" атрибута тегу **TYPE** вказує Браузеру створити саме прапорець.

Атрибут тегу без значення **CHECKED** дозволяє зробити прапорець встановленим спочатку. Якщо він присутній, прапорець буде встановлений спочатку, якщо відсутній — скинутий.

Інші атрибути тегу нам уже знайомі (лістинг 6.6).

Лістинг 6.6

```
<FORM ACTION="#">
   <P>
      <INPUT TYPE="checkbox" NAME="updates" CHECKED>
      Я хочу отримувати листи із списком відновлень Web-сайту
      </P>
</FORM>
```

Перемикач

Перемикачі в Web-формах, як і у вікнах Windows-додатків, застосовуються тільки групами. Група перемикачів надає відвідувачеві можливість вибрати одну з декількох доступних альтернатив. Поодинці ж перемикачі абсолютно некорисні — прапорці в таких випадках набагато зручніше.

Створюється перемикач за допомогою все того ж тегу **<INPUT>**:

```
<INPUT TYPE="radio" [CHECKED]
[TABINDEX="<номер у порядку обходу>"] [ACCESSKEY="<швидка клавіша>"] [DISABLED] [AUTOFOCUS]>
```

Значення "**radio**" атрибута тегу **TYPE** вказує Браузеру створити саме перемикач. Інші атрибути тегу нам уже знайомі.

В групі тільки один перемикач може бути встановлений. Це значить, що атрибут тегу без значення **CHECKED** можна вказувати тільки для одного перемикача в групі. Лістинг 6.7 містить приклад перемикача.

Лістинг 6.7

```
<FORM ACTION="#">
   <P>
     <INPUT TYPE="radio" NAME="updates" CHECKED>
     Я прагну одержувати листи зі списком відновлень Web-сайту
   </P>
   <P>
     <INPUT TYPE="radio" NAME="updates">
     Я не прагну одержувати листи зі списком відновлень Web-сайту
   </P>
</FORM>
```

Список звичайний або список, що розкривається

Списки, як звичайні, так і ті, що розкриваються, реалізують за допомогою парного тегу **<SELECT>**, усередині якого поміщають парні теги **<OPTION>,** що створюють пункти списку.

Почнемо з парного тегу **<SELECT>,** який створює сам список:

```
<SELECT [SIZE="<висота в пунктах (позиціях)>"]
[MULTIPLE] [TAB INDEX="<номер в порядку обходу>"]
[ACCESSKEY="<швидка клавіша>"] [DISABLED] [AUTOFOCUS]>
```

```
    <теги <OPTION>, що створюють пункти списку>
</SELECT>
```

Необов'язковий атрибут тегу **SIZE** задає висоту списку в пунктах (маються на увазі пункти списку). Якщо цей атрибут тега має значення, відмінне від одиниці, створюється звичайний список, що має висоту, рівну вказаному значенню пунктів. Якщо ж значення цього атрибуту тега дорівнює одиниці або взагалі відсутнє, створюється список, що розкривається (він має у висоту один пункт).

Атрибут тегу без значення **MULTIPLE** дозволяє створити список, в якому можна вибрати відразу кілька пунктів. Якщо цей атрибут тегу присутній, в списку можна буде вибрати відразу кілька пунктів, якщо відсутній — можна буде вибрати тільки один пункт. Зрозуміло, що даний атрибут тега має сенс вказувати тільки для звичайних списків (якщо атрибут тегу **SIZE** має значення, відмінне від 1); в списку, що розкривається, в кожному разі можна вибрати тільки один пункт.

Інші атрибути цього тега нам уже знайомі.

Парний тег **<OPTION>** створює окремий пункт списку. Він може бути присутнім тільки в тегу **<SELECT>:**

```
<OPTION [LABEL="<текст пункту>"] [SELECTED] [DISABLED]>
   [<текст пункту>]
</OPTION>
```

Текст пункту списку або поміщають усередину тегу **<OPTION>,** або вказують за допомогою атрибута тега **LABEL.**

Атрибут тега без значення **SELECTED** дозволяє зробити даний пункт списку спочатку вибраним. Якщо цей атрибут тегу вказаний, пункт списку буде спочатку вибраним, якщо не вказаний — не буде вибраним.

Уже знайомий нам атрибут тегу без значення **DISABLED** дозволяє зробити даний пункт недоступним для вибору. Лістинг 6.8 ілюструє приклад.

Лістинг 6.8

```
<FORM ACTION="#">
  <P>
     Виконувати пошук по
     <SELECT NAME="search_in">
       <OPTION>назвам</OPTION>
       <OPTION>ключовим словам</OPTION>
       <OPTION SELECTED>назвам і ключовим словам</OPTION>
```

```
      </SELECT>
   </P>
</FORM>
```

HTML також дозволяє поєднувати пункти списку в групи по якій-небудь родинній ознаці. Таку групу створюють за допомогою парного тегу **<OPTGROUP>**; в нього поміщають теги, що створюють пункти списку, які входять у цю групу. Тег **<OPTGROUP>** може бути присутнім тільки всередині тегу **<SELECT>:**

```
<OPTGROUP LABEL="<заголовок групи>" [DISABLED]>
   <теги <OPTION>, що створюють пункти, які входять в групу>
<OPTGROUP>
```

Обов'язковий в цьому випадку атрибут тегу **LABEL** задає заголовок групи. А атрибут тегу без значення **DISABLED** дозволяє зробити всі пункти даної групи недоступними для вибору (лістинг 6.9).

Лістинг 6.9

```
<FORM ACTION="#">
   <P>
      Виконувати пошук по
      <SELECT NAME="search_in">
         <OPTGROUP LABEL="Швидкий пошук">
            <OPTION>назвам</OPTION>
            <OPTION ключовим словам</OPTION>
         </OPTGROUP>
         <OPTION SELECTED>назвам і ключовим словам</OPTION>
      </SELECT>
   </P>
</FORM>
```

Напис

Строго кажучи, ***напис*** — це не елемент керування. Він просто задає для елемента керування текстовий напис, який описує його призначення. Якщо відвідувач клацне мишею на написі, елемент керування буде активований.

Напис створюють за допомогою парного тегу **<LABEL>:**

```
<LABEL [FOR="<ім'я елемента керування, до якого відноситься напис>"] [TABINDEX="<номер в порядку обходу>"] [ACCESSKEY="<швидка клавіша>"]>
    <текст напису>[<елемент керування>]
</LABEL>
```

Є два способи прив'язати напис до елемента керування, який він повинен описувати. Зараз ми їх розглянемо.

При першому способі (лістинг 6.10) елементу керування, до якого прив’язано напис, дають ім'я, як значення обов'язкового в такому випадку атрибута **FOR** тегу **<LABEL>**, що створює напис.

Лістинг 6.10

```
<FORM ACTION="#">
  <P><LABEL FOR="keyword">Знайти:</LABEL>
  <INPUT TYPE="search" NAME="keyword" SIZE="40"></P>
</FORM>
```

При другому способі (лістинг 6.11) елемент керування, до якого прив’язано напис, поміщають у сам тег **<LABEL>**, що створює його, відразу після тексту напису.

Лістинг 6.11

```
<FORM ACTION="#">
  <P><LABEL>Знайти:<INPUT TYPE="search" NAME="keyword"
  SIZE="40"></LABEL></P>
</FORM>
```

Написи в Web-формах зустрічаються досить рідко. Звичайно Web-дизайнери обмежуються простим текстом, який ставлять до або після елемента керування.

Група

Групу також не можна віднести до "ссправжніх" елементів керування. Вона поєднує кілька елементів керування, що мають подібне призначення. Візуально група являє собою рамку, що оточує елементи керування і, можливо, що має заголовок, розташований прямо на її верхній або нижній границі.

Групу створюють за допомогою парного тегу **<FIELDSET>**

```
<FIELDSET>
<елементи керування, поєднувані в групу>
</FIELDSET>
```

Ми бачимо, що теги, які створюють елементи керування і які повинні бути об'єднані в групу, поміщають прямо в тег **<FIELDSET>.**

Крім того, в тегу **<FIELDSET>** може бути присутнім парний тег **<LEGEND>,** що створює заголовок групи:

```
<LEGEND [ACCESSKEY ="<швидка клавіша>"]><текст заголовка>
</LEGEND >
```

Текст заголовка поміщають прямо всередині цього тегу.

Тег **<LEGEND>** повинен міститися або відразу ж після відкриваючого тегу **<FIELDSET>,** або перед закриваючим тегом **</FIELDSET>.** В першому випадку заголовок буде присутній на верхній границі групи, в другому випадку — на нижній границі. В лістингу 6.12 наведений приклад групи.

Лістинг 6.12

```
<FORM ACTION="#">
  <FIELDSET>
    <LEGEND>Знайти:</LEGEND>
    <P>
      <INPUT TYPE="search" NAME="keyword" SIZE="40">
      <INPUT TYPE="button" NAME=="find" VALUE="Шукати!">
    </P>
  </FIELDSET>
</FORM>
```

Інші елементи керування

HTML дозволяє створити ще кілька елементів керування, які необхідні тільки для взаємодії із серверними додатками.

Насамперед, це ***кнопка відправлення даних***, про яку ми вже говорили. Вона відрізняється від звичайної кнопки тільки значенням атрибута **TYPE** тегу **<INPUT>**— "**submit**".

Далі, в Web-формі може бути присутня ***кнопка очищення***. При натисканні на таку кнопку всі елементи керування в Web-формі отримують початкові значення, задані в HTML-коді. Значення атрибута **TYPE** тегу **<INPUT>**, що створює подібну кнопку, повинне бути "**reset**".

Поле введення імені файлу служить для вказання імені файлу, який буде відправлений серверному додатку (сам файл, а не його ім'я). Воно складається із власне поля введення і розташованої правіше його кнопки **Обзор,** при натисканні на яку на екрані з'явиться стандартне діалогове вікно відкриття файлу Windows, у якому можна вибрати файл, що відправляється.

Поле введення імені файлу відрізняється від звичайного поля введення значенням атрибута **TYPE** тегу **<INPUT>** — "**file**". В тегу **< INPUT >** в цьому випадку підтримуються атрибути **ACCESSKEY, AUTOFOCUS, DISABLED, SIZE і TABINDEX.**

Графічна кнопка відправлення даних — це графічне зображення, при клацанні на якому Web-форма запускає процес відправлення введених даних серверному додатку. Фактично це кнопка відправлення даних, в якості якої виступає зображення.

Графічну кнопку відправлення даних створюють за допомогою тегу **<INPUT>.** Значення атрибута **TYPE цього** тегу повинне бути "**image**". Атрибут тегу **SRC** задає інтернет-адресу файлу із графічним зображенням, а атрибут тегу **ALT** — текст заміни. Також підтримуються атрибути **ACCESSKEY**, **AUTOFOCUS, DISABLED і TABINDEX** тегу **<INPUT>.**

Приховане поле — це фактично взагалі не елемент керування, оскільки ніяк не відображається на Web-сторінці. Воно служить для зберігання яких-небудь службових даних, необхідних для серверного додатка, показ яких відвідувачеві небажаний.

Приховане поле створюють за допомогою тегу **<INPUT>.** Значення атрибута **TYPE** цього тегу повинне бути "**hidden**". Атрибут **VALUE** тегу **<INPUT>** задає збережене в прихованому полі значення.

КОНТРОЛЬНІ ПИТАННЯ

1. Web-форми і елементи керування HTML. Призначення Web-форм і елементів керування.
2. Створення Web-форм.
3. Створення елементів керування: поле введення.

4. Створення елементів керування: поле введення пароля.
5. Створення елементів керування: поле введення значення для пошуку.
6. Створення елементів керування: область редагування.
7. Створення елементів керування: кнопка.
8. Створення елементів керування: прапорець.
9. Створення елементів керування: перемикач.
10. Створення елементів керування: список звичайний або список, що розкривається.
11. Створення елементів керування: напис.
12. Створення елементів керування: група.
13. Створення елементів керування: кнопка відправлення даних, графічна кнопка відправлення даних, кнопка очищення.
14. Створення елементів керування: поле введення імені файлу, приховане поле.

ТЕМА 7. ЗАСОБИ НАВІГАЦІЇ

***Основна мета*:** вивчити засоби навігації, пропоновані мовою HTML, а саме — всілякі гіперпосилання; навчитися зв'язувати всі створені Web-сторінки в єдиний Web-сайт.

Ми навчилися наповнювати Web-сторінки вмістом: текстом, графічними зображеннями, аудіо- і відеороликами та таблицями. Залишилося зв'язати ці розрізнені Web-сторінки воєдино — в Web-сайт. Як це здійснити? За допомогою засобів навігації — гіперпосилань.

Гіперпосилання звичайно виглядає як підкреслений фрагмент тексту; якщо навести на нього курсор миші, він прийме вид "перста, що вказує". При клацанні на гіперпосиланні Браузер завантажить Web-сторінку, інтернет-адреса якої указана в параметрах даного гіперпосилання *(цільову Web*-сторінку). Гіперпосилання може мати вигляд графічного зображення або його фрагмента, такі гіперпосилання зараз дуже популярні.

ТЕКСТОВІ ГІПЕРПОСИЛАННЯ

Найпростіші гіперпосилання — *текстові* гіперпосилання, які є собою фрагментами тексту.

Створення гіперпосилань

Створити текстове гіперпосилання дуже просто. Досить знайти в блоковому елементі (наприклад, абзаці) фрагмент тексту, який потрібно перетворити в гіперпосилання, і вкласти його в парний тег **<A>.** Інтернет-адресу цільової Web-сторінки вказують в атрибуті **HREF** цього тегу.

Гіперпосилання (тобто тег **<A>**) є вбудованим елементом Web-сторінки, тобто це частина блокового елемента, наприклад, абзацу:

От гіперпосилання, яке вказує на Web-сторінку **page125.html**, що зберігається в папці **pages**, вкладеній в кореневу папку сайту, на сайті **http://www.somesite.ua:**

```
<A HREF="http://www.somgsite.ua/pages/page125.html">Страница
№125</A>
```

А це гіперпосилання вказує на архівний файл **archive.zip**, що зберігається в тій же папці, що і Web-сторінка, яка в цей момент відкрита в Браузері *(поточна Web*-сторінка):

```
<A HREF="archive.zip">Архив</A>
```

При клацанні на гіперпосиланні Браузер запропонує завантажити цей архівний файл і або відкрити його, або зберегти на диску клієнтського комп'ютера.

Приклад:

```
<P><A HREF="22.html">Попередня сторінка</A>,
<A HREF="24.html">Наступна сторінка</A>.</P>
```

Цей фрагмент HTML-коду створює абзац, що містить відразу два гіперпосилання, які вказують на різні цільові Web-сторінки.

Текст, що є гіперпосиланням, можна оформляти, використовуючи будь-які теги, вивчені раніше.

Приклад:

```
<A HREF="http://www.somgsite.ua/pages/page125.html">Страница
<EM>№125</EM></A>
```

Тег **<A>** підтримує необов'язковий атрибут **TARGET**. Він задає *мету гіперпосилання*, що вказує, де буде відкрита цільова Web-сторінка. Так, якщо атрибуту **TARGET** присвоїти значення **"_blank"**, цільова сторінка буде відкрита в новому вікні Браузера.

Наприклад, якщо ми змінимо код першого прикладу гіперпосилання в такий спосіб (виправлення виділені напівжирним шрифтом):

```
<A HREF="http://www.somgsite.ua/pages/page125.html" TARGET="_blank">Страница_№125</A>
```

"Сторінка №125" буде відкрита в новому вікні Браузера.

Щоб задати звичайну поведінку гіперпосилання (коли цільова Web-сторінка відкривається в тому ж вікні Браузера), потрібно присвоїти атрибуту **TARGET** значення **"_self"** (це його значення за замовчуванням) або взагалі видалити даний атрибут з тегу **<A>.**

Є також можливість створити гіперпосилання, яке нікуди не вказує *("порожня"* гіперпосилання). Для цього досить задати як значення атрибута **HREF** значок # ("ґрати"):

```
<A HREF="#">А я нікуди не веду!</A>
```

При клацанні на такому гіперпосиланні нічого не відбудеться.

Правила відображення гіперпосилань Браузером:

- звичайні гіперпосилання виділяються синім кольором;
- гіперпосилання, по яких відвідувач уже "ходив" *(*гіперпосилання, які *відвідав*), виводяться темно-червоним кольором;
- гіперпосилання, по якому відвідувач в цей момент клацає *(активне* гіперпосилання), виводиться яскраво-червоним кольором;
- текст будь-яких гіперпосилань підкреслюється;
- при приміщенні курсору миші на гіперпосилання Браузер міняє його форму на "перст, що вказує".

Така поведінка за замовчуванням, яке ми можемо змінити, створивши відповідне представлення. Про те, як це зробити, буде розказано далі.

Інтернет-адреси в WWW

Про інтернет-адреси файлів ми вже говорили.

Розглянемо перший приклад гіперпосилання з попереднього розділу. Його інтернет-адреса така: **http://www.somesite.ua/pages/page125.html.** Вона містить і інтернет-адресу Web-сервера, і шлях файлу, який потрібно отримати. Тому вона називається *повною.* Повні інтернет-адреси використовують, якщо потрібно створити гіперпосилання, що вказує на файл, у складі іншого Web-сайту.

Однак якщо гіперпосилання вказує на файл, що входить до складу того ж Web-сайту, що і файл поточної Web-сторінки, краще використовувати *скорочену* інтернет-адресу, що містить тільки ім'я потрібного файлу (інтернет-адреса Web-сервера і так відома Браузеру).

Існують два типи скорочених інтернет-адрес. Адреси першого типу задають шлях до файлу, який потрібно отримати *(цільового* файлу), відносно кореневої папки Web-сайту. Вони містять в своєму початку символ / (слеш), який і говорить Web-серверу, що шлях потрібно відраховувати щодо кореневої папки.

Примітка. Коренева папка сайту - це особлива папка, що знаходиться на диску комп'ютера, на якому зберігається Web-сайт і працює Web-сервер; у цій папці повинні знаходитися всі файли Web-сайту.

Наприклад, інтернет-адреса

```
/page3.html
```

вказує на файл **page3.html**, що зберігається в кореневій папці Web-сайту.

А інтернет-адреса

```
/articles/article1.html
```

вказує на файл **article 1.html**, що зберігається в папці **articles**, вкладеної в кореневу папку Web-сайту.

Такі інтернет-адреси називають *абсолютними* і використовують, якщо потрібно створити гіперпосилання на файл, що зберігається в "глибині" Web-сайту (скажемо, в іншій папці, ніж файл поточної Web-сторінки).

Скорочені інтернет-адреси другого типу задають шлях до цільового файлу відносно файлу поточної Web-сторінки. Вони не містять на початку символу слеша — і в цьому їхня важлива відмінність від абсолютних інтернет-адрес.

Розглянемо кілька прикладів подібних інтернет-адрес.

Наприклад, інтернет-адреса

```
archive.zip
```

вказує на файл **archive.zip**, що зберігається в тій же папці, що і файл поточної Web-сторінки.

Інтернет-адреса

```
chapter3/page1.html
```

вказує на Web-сторінку **page1.html**, що зберігається в папці **chapter3**, вкладеній в папку, в якій зберігається поточна Web-сторінка.

Наступна інтернет-адреса

```
../contents.html
```

вказує на Web-сторінку **contents.html**, що зберігається в папці, у яку вкладена папка, де зберігається поточна Web-сторінка. (Зверніть увагу на дві крапки на початку шляху — так позначається папка попереднього рівня вкладеності).

Такі інтернет-адреси називають *відносними.* Їх застосовують, якщо потрібно створити гіперпосилання на файл, що зберігається в тій же папці, що і поточна Web-сторінка, одній із вкладених в неї папок або папці попереднього рівня вкладеності.

Увага! В Web-сторінках, які не повинні бути опубліковані на Web-серверах, а будуть відкриватися з диска клієнтських комп'ютерів, слід застосовувати тільки відносні інтернет-адреси. Справа в тому, що файлова система комп'ютера не знає, яку папку вважати кореневою, тому не зможе правильно інтерпретувати абсолютні інтернет-адреси. (Зрозуміло, гіперпосилання, що посилаються на інші Web-сайти, повинні містити повні інтернет-адреси).

Розібравшись із гіперпосиланнями та інтернет-адресами, зв'яжемо, нарешті, наші Web-сторінки в єдиний Web-сайт. Щоб нам було зручніше, створимо в папці, де зберігаються всі файли нашого Web-сайту, папку **tags**. В цю папку перенесемо всі Web-сторінки, що описують теги HTML (їх у нас поки що чотири) файли, що їх супроводжують (їх три: зображення, аудіо- і відеоролик). Файл **index.htm** нікуди з кореневої папки переміщати не будемо — адже він зберігає Web-сторінку за замовчуванням.

Відкриємо Web-сторінку **index.htm** і знайдемо в ній HTML-код, що формує список тегів. Створимо там гіперпосилання, що вказують на відповідні Web-сторінки.

От HTML-код, який створює гіперпосилання, що вказує на Web-сторінку з описом тегу **<AUDIO>**:

```
<CODE><A HREF="tags/t_audio.htm">AUDIO</A></CODE>
```

Інші гіперпосилання створюються аналогічно.

Перейшовши на Web-сторінку, що описує який-небудь тег, відвідувач повинен мати можливість повернутися назад — на головну Web-сторінку. Звичайно, це можна зробити, натиснувши кнопку повернення назад на панелі

інструментів Браузера або клавішу <**Backspace**> на клавіатурі. Але правила гарного тону Web-дизайну вимагають, щоб на Web-сторінці була присутня відповідне гіперпосилання.

Створимо таке гіперпосилання на всіх Web-сторінках, що описують теги. Помістимо його в самому кінці кожної Web-сторінки — звичайно воно знаходиться саме там. От так виглядає формуючий її HTML-код:

```
<P><A HREF="../index.htm">На головну Web-сторінку</A></P>
```

Залишилося створити на головній Web-сторінці гіперпосилання на Web-сторінку Вікіпедії, яка містить статтю, присвячену мові HTML. Вставимо її в кінець великої цитати (лістинг 7.1).

Лістинг 7.1

```
<BLOCKQUOTE>
    <P>HTML (від англ. Hypertext Markup Language — мова
    розмітки гіпертексту) — стандартна мова розмітки
    документів у Всесвітній павутині.
    (<A HREF="http://ru.wikipedia.org/wiki/HTML"
    TARGET="_blank">вся стаття</A>)</P>
</BLOCKQUOTE>
```

Тут ми вказали для тегу **<A>** атрибут **TARGET** із значенням **"_blank"**. І Web-сторінка з текстом статті про HTML буде відкриватися в новому вікні Браузера. Так що відвідувач зможе "залізти" у Вікіпедію, не залишаючи нашого Web-сайту.

Поштові гіперпосилання

HTML дозволяє нам створити гіперпосилання, що вказує на адресу електронної пошти *(поштове гіперпосилання)*. Клацання на ньому запустить програму поштового клієнта, встановлену в системі за замовчуванням. Інтернет-адреса такого гіперпосилання записується особливим способом.

Нехай ми намагаэмось створити гіперпосилання, що вказує на поштову адресу:

user@mailserver.ua

Згідно зі стандартом HTML, ця поштова адреса повинна бути записана так:

```
mailto:user@mailserver.ua
```

причому між двокрапкою після **mailto** і власне адресою не повинно бути пробілів.

Тоді наше поштове гіперпосилання буде виглядати так:

```
<A HREF="mailto:user@mailserver.ua">Отправити лист</A>
```

Наприкінці головної Web-сторінки в нас наведено відомості про права розроблювачів. Давайте перетворимо слово "Студенти" в поштове гіперпосилання. Html-Код , що створює її, буде виглядати так:

```
<A HREF="mailto:user@mailserver.ua">Студенти</A>
```

Додаткові можливості гіперпосилань

Мова HTML пропонує нам деякі додаткові можливості для створення гіперпосилань. Їх застосовують нечасто, але іноді вони корисні.

Насамперед, ми можемо вказати для гіперпосилання *"гарячу" клавішу*. Якщо відвідувач натисне цю клавішу, утримуючи натиснутою клавішу **<Alt>,** Браузер виконає перехід по даному гіперпосиланню.

Для вказівки "гарячої" клавіші передбачений необов'язковий атрибут **ACCESSKEY** тегу **<A>**. Значення цього атрибута — латинська буква, відповідна до потрібної клавіші:

```
<A HREF="http://www.somesite.ua/pages/page125.html"
ACCESSKEY="d">Сторінка №125</A>
```

Тут ми вказали для гіперпосилання "гарячу" клавішу **<D>.** І, щоб перейти по ній, відвідувачі буде досить натиснути комбінацію клавіш **<Alt>+<D>.**

На гіперпосиланнях можна клацати мишею — так робить більшість користувачів. Але по них також можна "подорожувати" за допомогою клавіатури. В цьому випадку говорять про *фокус введення* — ознаці, яке гіперпосилання буде обробляти натискання клавіш. Гіперпосилання, що має фокус введення, виділяється тонкою чорною штриховою рамкою.

- Якщо натиснути клавішу **<Enter>,** Браузер виконає перехід по гіперпосиланню, що має в цей момент фокус введення.
- Якщо натиснути клавішу **<Tab>,** Браузер перенесе фокус введення на наступне гіперпосилання.

- Якщо натиснути комбінацію клавіш **<Shift>+<Tab>,** Браузер перенесе фокус введення на попереднє гіперпосилання.

Порядок, у якому виконується перенос фокуса введення з одного гіперпосилання на інше при натисканні клавіш **<Tab>** або **<Shift>+<Tab>,** так і називається — *порядок обходу.* За замовчуванням він збігається з порядком, в якому гіперпосилання визначені в HTML-коді Web-сторінки. Але ми можемо вказати свій порядок обходу за допомогою атрибута **TABINDEX** тегу **<A>.** Його значення — ціле число від -32 767 до 32 767 — *номер в порядку обходу.*

- Якщо вказаний додатний номер, саме він буде визначати порядок обходу. Іншими словами, спочатку фокус введення отримає гіперпосилання з номером 1, потім — з номером 2, далі — з номером 3 і т.д.
- Якщо вказаний номер, що дорівнює нулю, обхід буде здійснюватися в порядку, в якому гіперпосилання визначене в HTML-коді Web-сторінки. Фактично нуль — значення атрибута тегу **TABINDEX** за замовчуванням.
- Якщо вказаний від'ємний номер, дане гіперпосилання взагалі виключається з порядку обходу. До нього неможливо буде добратися за допомогою клавіатури — можна буде тільки клацати мишею.

Розглянемо **приклад**:

```
<P><A HREF="page1.htm" TABINDEX="3">Сторінка 1</A></P>
<P><A HREF="page2.htm" TABINDEX="2">Сторінка 2</A></P>
<P><A HREF="page3.htm" TABINDEX="1">Сторінка 3</A></P>
```

Цей HTML-код створює три гіперпосилання із "зворотним" порядком обходу. Спочатку фокус введення отримає гіперпосилання "Сторінка 3", потім — "Сторінка 2" і наприкінці — "Сторінка 1".

ГРАФІЧНІ ГІПЕРПОСИЛАННЯ

Гіперпосилання може бути у вигляді не тільки фрагмента тексту, але й картинки або навіть являти собою фрагмент графічного зображення. Вивчимо *графічні гіперпосилання*.

Зображення-гіперпосилання

Мова HTML дозволяє використовувати як вміст гіперпосилання будь-який фрагмент будь-якого блокового елемента, в тому числі і графічне зображення, тобто створити *зображення-гіперпосилання.*

Для створення зображення гіперпосилання достатньо помістити всередину тегу **<A>** тег **<IMG>:**

```
<A HREF="http://www.w3.org"><IMG SRC="w3logo.gif"></A>
```

Цей HTML-код створює зображення-гіперпосилання, що вказує на Web-сайт організації W3C. А саме зображення - логотип цієї організації, який ми зберегли в файлі в тій же папці, де знаходиться файл поточної Web-сторінки.

```
<A HREF="mailto:user@mailserver.ua"><IMG SRC="email.gif"></A>
```

А цей HTML-код створює поштове зображення-гіперпосилання.

Правила виведення зображень-гіперпосилань Браузером:

- зображення-гіперпосилання оточується рамкою, що має відповідний до гіперпосилання колір: синій — для невідвіданого, темно-червоний — для відвіданого і т.д.;
- при наведенні курсору миші на зображення-гіперпосилання Web-Браузер змінює його форму на "перст, що вказує", як і у випадку текстового гіперпосилання.

Рамка навколо зображення-гіперпосилання найчастіше виглядає непрезентабельно, тому її звичайно видаляють, задавши відповідне представлення.

Зображення-карти

А ще HTML дозволяє перетворити в гіперпосилання частину графічного зображення. Більш того, ми можемо розбити зображення на частини, кожна з яких буде гіперпосиланням, що вказує на свою інтернет-адресу. Такі зображення називають зображеннями-*картами,* а її частини-гіперпосилання — *"гарячими" областями.*

За допомогою зображень-карт часто створюють заголовки Web-сайтів, фрагменти якого є гіперпосиланнями, що вказують на певну Web-сторінку.

Зображення-карту створюють в три етапи. Перший етап — створення самого зображення за допомогою добре нам знайомого тегу **<IMG>:**

```
<IMG SRC="map.gif">
```

Другий етап — створення *карти,* особливого елемента Web-сторінки, який описує набір "гарячих" областей зображення-карти. Сама карта на Web-сторінці ніяк не відображається.

Карту створюють за допомогою парного тегу **<MAP>**:

```
<MAP NAME="<ім'я карти>">
</MAP>
```

За допомогою обов'язкового атрибута **NAME** тегу **<MAP>** задається унікальне в межах Web-сторінки ім'я карти. Воно однозначно ідентифікує дану карту, може містити тільки латинські букви, цифри та знаки підкреслення і починатися повинне з букви:

```
<MAP NAME="samplemap"> </MAP>
```

Після створення карти слід прив'язати її до створеного на першому етапі зображення. Для цього ми застосуємо обов'язковий в цьому випадку атрибут **USEMAP** тегу **<IMG>.** Його значення — ім'я карты, що привязується до зображення, причому на початку цього імені обов'язково слід поставити символ **#** ("ґрати"). (В імені, заданому атрибутом **NAME** тегу **<MAP>**, символ **#** бути присутнім не повинен.)

```
<IMG SRC="map.gif" USEMAP="#samplemap">
```

На третьому етапі створюють описи сам*и*х "гарячих" областей в карті. Їх поміщають всередину відповідного тегу **<MAP>** і формують за допомогою одинарних тегів **<AREA>**:

```
<AREA [SHAPE="rect|circle|poly"] COORDS="<набір параметрів>"
 HREF="<інтернет-адреса гіперпосилання>"|NOHREF
 TARGET="<ціль гіперпосилання>">
```

Необов'язковий атрибут **SHAPE** задає форму "гарячої" області. Обов'язковий атрибут **COORDS** наводить координати, необхідні для побудови цієї області. Значення атрибута **SHAPE**:

- **"rect"** — прямокутна "гаряча" область. Атрибут **COORDS** в цьому випадку записується у вигляді **COORDS="*<X1>,<Y1>,<X2>,<Y2>*"**, де **X1** і **Y1** — координати верхнього лівого, а **X2** і **Y2** — правого нижнього кута прямокутника. До речі, якщо атрибут **SHAPE** відсутній, створюється саме прямокутна область;

- **"circle"** — кругла "гаряча" область. В цьому випадку атрибут **COORDS** має вигляд **COORDS="*<X центру>,<Y центру>,<радіус>*";**
- **"poly"** — "гаряча" область у вигляді багатокутника. Атрибут **COORDS** повинен мати вигляд **COORDS="<X1>,*<Y1>*,<X2>,*<Y2>*,<X3>,*<Y3>*..."**, де ***Xn*** **і** ***Yn*** координати відповідної вершини багатокутника.

Атрибут **HREF** задає інтернет-адресу гіперпосилання. Він може бути замінений атрибутом без значення **NOHREF**, що задає область, не зв'язану ні з якою інтернет-адресою. Це дозволяє створювати оригінальні зображення-карти, наприклад, карту у вигляді бублика, "дірка" якого нікуди не вказує.

Також знайомий нам атрибут **TARGET** задає мету гіперпосилання. (Звичайно, вказувати його має сенс тільки в тому випадку, якщо ми створюємо саме "гарячу" область, а не "дірку" з атрибутом **NOHREF**.)

Лістинг 7.2 містить повний HTML-код, що створює зображення-карту.

Лістинг 7.2

```
<IMG SRC="map.gif" USEMAP="#samplemap">
.....
<MAP NAME="s amplemap">
   <AREA SHAPE="circle" COORDS="50,50,30" HREF="page1.html">
   <AREA SHAPE="circle" COORDS="50,150,30"
   HREF="page2.html"> <AREA SHAPE="poly"
   COORDS="100,50,100,100,150,50,100,50" NOHREF>
   <AREA SHAPE="rect" COORDS="0,100,30,100"
   HREF="appendix.html" TARGET="_blank">
</MAP>
```

Тут ми створили дві круглі "гарячі" області, що вказують на Web-сторінки **page1.html** і **page2.html**, багатокутну область, що не посилає нікуди, і прямокутну область, що посилається на Web-сторінку **appendix.html**. Причому остання "гаряча" область при клацанні на ній відкриє Web-сторінку в новому вікні Браузера.

СМУГА НАВІГАЦІЇ

Гіперпосилання не завжди "ходять поодинці". Досить часто на Web-сторінках присутні цілі набори гіперпосилань, що посилаються на різні Web-сторінки даного Web-сайту. Такі набори називаються *смугами навігації.*

Смуга навігації може бути розташована по горизонталі, вгорі або внизу Web-сторінки, або по вертикалі, ліворуч або праворуч. Горизонтальну смугу навігації можна сформувати за допомогою звичайного абзацу або таблиці з одним рядком і потрібним числом комірок (кожне гіперпосилання розташовується в своїй комірці таблиці). Вертикальну смугу навігації звичайно формують за допомогою таблиці з однієї колонки і потрібного числа комірок (знову ж, кожне гіперпосилання розташовується в своїй комірці таблиці), набору абзаців (кожне гіперпосилання є окремим абзацем) або у вигляді списку (гіперпосилання є пунктом цього списку).

Гіперпосилання смуги навігації можуть бути текстовими або графічними. В останньому випадку практично завжди застосовують зображення-гіперпосилання. Найчастіше для зображень-гіперпосилань реалізують особливу поведінку, що заміняє зображення іншим при наведенні на відповідне гіперпосилання курсору миші.

Смугу гіперпосилань завжди виділяють, щоб привернути до неї увагу відвідувача. Її можуть виділити кольором тексту і фона, рамкою, збільшеним розміром шрифту або всім разом. Все це реалізується за допомогою завдання відповідного представлення Web-сторінки.

Давайте створимо на головній Web-сторінці нашого Web-сайту смугу навігації з гіперпосиланнями, що вказують на інші його Web-сторінки: присвячену CSS, що містить список прикладів і відомості про розроблювачів. Для простоти реалізуємо її у вигляді абзацу, що містить потрібні гіперпосилання. Помістимо її в самому верху Web-сторінки, перед заголовком (лістинг 7.3).

Лістинг 7.3

```
<BODY>
   <P><A HREF="css_index.htm">CSS</A> |
   <A HREF="samples_index.htm">Приклади</A> |
   <A HREF="about.htm"> Про розроблювачів</A></P>
   <H1>Довідник по HTML і CSS</H1>

   ..................
```

Як бачимо, така смуга навігації дуже проста — звичайний абзац із набором гіперпосилань, розділених символом | (вертикальна риса).

Якоря (внутрішні гіперпосилання)

Наприкінці розглянемо ще одну можливість, пропоновану нам мовою HTML і здатну сильно спростити відвідувачам читання довгих текстів. Хоча вона і не належить до гіперпосилань напряму, але діє разом з ними.

Це так звані *якорі* (anchors, внутрішні гіперпосилання). Вони не вказують на іншу Web-сторінку (файл, адресу електронної пошти), а позначають деякий фрагмент поточної Web-сторінки, щоб інше гіперпосилання могло на нього послатися. Так можна позначити окремі розділи довгого текстового документа, і відвідувач зможе "перескочити" до потрібного йому розділу, клацнувши гіперпосилання в змісті. Дуже зручно!

Якоря створюють за допомогою тегу **<A>**, як і гіперпосилання. Тільки в цьому випадку атрибут **HREF** в ньому бути присутнім не повинен. Замість нього в тег **<A>** поміщають обов'язковий тут атрибут **ID**, що задає унікальне в межах поточної Web-сторінки ім'я створюваного якоря. До нього пред'являються ті ж вимоги, що і до імені карти.

І ще. Ми вже знаємо, що тег **<A>** парний і у випадку гіперпосилання в нього поміщають текст (або зображення), який цим самим гіперпосиланням і стане. Коли створюють якір, в цей тег не поміщають нічого (так званий *порожній тег)*. Принаймні, так роблять найчастіше.

Лістинг 7.4 ілюструє приклад HTML-коду, що створює якір.

Лістинг 7.4

```
.....................
<P>Закінчення другого розділу...</P>
<A ID="chapter3"></A>
<P>Початок третього розділу...<P>
.....................
```

Тут ми помістили якір з ім'ям **chapter3** перед початком третього розділу документа.

Для того, щоб на цей якір послатися з іншої Web-сторінки, досить створити звичайне гіперпосилання, додавши в її інтернет-адресу ім'я потрібного нам якоря. Ім'я якоря ставлять в самий кінець інтернет-адреси і відокремлюють від нього символом **#** ("ґрати").

Припустимо, що Web-сторінка, що містить якір **chapter3**, зберігається у файлі **novel.htm**. Тоді, щоб послатися на цей якір з іншої Web-сторінки, ми створимо на останній таке гіперпосилання:

```
<A HREF="novel.htm#chapter3"> Розділ 3</A>
```

При клацанні на такому гіперпосиланні Браузер відкриє Web-сторінку **novel.htm** і прокрутить її у вікні так, щоб досягтися місця, де знаходиться якір **chapter3**.

Якщо ж нам потрібно послатися на якір з тієї ж Web-сторінки, де він знаходиться, то можна використовувати як інтернет-адресу тільки ім'я даного якоря, випередивши його символом "ґрат":

```
<A HREF="#chapter3">Розділ 3</A>
```

КОНТРОЛЬНІ ПИТАННЯ

1. Що таке «Засоби навігації» Web-сторінок? Якими вони бувають?
2. Створення текстових гіперпосилань.
3. Інтернет-адреси в WWW.
4. Поштові гіперпосилання.
5. «Гарячі» клавіші для створення гіперпосилань.
6. Фокус увведення і порядок обходу гіперпосилань.
7. Створення зображень-гіперпосилань.
8. Створення зображень-карт.
9. Що таке Смуга навігації?
10. Створення Якоря (внутрішнього гіперпосилання).

РОЗДІЛ 2. ПРЕДСТАВЛЕННЯ WEB-СТОРІНОК. КАСКАДНІ ТАБЛИЦІ СТИЛІВ

ТЕМА 8. ВВЕДЕННЯ В СТИЛІ CSS

***Основна мета*:** одержати поняття про стилі, таблиці стилів і використану для їхнього створення мову CSS; розібратися із правилами каскадності та з'ясувати, які стилі в яких випадках слід застосовувати.

.

Web-сторінки, створені за допомогою тегів HTML, можуть містити великий об'єм тексту, списки, таблицю, графічне зображення, аудіо- і відеоролик. Тільки от виглядають ці Web-сторінки якось непоказно. Одноманітний текст, схожі один на одного абзаци, таблиці без рамок, тужливе чорно-біле розцвічення...

За оформлення Web-сторінок і окремих їхніх елементів "відповідає" представлення. Саме представлення допоможе нам оформити абзаци, таблиці та гіперпосилання так, як хочемо ми, а не Браузер (точніше, його розроблювачі). Саме представлення допоможе нам зробити Web-сторінки привабливими.

Поняття про стилі CSS

Для створення представлення Web-сторінок використовується технологія *каскадних таблиць стилів* (Cascading Style Sheets, CSS), або просто *таблиць стилів.* Таблиця стилів містить набір правил *(стилів),* що описують оформлення самої Web-сторінки і окремих її фрагментів. Ці правила визначають колір тексту і вирівнювання абзацу, відступи між графічним зображенням та текстом, що його обтікає, наявність і параметри рамки в таблиці, колір фона Web-сторінки та багато чого іншого.

Кожний стиль повинен бути прив'язаний до відповідного елемента Web-сторінки (або самої Web-сторінки). Після прив'язки описувані вибраним стилем параметри починають застосовуватися до даного елемента. Прив'язка може бути явна, коли ми самі вказуємо, який стиль до якого елемента Web-сторінки прив'язаний, або неявна, коли стиль автоматично прив'язується до всіх елементів Web-сторінки, створеним за допомогою певного тегу.

Таблиця стилів може зберігатися прямо в HTML-коді Web-сторінки або в окремому файлі. Останній підхід більш відповідає концепції Web 2.0; вона вимагає, щоб вміст і представлення Web-сторінки були розділені. Крім того, окремі стилі можна помістити прямо в тег HTML, що створює елемент Web-сторінки; такий підхід використовується зараз досить рідко та, в основному, при експериментах із стилями.

Таблиці стилів пишуть особливою мовою, яка так і називається — CSS. Стандарт, що описує першу версію цієї мови (CSS 1), з'явився ще в 1996 році. В цей час широке підтримується і застосовується на практиці стандарт CSS 2 та розроблений і впроваджується CSS 3.

Створення стилів CSS

Звичайний формат визначення стилю CSS ілюструє лістинг 8.1.

Лістинг 8.1

```
<селектор> {
<атрибут стилю 1>: <значення 1>;
<атрибут стилю 2>: <значення 2>;
.....................................
<атрибут стилю n-1>: <значення n-1>;
<атрибут стилю n>: <значення n>
}
```

От основні **правила створення стилю**.

- Визначення стилю включає селектор і список атрибутів стилю з їхніми значеннями.
- *Селектор* використовується для прив'язки стилю до елемента Web-сторінки, на який він повинен розповсюджувати свою дію. Фактично селектор однозначно ідентифікує даний стиль.
- За селектором, через пробіл, вказують список атрибутів стилю і їх значень, розміщений в фігурних дужках.
- *Атрибут стилю* (не плутати з атрибутом тегу!) є одним з параметрів елемента Web-сторінки: колір шрифту, вирівнювання тексту, величина відступу, товщину рамки та ін. *Значення* атрибута стилю вказують після нього через символ **:** (двокрапка). В деяких випадках значення атрибута стилю беруть в лапки.
- Пари **<атрибут стилю>:<значення>** відокремлюють друг від друга символом **;** (крапка з комою).
- Між останньою парою **<атрибут стилю>:<значення>** і закриваючою фігурною дужкою символ **;** не ставлять, інакше деякі Браузери можуть неправильно обробити визначення стилю.
- Визначення різних стилів розділяють пробілами або перенесеннями рядків.
- Всередині селекторів та імен стилів не повинні бути присутнім пробіли і перенесення рядка. Що стосується пробілів і перенесень рядків, поставлених в інших місцях визначення стилю, то Браузер їх ігнорує. Тому ми можемо форматувати CSS-код для зручності його читання, як робили це з HTML-кодом.

Стиль перевизначення тегу

Розглянемо **приклад** стилю:

```
P{color #0000FF}
```

Розберемо його по частинам.

- **P** — це селектор. Він являє собою ім'я тегу **<P>.**
- **color** — це атрибут стилю. Він задає колір тексту.
- **#0000FF** — це значення атрибута стилю **color**. Воно задає код синього кольору, записаний в форматі RGB.

Коли Браузер считує описаний стиль, він автоматично застосує його до всіх абзаців Web-сторінки (тегам **<P>**). Це, до речі, типовий приклад неявної прив'язки стилю.

Стиль, який ми розглянули, називається ***стилем перевизначення тегу.*** Як селектор тут вказане ім'я перевизначеного цим стилем HTML-тегу без символів < і >. Селектор можна набирати як прописними, так і малими літерами.

Розглянемо ще **приклади.**

От стиль перевизначення тегу **<EM>**:

```
EM{color:#00FF00;
    font-weight:bold}
```

Будь-який текст, поміщений у тег **<EM>,** Браузер виведе зеленим напівжирним шрифтом. Атрибут стилю **font-weight** задає ступінь "жирності" шрифту, а його значення **bold** — напівжирний шрифт.

А це стиль перевизначення тегу **<BODY>:**

```
BODY{background-color:#000000;
     color:#FFFFFF}
```

Він задає для всієї Web-сторінки білий колір тексту (RGB-код **#FFFFFF**) і чорний колір фона (RGB-код **#000000**). (Атрибут стилю **background-color** задає колір фона, атрибут стилю **color** задає колір тексту).

Стильовий клас

А тепер розглянемо зовсім інший стиль:

```
.redtext{color:#FF0000}
```

Він задає червоний колір тексту (RGB-код **#FF0000**). Але як селектор використовується явно не ім'я тегу — HTML-тегу **<REDTEXT>** не існує.

Це інший різновид стилю CSS — ***стильовий клас.*** Він може бути прив'язаний до будь-якого тегу. Як селектор тут вказують *ім'я стильового класу,* яке його однозначно ідентифікує. Ім'я стильового класу повинно складатися з букв латинського алфавіту, цифр і дефісів, причому починатися повинне з букви. В визначенні стильового класу його ім'я обов'язкове випереджається символом точки — це ознака того, що визначається саме стильовий клас.

Стильовий клас вимагає явної прив'язки до тегу. Для цього служить атрибут **CLASS,** підтримуваний практично всіма тегами. Як значення цього атрибута вказують ім'я потрібного стильового класу без символу точки:

```
<P CLASS="redtext">Увага!</P>
```

Тут ми прив'язали тільки що створений стильовий клас **redtext** до абзацу **"Увага!".** В результаті цей абзац буде набраний червоним шрифтом.

Лістинг 8.2

```
.attention{color:#FF0000;
           font-style:italic}
......
<P><STRONG CIASS="attention">Стильовий клас вимагає
явної прив'язки атрибутом тегу CLASS!</STRONG></P>
```

В лістингу 8.2 ми створили стильовий клас **attention**, який задає червоний колір і курсивне накреслення шрифту. (Атрибут стилю **font-style** задає накреслення шрифту, а його значення **italic** саме робить шрифт курсивним.) Потім ми прив'язали його до тегу **<STRONG>**. В результаті вміст цього тегу буде набраний напівжирним курсивним шрифтом червоного кольору; особливу важливість і пов'язану з ним "напівжирність" тексту задає тег **<STRONG>**, а курсивне накреслення і червоний колір — стильовий клас **attention.**

Як значення атрибута **CLASS** можна вказати відразу кілька імен стильових класів, розділивши їх пробілами. В такому випадку дія стильових класів як би складається.

Лістинг 8.3

```
.attention{color:#FF0000;
           font-style:italic}
.bigtext{font-size:large}
......
<P><STRONG CLASS="attention bigtext">Стильовий клас
вимагає явної прив'язки атрибутом тегу
CLASS!</STRONG></P>
```

В прикладі з лістингу 8.3 ми прив'язали до тегу **<STRONG>** відразу два стильові класи: **attention** і **bigtext**. В результаті вміст цього тегу буде виведений напівжирним курсивним шрифтом червоного кольору і великого розміру. (Атрибут стилю **font-size** вказує розмір шрифту, а його значення **large** — великий розмір, порівнянний з розміром шрифту, яким виводяться заголовки першого рівня).

Іменований стиль

Іменований стиль багато в чому схожий на стильовий клас. Селектором цього стилю також є ім'я, яке його однозначно ідентифікує, і привязується він до тегу також явно. А далі починаються відмінності.

- У визначенні іменованого стилю перед його іменем ставлять символ **#** ("ґрати"). Він повідомляє Браузеру, що перед ним іменований стиль.
- Прив'язку іменованого стилю до тегу реалізують через атрибут **ID**, також підтримуваний практично всіма тегами. Як значення цього атрибута вказують ім'я потрібного іменованого стилю, уже без символу **#.**
- Значення атрибута тегу **ID** повинно бути унікальним в межах Web-сторінки. Інакше кажучи, в HTML-коді Web-сторінки може бути присутнім тільки один тег із заданим значенням атрибута **ID.** Тому іменовані стилі використовують, якщо який-небудь стиль слід прив'язати до одному-єдиного елемента Web-сторінки.

В **прикладі**

```
#bigtext{font-size:large}
.........
<P ID="bigtext">Це великий текст</P>
```

абзац "Це великий текст." буде набраний великим шрифтом.

Комбінований стиль

У всіх розглянуті нами різновидах стилів був один селектор, за допомогою якого і виконувалася прив'язка. Однак CSS дозволяє створювати стилі з декількома селекторами — так звані *комбіновані стилі.* Такі стилі служать для прив'язки до тегів, що задовольняють відразу декільком умовам. Так, ми можемо вказати, що комбінований стиль повинен бути прив'язаний до

тегу, вкладеного в інший тег, або до тегу, для якого вказаний певний стильовий клас.

Правила, які встановлені стандартом CSS при написанні **селекторів у комбінованому стилі.**

- Селекторами можуть виступати імена тегів, імена стильових класів і імена іменованих стилів.
- Селектори перераховують зліва направо і позначають рівень вкладеності відповідних тегів, який збільшується при руху зліва направо: теги, вказані правіше, повинні бути вкладені в теги, що знаходяться лівіше.
- Якщо ім'я тегу скомбіноване з ім'ям стильового класу або іменованого стилю, виходить, для даного тегу повинно бути вказане це ім'я стильового класу або іменованого стилю.
- Селектори розділяють пробілами.
- Стиль прив'язують до тегу, позначеного самим правим селектором.

Щоб зрозуміти ці правила, розглянемо кілька **прикладів**. Почнемо з найпростішого комбінованого стилю:

```
P EM{color:#0000FF}
```

- Як селектори використані імена тегів **<P>** і **<EM>.**
- Спочатку йде тег **<P>,** за ним — тег **<EM>.** Виходить, тег **<EM>** повинен бути вкладений в тег **<P>.**
- Стиль буде прив'язаний до тегу **<EM>.**

```
<P><EM>Цей текст</EM> стане синім</P>
<P>А цей не стане</P>
<P><STRONG>Цей</STRONG> - теж.</P>
```

Тут слова "Цей текст" будуть набрані синім шрифтом.

От ще один **приклад** комбінованого стилю:

```
P.mini{color:#FF0000;
       font-size:smaller}
```

Ім'я тегу **<P>** скомбіноване з ім'ям стильового класу **mini**. Виходить, даний стиль буде застосований до будь-якого тегу **<P>,** для якого вказано ім'я стильового класу **mini**. (Значення **smaller** атрибута стилю **font-size** задає зменшений розмір шрифту.)

```
<P CLASS="mini">Маленький червоний текстик.</P>
```

І останній **приклад** комбінованого стилю:

```
P.sel <STRONG>{color:#FF0000}
```

Цей стиль буде застосований до тегу **<STRONG>, що** знаходиться всередині тегу **<P>,** до якого прив'язаний стильовий клас **sel.**

```
<P CLASS="mini"><STRONG>Цей<STRONG> текст стане червоним</P>
```

В даному прикладі слово **"Цей"** буде виділено червоним кольором.

Стандарт CSS дозволяє визначити відразу кілька однакових стилів, перерахувавши їх селектори через кому:

```
H1, .redtext, P EM <STRONG>{color: #FF0000 }
```

Тут ми створили відразу три однакові стилі: стиль перевизначення тегу **<H1>**, стильовий клас **redtext** і комбінований стиль **P EM**. Всі вони задають червоний колір шрифту.

Вбудований стиль

Всі чотири розглянуті нами різновиди стилів CSS можуть бути присутніми тільки в таблицях стилів. Якщо вказати їх в HTML-коді Web-сторінки, вони, швидше за все, будуть зігноровані.

Стилі останнього — п'ятого — різновиду вказують прямо в HTML-коді Web-сторінки, у відповідному тегу. Це ***вбудовані стилі.*** В сімействі стилів вони стоять особняком.

- Вони не мають селектора, тому що ставляться прямо в потрібний тег. Селектор в цьому випадку просто не потрібний.
- В них відсутні фігурні дужки, оскільки немає потреби відокремлювати список атрибутів стилю від селектора, якого немає.
- Вбудований стиль може бути прив'язаний тільки до одного тега — того, в якому він знаходиться.

Визначення вбудованого стилю вказують як значення атрибута **STYLE** потрібного тегу, який підтримується практично всіма тегами:

```
<P STYLE="font-size:smaller; font-style:italic">Маленький
курсивчик.</P>
```

Раніше ми згадали, що в деяких випадках значення атрибута стилю потрібно брати в лапки. Але у визначенні вбудованого стилю замість лапок використовуються апострофи.

Вбудовані стилі застосовуються зараз досить рідко, тому що порушують вимогу концепції Web 2.0 розділяти вміст і представлення Web-сторінок. В основному їх застосовують для прив'язки стилів до одного-єдиного елемента Web-сторінки (дуже рідко) і під час експериментів із стилями.

Таблиці стилів

Ми розглянули п'ять різновидів стилів CSS. Чотири з них — стильові класи, стилі перевизначення тегу, іменовані та комбіновані стилі — можуть бути присутнім тільки в таблицях стилів.

Таблиці стилів, залежно від місця їх зберігання, розділяються на два види.

Зовнішні таблиці стилів

Зовнішні таблиці стилів зберігаються окремо від Web-сторінок, в файлах з розширенням **css**. Вони містять CSS-код визначень стилів.

Лістинг 8.4 ілюструє приклад зовнішньої таблиці стилів.

Лістинг 8.4

```
.redtext {color:#FF0000}
#bigtext {font-size:large}
EM {color:#00FF00;
    font-weight:bold}
P EM {color:#0000FF}
```

Як бачимо, тут визначено чотири стилі.

Якщо зовнішня таблиця стилів зберігається окремо від Web-сторінки, виходить, потрібно якось прив'язати її до Web-сторінці. Для цього призначений одинарний метатег **<LINK>**, який поміщається в секцію заголовка відповідної Web-сторінки. От формат його написання:

```
<LINK REL="stylesheet" HREF="<інтернет-адреса файлу таблиці стилів>"
TYPE="text/css">
```

Інтернет-адресу файлу таблиці стилів записують як значення атрибута **HREF** цього тегу.

Інші атрибути тегу **<LINK>** для нас несуттєві. Атрибут **REL** вказує, чим є файл, на який посилається тег **<LINK>**, для поточної Web-сторінки; його значення **"stylesheet"** говорить, що цей файл — зовнішня таблиця стилів. А атрибут **TYPE** вказує тип MIME файлу, на який посилається даний тег; зовнішня таблиця стилів має тип MIME **text/css.**

Лістинг 8.5

```
<HEAD>
    .........
    <LINK          REL="stylesheet"          HREF="main.css"
    TYPI="text/css">
    .........
</HEAD>
```

В прикладі з лістингу 8.5 ми прив'язали зовнішню таблицю стилів, що зберігається в файлі **main.css**, до поточної Web-сторінки.

Перевага зовнішніх таблиць стилів в тому, що їх можна прив'язати відразу до кількох Web-сторінок. Недолік всього один, та і той несуттєвий, — зовнішня таблиця стилів зберігається в окремому файлі, так що є ймовірність його "загубити".

Внутрішні таблиці стилів

Внутрішня таблиця стилів (лістинг 8.6) записується прямо в HTML-код Web-сторінки. Її розміщають в парному тегу <STYLE> і поміщають в секцію заголовка. В іншому вона не відрізняється від її зовнішньої "колеги".

Лістинг 8.6

```
<HEAD>
    .........
    <STYLE>
        .redtext {color:#FF0000}
        #bigtext {font-size:large}
        EM       {color:#00FF00;
                    font-weight:bold}
        P EM     {color:#0000FF}
```

```
    </STYLE>
    .........
</HEAD>
```

Перевага внутрішньої таблиці стилів у тому, що вона є невід'ємною частиною Web-сторінки і ніколи не "загубиться". Недоліків два. По-перше, стилі, визначені у внутрішній таблиці стилів, застосовуються тільки до тієї Web-сторінки, в якій ця таблиця стилів знаходиться. По-друге, внутрішня таблиця стилів не відповідає концепції Web 2.0, що вимагає відокремлювати вміст Web-сторінки від її представлення.

В одній і тій же We-сторінці можуть бути присутнім відразу кілька таблиць стилів: декілька зовнішніх і внутрішня (лістинг 8.7).

Лістинг 8.7

```
<HEAD>
   .........
   <LINK REL="stylesheet" HREF="sty1es1.css" TYPE="text/css">
   <LINK REL="sty1esheet" HREF="sty1es2.css" TYPE="text/css">
   .........
   <STYLE>
   .........
   </STYLE>
   .........
</HEAD>
```

В такому випадку дія всіх цих таблиць стилів сумується. А за якими правилами — ми зараз з'ясуємо.

Пріоритет стилів і правила каскадності

Як ми вже з'ясували, на той самий елемент Web-сторінки можуть діяти відразу кілька стилів. Це можуть бути стилі різних видів (стиль перевизначення тегу, стильовий клас, комбінований стиль, вбудований стиль) або визначені в різних таблицях стилів (зовнішніх і внутрішньої). Таке зустрічається досить часто, особливо на Web-сторінках із складним оформленням.

Але як Браузер визначає, який саме стиль застосувати до того або іншому елементу Web-сторінки? Ми вже знаємо, що в таких випадках дія стилів як би сумується. Але за якими правилами?

Припустимо, що ми створили зовнішню таблицю стилів (лістинг 8.8).

Лістинг 8.8

```
.redtext {color:#FF0000}
#bigtext {font-size:large}
EM       {color:#00FF00;
          font-weight:bold}
```

Після цього ми виготовили Web-сторінку, що містить внутрішню таблицю стилів (лістинг 8.9).

Лістинг 8.9

```
<STYLE>
.redtext {color:#0000FF}
EM {font-size:smaller}
</STYLE>
```

А в самій Web-сторінці написали от такий фрагмент HTML-коду:

```
<P CLASS="redtext"Це червоний текст</P>
<P ID="bigtext" STYLE="color:#FFFF00">Це великий текст</P>
<P><EM>Це курсив</EM</P>
```

Ми бачимо, що на елементи цієї Web-сторінки діють відразу по декілька стилів. Так, в другому рядку коду до тегу **<P>** прив'язані й іменований стиль **bigtext**, і вбудований стиль. Але цього мало — і зовнішня, і внутрішня таблиці стилів містять визначення двох однакових стилів — стильового класу **redtext** і стилю перевизначення тегу **<EM>!**

Так що ж ми отримаємо в результаті?

Розглянемо спочатку останній рядок приведеного HTML-коду з тегом **<EM>**. Спочатку Браузер завантажить, обробить і збереже в пам'яті зовнішню таблицю стилів. Потім він обробить внутрішню таблицю стилів і додасть всі визначення стилів, що містяться в ній, до тих стилів, що вже зберігаються в зовнішній таблиці стилів. Це значить, що стилі перевизначення тегу **<EM>**, задані в різних таблицях стилів, будуть сумовані, і результуючий стиль, написаний мовою CSS, стане таким:

```
EM {color:#00FF00;
    font-size:smaller;
```

```
font-weight:bold}
```

Саме його і застосує Браузер до всіх тегів **<EM>**, що присутні на Web-сторінці.

Другий рядок HTML-коду, що містить тег із вбудованим стилем, буде оброблено так само. Браузер додасть до зчитаного із зовнішньої таблиці стилів визначенню іменованого стиля **bigtext** визначення вбудованого стилю. Результуючий стиль, якщо записати його мовою CSS, буде таким:

```
#bigtext {color:#FFFF00;
          font-size:large}
```

І, нарешті, найбільш важка задача — перший рядок HTML-коду. Оскільки обидва визначення стильового класу **redtext** задають той самий параметр — колір тексту (атрибут стилю **color**) — Браузер зробить так. Він відмінить значення цього атрибута, задане в стилі із зовнішньої таблиці стилів, і замінить його тим, що задане в стилі із внутрішньої таблиці стилів. Оскільки, з його погляду і з погляду стандартів CSS, внутрішня таблиця стилів пріорітетна. І тоді результуючий стиль буде таким:

```
.redtext { color: #0000FF }
```

Тут ми зіткнулися з тим, що стилі різних видів і задані в різних таблицях стилів мають різний *пріоритет*. Браузер керується цим пріоритетом, коли формує в своїй пам'яті остаточні визначення стилів.

Тепер ознайомимося із правилами, що описують поведінку Браузера при формуванні остаточних стилів. Їхнім ще називають **правилами каскадності.**

- Зовнішня таблиця стилів, посилання на яку (тег **<LINK>**) зустрічається в HTML-коді сторінки пізніше, має пріоритет перед тою, посилання на яку зустрілася раніше.
- Внутрішня таблиця стилів має пріоритет перед зовнішніми.
- Вбудовані стилі мають пріоритет перед будь-якими стилями, заданими в таблицях стилів.
- Більш конкретні стилі мають пріоритет перед менш конкретними. Це значить, наприклад, що стильовий клас має пріоритет перед стилем перевизначення тегу, оскільки стильовий клас прив’язується до конкретних тегів. Точно так само комбінований стиль має пріоритет перед стильовим класом.
- Якщо до тегу прив'язано кілька стильових класів, то ті, що вказані правіше, мають пріоритет перед вказаними левіше.

Важливі атрибути стилів

А тепер уявимо собі наступну ситуацію. Припустимо, ми створили стилі, приведені в лістингу 8.10.

Лістинг 8.10

```
.redtext {color:#FF0000;
          font-weight:normal}
EM       {color:#00FF00;
          font-weight:bold}
```

Значення **normal** атрибута стилю **font-weight** задає звичайну "жирність" шрифту, тобто простий, світлий шрифт.

Далі ми помістили на Web-сторінку от такий абзац:

```
<P><EM CLASS="redtext">Це курсив</ЇM></P>
```

Правила каскадності ми вже розглянули, так що можна відразу сказати, що вийде в результаті. Текст цього абзацу буде виведений звичайним шрифтом червоного кольору.

Але припустимо, начебто нам потрібно, щоб весь текст, виділений тегом **<EM>**, обов’язково виводився напівжирним шрифтом! Що робити? Створювати інший стильовий клас, спеціально для такого випадку?

Зовсім не обов'язково. Стандарт CSS надає нам чудову можливість перетворити окремі атрибути стилю у визначенні стилю у *важливі*. Параметри, що задаються важливими атрибутами стилю, будуть мати пріоритет над усіма аналогічними атрибутами стилю, заданими в інших стилях, навіть більш конкретних. Фактично в такий спосіб ми порушимо правила каскадності стандартними засобами CSS.

Зверніть увагу, що важливими можна зробити тільки окремі атрибути стилю у визначенні стилю. Атрибути стилю, не оголошені важливими, як і раніше будуть підкорятися правилам каскадності.

Щоб зробити атрибут стилю важливим, досить після його значення через пробіл поставити слово **!important** (пишеться разом, без пробілів між знак оклику і словом "**important**").

```
EM {color:#00FF00;
    font-weight:bold !important}
```

Тепер текст, виділений тегом **<EM>**, завжди буде виводитися напівжирним шрифтом, навіть якщо даний параметр перевизначений у більш конкретному стилі.

Які стилі в яких випадках застосовувати

Вдало підібраний набір стилів — результат довгих експериментів. Нам доведеться неабияк повозитися, перш ніж ми його отримаємо. Але кілька правил, приведених далі, допоможуть нам отримати його помітно швидше.

Насамперед, слід виділити всі правила оформлення, які повинні бути застосовані до всіх Web-сторінок і їх елементів. До таких правил відносяться параметри шрифту звичайних абзаців і заголовків, колір фона Web-сторінки, вирівнювання тексту, величини відступів і параметри рамки звичайних таблиць та ін. Загалом, те, що визначає вигляд всіх Web-сторінок.

Ці правила перетворюються в загальні стилі, звичайно стилі перевизначення тегів. Їх поміщають в одну зовнішню таблицю стилів і прив'язують її до всіх Web-сторінок Web-сайту. Така таблиця стилів називається ***глобальною.***

Далі виділяють правила оформлення, які повинні застосовуватися до деяких елементів Web-сторінок. Це можуть бути параметри шрифту якихось вибраних абзаців (наприклад, попереджень про щось), їхніх фрагментів, параметри гіперпосилань, що входять у смугу навігації, та ін. Дані правила формують так, щоб доповнювати отримані раніше, але не перекривати їх повністю, — це дозволить скоротити розмір CSS-коду таблиць стилів.

Так ми отримаємо більш конкретні стилі, що доповнюють створені раніше. Їх ми оформимо у вигляді стильових класів і комбінованих стилів і помістимо все в ту ж глобальну таблицю стилів. Також можна помістити їх в окрему таблицю стилів *(вторинну),* яку прив'язати тільки до тих Web-сторінок, де вони використовуються; цей підхід виправданий тільки якщо таких стилів занадто багато, і немає резону занадто збільшувати глобальну таблицю стилів.

На наступному етапі ми виділимо правила, застосовувані тільки до одного елемента Web-сторінок. Це можуть бути параметри смуги навігації, заголовка Web-сайту, різних контейнерів, що визначають дизайн Web-страниці). Вони також повинні доповнювати правила, отримані на попередніх етапах, але не заміняти їх повністю.

Ці ще більш конкретні стилі ми оформимо у вигляді іменованих стилів — в цьому випадку це буде найкращим вибором. І помістимо їх у глобальну таблицю стилів. Якщо їх занадто багато, ми також можемо помістити їх у вторинну таблицю стилів і прив'язати її до всіх Web-сторінок.

Останній етап — виділення правил, застосовуваних до зовсім невеликої кількості елементів, які присутні тільки на окремих Web-сторінках, а то і взагалі на одній з них. До таких правил можна віднести параметри дуже специфічних абзаців, окремих ілюстрацій і таблиць. Пам’ятаємо, що ці правила, по можливості, не повинні повністю перекривати отримані раніше.

Так ми виділимо самі конкретні стилі. Їх можна оформити у вигляді стильових класів або іменованих стилів. Поміщатися вони можуть у глобальну таблицю стилів або у вторинну таблицю стилів, прив'язану тільки до тих Web-сторінок, де повинні використовуватися.

Що стосується внутрішніх таблиць стилів і вбудованих стилів, то в готових Web-сторінках від них рекомендується відмовитися. Концепція Web 2.0 настійно рекомендує розділяти вміст і представлення Web-Сторінок. А внутрішні таблиці стилів і вбудовані стилі порушують дане правило.

Зараз внутрішні таблиці стилів і вбудовані стилі застосовують, в основному, під час експериментів, щоб з'ясувати, як той або інший стиль впливає на відображення елементів Web-сторінки. По завершенню експериментів готові стилі переносять у зовнішні таблиці стилів і відповідним чином оформляють.

Коментарі CSS

Коментарі — особливі фрагментах HTML-коду, які не обробляються Браузером і служать для того, щоб Web-дизайнер зміг залишити якісь замітки для себе або своїх колег. Для цього мова HTML використовує спеціальний тег.

Створювати коментарі дозволяє і мова CSS. Більше того, з його допомогою ми можемо формувати коментарі двох видів.

Коментар, що складається з одного рядка, створюють за допомогою символу **/** (слеш) на самому початку рядка, який стане коментарем:

```
/Це коментар
P {color:#0000FF }
```

Однорядковий коментар починається із символу **/** і закінчується кінцем рядка.

Коментар, що складається з довільного числа рядків, створюють за допомогою послідовностей символів **/*** і ***/**. Між ними поміщають рядки, які стануть коментарем (лістинг 8.11).

Лістинг 8.11

```
/*
Це коментар, що складається з декількох рядків.
*/
P {color:#0000FF }
```

Багаторядковий коментар починається з послідовності символів **/*** і закінчується послідовністю ***/.**

КОНТРОЛЬНІ ПИТАННЯ

1. Для чого потрібні стилі CSS?
2. Перелічіть способи створення стилів.
3. Розкажіть про стиль перевизначення тегу.
4. Розкажіть про стильові класи.
5. Розкажіть про іменований стиль.
6. Розкажіть про комбіновані стилі.
7. Розкажіть про вбудовані стилі.
8. Що таке зовнішні і внутрішні таблиці стилів?
9. Пріоритет стилів.
10. Що таке правила каскадності стилів?
11. Що таке Важливі атрибути стилів?
12. Які стилі в яких випадках застосовувати?
13. Як записуються коментарі CSS.

ТЕМА 9. ПАРАМЕТРИ ШРИФТУ І ФОНА. КОНТЕЙНЕРИ

***Основна мета*:** навчитися оформляти текст і фон, використовуючи особливі атрибути стилю; почати створювати представлення для своїх Web-сторінок.

.

ПАРАМЕТРИ ШРИФТУ І ФОНА. КОНТЕЙНЕРИ

Параметри шрифту

Атрибут стилю ***font-family*** *- ім'я шрифту*

Атрибут стилю ***font-size*** *- розмір шрифту*

Атрибут стилю ***color*** *- колір тексту*

Атрибут стилю ***opacity*** *- ступінь напівпрозорості*

Атрибут стилю ***font-weight*** *- "жирність" шрифту:*

Атрибут ***font-style*** *- накреслення шрифту*

Атрибут стилю ***text-decoration*** *- підкреслення або закреслювання тексту*

Атрибут стилю ***font-variant*** *малі прописні букви шрифту*

Атрибут стилю ***text-transform*** *- змінити регістр символів тексту:*

Атрибут стилю ***line-height*** *задає висоту рядка тексту:*

Атрибут стилю ***letter-spacing*** *- додаткова відстань між символами тексту*

Атрибут стилю ***word-spacing*** *- додаткова відстань між словами тексту*

Атрибут стилю ***font*** *- одночасно відразу кілька параметрів* шрифту

Параметри, що управляють розривом рядків

Атрибут стилю ***white-space***

Атрибут стилю ***word-wrap*** *- місця розриву тексту*

Параметри вертикального вирівнювання

Параметри тіні в тексту

Параметри фона

Атрибут стилю ***background-color*** *- для завдання кольору фона*

Атрибут стилю ***background-image*** *для завдання фонового зображення*

Атрибут стилю ***background-repeat*** *для повтору фонового зображення*

Атрибут стилю ***background-position*** *для вказівки позиції фонового зображення*

Атрибут стилю ***background-attachment*** *управляє фіксацією фона*

Контейнери. Вбудовані контейнери

Представлення для нашого Web-сайту, частина 1

КОНТРОЛЬНІ ПИТАННЯ

Ми познайомилися із стилями і таблицями стилів CSS, за допомогою яких створюється представлення Web-сторінок: чотири різновиди стилів і два різновиди таблиць стилів.

Ми почнемо вивчати можливості мови CSS. Вивчимо атрибути стилю, за допомогою яких задають параметри абзаців, в тому числі заголовків, списків, тегів адреси, великих цитат - блокових елементів, призначених для структурування тексту. Розглянемо атрибути стилю, що задають специфічні параметри списків і їх пунктів (параметри маркерів і нумерації).

ПАРАМЕТРИ ШРИФТУ І ФОНА. КОНТЕЙНЕРИ

Параметри шрифту

Почнемо з атрибутів стилю, що задають параметри шрифту, яким набраний текст.

Атрибут стилю ***font-family*** *- ім'я шрифту*

Атрибут стилю **font-family** задає ім'я шрифту, яким буде виведений текст:

```
font-family:<список імен шрифтів, розділених комами>|inherit
```

Імена шрифтів задаються у вигляді їх назв, наприклад, **Arial** або **Times New Roman**. Якщо ім'я шрифту містить пробіли, його потрібно взяти в лапки:

```
P {font-family:Arial}
H1 {font-family:"Times New Roman"}
```

Якщо даний атрибут стилю присутній у вбудованому стилі, лапки заміняють апострофами:

```
<P STYLE="font-family:'Times New Roman'">
```

Якщо вказаний нами шрифт присутній на комп'ютері відвідувача, Браузер його використовує. Якщо ж такого шрифту нема, то текст виводиться шрифтом, заданим в налаштуваннях за замовчуванням. І наша Web-сторінка, можливо, буде виглядати не так, як ми задумували. (Втім, шрифти **Arial** і **Times New Roman** присутні на будь-якому комп'ютері, що працює під управлінням Windows.)

Можна вказати кілька найменувань шрифтів через кому:

```
P {font-family:Verdana,Arial}
```

Тоді Браузер спочатку буде шукати перший із вказаних шрифтів, у випадку невдалого пошуку — другий, потім третій і т.д.

Замість імені конкретного шрифту можна задати ім'я одного із сімейств шрифтів, що представляють цілі набори аналогічних шрифтів. Таких сімейств п'ять:

- **serif** (шрифти із зарубками),
- **sans-serif** (шрифти без зарубок),
- **cursive** (шрифти, що імітують рукописний текст),
- **fantasy** (декоративні шрифти) і
- **monospace** (моноширинні шрифти):

```
H2 {font-family:Verdana,Arial,sans-serif}
```

Особливе значення **inherit** вказує, що текст даного елемента Web-сторінки повинен бути набраний тим же шрифтом, що і текст батьківського елемента. Говорять, що в цьому випадку елемент Web-сторінки "успадковує" шрифт від батьківського елемента. Це, до речі, значення атрибута стилю **font-family** за замовчуванням.

Атрибут стилю `font-size` - розмір шрифту

Атрибут стилю **font-size** визначає розмір шрифту:

```
font-size:<розмір>|xx-small|x-small|small|medium|large|x-
large|
xx-large|larger|smaller|inherit
```

Розмір шрифту можна задати в абсолютних і відносних величинах. Для цього використовується одна з одиниць виміру, підтримувана CSS (табл. 9.1).

Табл. 9.1. **Одиниці виміру розміру, підтримувані стандартом CSS**

Назва	**Позначення в CSS**
Пікселі	**px**
Пункти	**pt**
Дюйми	**in**
Сантиметри	**cm**
Міліметри	**mm**
Піки	**pc**
Розмір букви "**m**" поточного шрифту	**em**
Розмір букви "**x**" поточного шрифту	**ex**
Відсотки від розміру шрифту батьківського елемента	**%**

Позначення вибраної одиниці виміру вказують після самого значення:

```
P {font-size:10pt}
STRONG {font-size:1cm}
EM {font-size:150%}
```

Відзначимо, що всі приведені в таблиці 9.1 одиниці виміру підходять для завдання значень інших атрибутів стилів CSS.

Крім числових, атрибут **font-size** може приймати і символьні значення. Так, значення від **xx-small** до **xx-large** задають сім визначених розмірів шрифту, від самого маленького до найбільшого. А значення **larger** і **smaller** визначають наступний розмір шрифту, відповідно, по зростанню і спаданню. Наприклад, якщо для батьківського елемента визначений шрифт розміру **medium**, то значення **larger** встановить для поточного елемента розмір шрифту **large**.

Значення **inherit** вказує, що даний елемент Web-сторінки повинен мати той же розмір шрифту, що і батьківський елемент. Це значення атрибута стилю **font-size** за замовчуванням.

Атрибут стилю `color` - колір тексту

Атрибут стилю **color** задає колір тексту:

```
color:<колір>|inherit
```

Колір можна задати так званим RGB-кодом (*Red, Green, Blue* — червоний, зелений, синій). Він записується в форматі

#<частка червоного кольору><частка зеленого кольору><частка синього кольору>,

де частки всіх кольорів вказані у вигляді шістнадцяткових чисел від **00** до **FF.**

Задамо для тексту червоний колір:

```
H1 {color:#FF0000}
```

А тепер сірий колір:

```
ADDRESS {color:#CCCCCC}
```

Крім того, CSS дозволяє задавати кольори по іменам. Так, значення **black** відповідає чорному кольору, **white** — білому, **red** — червоному, **green** — зеленому, а **blue** — синьому.

Приклад:

```
H1 {color:red}
```

В табл. 9.2 приведені деякі кольори тексту.

Табл. 9.2. **Кольори тексту**

Колір			**RRGGBB**
білий	white		**FFFFFF**
чорний	black		**000000**
червоний	red		**FF0000**
темно-бордовий	maroon		**800000**
оливковий	olive		**808000**
зелений	green		**00FF00**
бірюзовий	azure		**00FFFF**
синьо-зелений	teal		**008080**
синій	blue		**0000FF**
темно-синій	navy		**000080**
голубий	aqva		**00FFFF**
сірий	gray		**CCCCCC** **A0A0A0** **808080**
сріблястий	silver		**C0C0C0**
фуксія	fuchsia		**FF00FF**
фіолетовий	purple		**800080**
ліловий	violet		**8000FF**
жовтий	yellow		**FFFF00**
коричневий	brown		**996633**
помаранчевий	orange		**FF8000**

Повний список імен і відповідних їм кольорів можна подивитися на Web-сторінці http://msdn.mkrosoft.com/en-us/library/aa358802%28v=VS.85%29.aspx.

Значення **inherit** вказує, що даний елемент Web-сторінки повинен мати той же колір шрифту, що і батьківський елемент. Це значення атрибута стилю **color** за замовчуванням.

За допомогою атрибута стилю **color** ми можемо також задати колір горизонтальної лінії HTML.

Атрибут стилю `opacity` - ступінь напівпрозорості

Атрибут стилю **opacity** дозволяє вказати ступінь напівпрозорості елемента Web- сторінки:

```
opacity:<числове значення>|inherit
```

Значення напівпрозорості є число від **0** до **1**. При цьому **0** позначає повну прозорість елемента (тобто елемент фактично не видний), а **1** — повну непрозорість (це звичайна поведінка).

От приклад завдання половинної прозорості (значення 0,5) для тексту фіксованого форматування:

```
PRE {opacity:0.5}
```

Відзначимо, як ми вказали дробове число — замість символу коми тут використовується точка.

Зауваження. Напівпрозорість звичайно доцільна тільки для створення спеціальних ефектів.

Атрибут стилю `font-weight` - "жирність" шрифту:

Атрибут стилю **font-weight** встановлює "жирність" шрифту:

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

Тут доступні сім абсолютних значень від 100 до **900,** що визначають різну "жирність" шрифту, від мінімальної до максимальної; при цьому звичайний шрифт буде мати "жирність" **400** (або **normal**), а напівжирний — **700** (або **bold**). Значення за замовчуванням — **400** (**normal**). Значення **bolder** і **lighter** є відносними і визначають наступні ступені "жирності" відповідно в більшу і меншу сторону.

Приклад:

```
CODE {font-weight:bold}
```

Атрибут `font-style` - накреслення шрифту

Атрибут **`font-style`** задає накреслення шрифту:

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

Доступні три значення, що визначають:

- звичайний шрифт (**`normal`**),
- курсив (**`italic`**) і
- особливе декоративне накреслення, схоже на курсив (**`oblique`**).

Атрибут стилю `text-decoration` - підкреслення або закреслювання тексту

Атрибут стилю **`text-decoration`** задає "прикрасу" (підкреслення або закреслювання), яке буде застосовано до тексту:

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

Тут доступні п'ять значень (не враховуючи **`inherit`**):

- **`none`** забирає всі "прикраси", задані для шрифту батьківського елемента;
- **`underline`** підкреслює текст;
- **`overline`** "надкреслює" текст, тобто проводить лінію над рядками;
- **`line-through`** закреслює текст;
- **`blink`** робить шрифт мерехтливим (на даний момент не підтримується Safari).

Увага! Не потрібно без особливої необхідності задавати для тексту підкреслення. Справа в тому, що Браузери за замовчуванням виводять гіперпосилання підкресленим, і підкреслений текст, що не є гіперпосиланням, може сприйнятися як гіперпосилання.

Атрибут стилю `font-variant` малі прописні букви шрифту

Атрибут стилю **`font-variant`** дозволяє вказати, як будуть виглядати малі літери шрифту:

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

Значення **small-caps** задає таку поведінку шрифту, коли його малі літери виглядають точно так само, як прописні, просто мають менший розмір. Значення **normal** задає для шрифту звичайні прописні букви.

Атрибут стилю `text-transform` - змінити регістр символів тексту:

Атрибут стилю **text-transform** дозволяє змінити регістр символів тексту:

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

Ми можемо трансформувати текст до верхнього (значення **uppercase** цього атрибута) або нижньому (**lowercase**) регістру, перетворити до верхнього регістру першу букву кожного слова (**capitalize**) або залишити в початковому вигляді (**none**).

Атрибут стилю `line-height` задає висоту рядка тексту:

Атрибут стилю **line-height** задає висоту рядка тексту:

```
line-height: normal|<відстань>|inherit
```

Тут можна задати абсолютну і відносну величину відстані, вказавши відповідну одиницю виміру CSS (див. таблицю 9.1). При її відсутності задане нами значення спочатку множиться на висоту поточного шрифту і потім використовується. Таким чином, щоб збільшити висоту рядка вдвічі в порівнянні зі звичайною, ми можемо написати:

```
P {line-height:2}
```

Значення **normal** цього атрибута повертає управління висотою рядка Браузеру.

Атрибут стилю `letter-spacing` - додаткова відстань між символами тексту

Атрибут стилю **letter-spacing** дозволяє задати додаткову відстань між символами тексту:

```
letter-spacing:normal|<відстань>
```

Відзначимо, що це саме додаткова відстань; вона буде додана до початкової, встановленої самим Браузером.

Тут також можна задати абсолютну і відносну відстань, вказавши відповідну одиницю виміру CSS (див. таблицю 9.1). Відстань може бути додатньою і від'ємною; в останньому випадку символи шрифту будуть розташовуватися друг до друга ближче звичайного. Значення **normal** встановлює додаткову відстань за замовчуванням, рівну нулю.

Атрибут стилю **letter-spacing** не підтримує значення **inherit.**

От приклад завдання додаткової відстані між символами рівного п'яти пікселам:

```
H1 {letter-spacing:5px}
```

Текст, набраний такими символами, буде виглядати розрідженим.

А тут ми задали від'ємну додаткову відстань між символами рівну двом пікселам:

```
H6 {letter-spacing:-2px}
```

Ці два піксела будуть відняті від початкової відстані, в результаті символи зблизяться, а текст стане виглядати стиснутим. Можливо, символи навіть налізуть один на одного.

Атрибут стилю word-spacing - додаткова відстань між словами тексту

Аналогічний атрибут стилю **word-spacing** задає додаткову відстань між окремими словами тексту:

```
word-spacing:normal|<відстань>
```

Приклад:

```
H1 {word-spacing:5mm }
```

Атрибут стилю font - одночасно відразу кілька параметрів шрифту

І наприкінці розглянемо атрибут стилю **font**, що дозволяє задати одночасно відразу кілька параметрів шрифту:

```
font:[<накреслення>] [<вид малих літер>] [<"жирність">]
[<розмір>[/<висота рядка тексту>]] <ім'я шрифту>
```

Як бачимо, обов'язковим є тільки ім'я шрифту — інші параметри можуть бути відсутніми.

Приклад. Задаємо для тексту абзаців шрифт **Times New Roman** розміром **10** пунктів. А для заголовків шостого рівня — шрифт **Arial** розміром **12** пунктів і курсивного накреслення

```
P {font:10pt "Times New Roman"}
H6 {font:italic 12pt Verdana}
```

Параметри, що управляють розривом рядків

За замовчуванням Браузер розбиває текст на рядки так, щоб умістити його у вікно і уникнути горизонтального прокручування. Далеко не завжди при цьому він розриває рядки, як нам потрібно. Звичайно, ми можемо встановити фіксоване форматування тексту і примусово вказати, де слід переносити рядки, але не завжди це кращий підхід.

CSS пропонує два атрибути стилю, що дозволяють нам указати, як Браузеру слід розбивати текст на рядки. Зараз ми їх розглянемо.

Атрибут стилю `white-space`

Атрибут стилю **`white-space`** задає правила, якими Браузер керується при виведенні тексту. Зокрема, з його допомогою ми можемо змінити правила, встановлені за замовчуванням. Формат запису цього атрибута стилю:

```
white-space:normal|pre|nowrap|pre-wrap|pre-line|inherit
```

Атрибут стилю **`white-space`** може мати п'ять значень.

- **`normal`** — послідовності пробілів перетворяються в один пробіл, перенесення рядків також перетворяються в пробіли. Браузер сам розриває текст на рядки, щоб умістити його в своє вікно по ширині. Фактично це значення наказує Браузеру застосовувати для виведення тексту блокових елементів правила за замовчуванням.
- **`pre`** — послідовності пробілів і перенесення рядків зберігаються; текст виводиться точно в такому вигляді, у якому він записаний в HTML-коді. Браузер сам не розриває текст на рядки. Фактично текст виводиться так, немов він поміщений у тег **`<PRE>`** (текст фіксованого форматування).
- **`nowrap`** — послідовності пробілів перетворяться в один пробіл, перенесення рядків також перетворяться в пробіли. Однак Браузер сам не розриває текст на рядки.

- **pre-wrap** — послідовності пробілів і перенесення рядків зберігаються. Браузер може розірвати занадто довгі рядки, щоб уникнути горизонтального прокручування.
- **pre-line** — послідовності пробілів перетворяться в один пробіл, перенесення рядків зберігаються. Браузер може розірвати занадто довгі рядки, щоб уникнути горизонтального прокручування.

У табл. 9.3 приведені доступні для атрибута стилю **white-space** значення.

Табл. 9.3. **Значення атрибута стилю white-space і результати їх застосування**

Значення	Перенесення рядків	Послідовності пробілів	Розрив тексту на рядки
normal	Перетворяться в пробіли	Перетворяться в один пробіл	Виконується
pre	Зберігаються	Зберігаються	Не виконується
nowrap	Перетворяться в пробіли	Перетворяться в один пробіл	Не виконується
pre-wrap	Зберігаються	Зберігаються	Виконується
pre-line	Перетворяться в пробіли	Зберігаються	Виконується

От стиль, що перевизначає тег **<PRE>** так, щоб при необхідності його вміст розривався на рядки:

```
PRE {white-space:pre-wrap}
```

Атрибут стилю word-wrap - місця розриву тексту

Атрибут стилю **word-wrap** застосовується нечасто, але в деяких випадках без нього не обійтися. Він дозволяє вказати місця, у яких Браузер може виконати розрив тексту:

```
word-wrap:normal|break-word|inherit
```

Тут доступні два значення.

- **normal** — вказує Браузеру, що він може розривати текст на рядки тільки по пробілах. Це звичайна поведінка Браузера.
- **break-word** — дозволяє Браузеру виконувати розрив тексту на рядки всередині слів. Це може пригодитися, якщо текст містить багато дуже довгих слів, які по ширині не поміщаються в батьківський елемент.

Приклад:

```
P {word-wrap:break-word}
```

Тут ми дозволили Браузеру виконувати розрив тексту на рядки всередині слів у вмісті тегів **<P>** (тобто в абзацах).

Параметри вертикального вирівнювання

Іноді виникає ситуація, коли потрібно змістити фрагмент по вертикалі щодо тексту, який його оточує, тобто задати вертикальне вирівнювання тексту.

Атрибут стилю **vertical-align** саме задає вертикальне вирівнювання тексту:

```
vertical-align:baseline|sub|super|top|text-top|middle|
bottom|text-bottom|<проміжок між базовими лініями>|inherit
```

Цей атрибут стилю приймає вісім визначених значень:

- **baseline** — задає вирівнювання базової лінії фрагмента тексту по базовій лінії тексту батьківського елемента (ця поведінка за замовчуванням). Базовою називається уявлювана лінія, на якій розташовується текст.
- **sub** — вирівнює базову лінію фрагмента тексту по базовій лінії нижнього індексу батьківського елемента.
- **super** — вирівнює базову лінію фрагмента тексту по базовій лінії верхнього індексу батьківського елемента.
- **top** — вирівнює верхній край фрагмента тексту по верхньому краю батьківського елемента.
- **text-top** — вирівнює верхній край фрагмента тексту по верхньому краю тексту батьківського елемента.
- **middle** — вирівнює центр фрагмента тексту по центру батьківського елемента.
- **bottom** — вирівнює нижній край фрагмента тексту по нижньому краю батьківського елемента.
- **text-bottom** — вирівнює нижній край фрагмента тексту по нижньому краю тексту батьківського елемента.

Крім того, ми можемо вказати для даного атрибута стилю абсолютне або відносне значення, що задає, наскільки вище або нижче базової лінії тексту батьківського елемента повинна знаходитися базова лінія фрагмента тексту:

```
STRONG {vertical-align:super;
font-size:smaller}
```

Цей стиль перевизначення тегу **<STRONG>** задає для тексту розташування, що збігається з базовою лінією верхнього індексу, і зменшений розмір шрифту. Фактично за допомогою цього стилю ми перетворюємо вміст тегу у верхній індекс **<STRONG>**.

Той же атрибут стилю придатний для вирівнювання графічних зображень, що є частиною абзацу:

```
<P>Це картинка:<IMG STYLE="vertical-align:text-bottom"
SRC="picture.png">.</P>
```

Даний HTML-код створює абзац із графічним зображенням. Низ цього зображення буде вирівняний по нижньому краю тексту абзацу. Іншими словами, зображення буде як би підніматися над текстом.

Швидше за все, для досягнення потрібного результату прийдеться поекспериментувати із різними значеннями атрибута стилю **vertical-align**.

Параметри тіні в тексту

Аматорам все прикрашати стандарт CSS 3 пропонує одну дуже цікаву можливість — створення тіні в тексті. При помірному вживанню вона може помітно пожвавити Web-сторінку.

Параметри тіні задає атрибут стилю **text-shadow**:

```
text-shadow:none|<колір> <горизонтальний зсув>
<вертикальний зсув> [<радіус розмиття>]
```

Значення **none** (встановлене за замовчуванням) забирає тінь у текста.

Колір тіні задається у вигляді RGB-коду або іменованого значення.

Горизонтальний зсув тіні задається в будь-якій одиниці виміру, підтримуваної CSS (див. таблиця 9.1). Якщо заданий додатний зсув, тінь буде розташована праіше тексту, якщо від'ємне— лівіше. Можна також задати і

нульовий зсув; тоді тінь буде розташовуватися прямо під текстом. Нульовий зсув має сенс тільки в тому випадку, якщо для тіні задане розмиття.

Вертикальний зсув тіні також задається в будь-якій одиниці виміру, підтримуваної CSS. Якщо заданий додатний зсув, тінь буде розташована нижче тексту, якщо від'ємне — вище. Можна також задати і нульовий зсув; тоді тінь буде розташовуватися прямо під текстом.

Радіус розмиття тіні також задається в будь-якій одиниці виміру, підтримуваної CSS. Якщо радіус розмиття не вказаний, його значення вважаається рівним нулю; в такому випадку тінь не буде мати ефекту розмиття.

Приклад:

```
H1 {text-shadow:black 1mm 1mm 1px}
```

Тут ми задали для заголовків першого рівня (тегу **<H1>**) тінь, розташовану правіше і нижче тексту на **1** мм, що має радіус розмиття **1** піксел.

Параметри фона

Закінчивши з параметрами тексту, займемося фоном. Фон можна вказати для фрагмента тексту (вбудованого елемента), блокового елемента, таблиці, її комірки і всієї Web-сторінки. Добре підібраний фон може пожвавити Web-сторінку і виділити окремі її елементи.

Фон у окремих елементів, відмінний від фона самої Web-сторінки, слід задавати тільки в крайніх випадках. Інакше Web-сторінка стане занадто строкатої і незручної для читання.

Атрибут стилю `background-color` - для завдання кольору фона

Атрибут стилю **background-color** служить для завдання кольору фона:

```
background-color:transparent|<колір>|inherit
```

Колір можна задати у вигляді RGB-кода або імені. Значення **transparent** забирає фон зовсім; тоді елемент Web-сторінки отримає "прозорий" фон. За замовчуванням фон в елементів Web-сторінки відсутній, а фон самої Web-сторінки задає Браузер.

Приклад:

```
BODY {color:white;
```

```
background-color:black}
```

Тут ми задали для всієї Web-сторінки чорний фон і білий текст.

Атрибут стилю `background-image` для завдання фонового зображення

Атрибут стилю **`background-image`** дозволяє призначити як фон графічне зображення (фонове зображення):

```
background-image:none|url(<інтернет-адреса файлу зображення>);
```

Звернемо увага, в якому вигляді задається інтернет-адреса файлу з фоновим зображенням: її беруть у дужки, а перед ними ставлять символи **`url:`**

```
TABLE.bgr {background-image:url("/table_background.png")}
```

Значення **`none`** забирає графічний фон.

Графічний фон виводиться поверх звичайного фона, заданого нами за допомогою атрибута стилю **`background-color`**. І, якщо фонове зображення містить "прозорі" фрагменти (таку можливість підтримують формати **GIF** і **PNG**), ций фон буде "просвічувати" крізь них.

Приклад:

```
TABLE.yellow {background-color:yellow;
              background-
              image:url("/yellow_background.png")}
```

Тут ми задали для таблиці і звичайний, і графічний фон. Це, до речі, поширена практика в Web-дизайні.

Атрибут стилю `background-repeat` для повтору фонового зображення

Якщо фонове зображення менше, чим елемент Web-сторінки (або сама Web-сторінка), для якого воно задано, Браузер буде повторювати це зображення, поки не "забрукує" їм весь елемент. Параметри цього повторення задає атрибут стилю **`background-repeat`**:

```
background-repeat:no-repeat|repeat|repeat-x|repeat-
y|inherit
```

Тут доступні чотири значення.

- **no-repeat** — фонове зображення не буде повторюватися ніколи; в цьому випадку частина фона елемента Web-сторінки залишиться не заповненою їм.
- **repeat** — фонове зображення буде повторюватися по горизонталі і вертикалі (звичайна поведінка).
- **repeat-x** — фонове зображення буде повторюватися тільки по горизонталі.
- **repeat-y** — фонове зображення буде повторюватися тільки по вертикалі.

Атрибут стилю background-position для вказівки позиції фонового зображення

За допомогою атрибута стилю **background-position** можна вказати позицію фонового зображення щодо елемента Web-сторінки, для якого воно призначене:

```
background-position:<горизонтальна позиція>
[<вертикальна позиція>]|inherit;
```

Горизонтальна позиція фонового зображення задається в такому форматі:

```
<числове значення>|left|center|right
```

Числове значення вказує місце розташування фонового зображення в елементі Web-сторінки по горизонталі і може бути задане із застосуванням кожної з підтримуваних CSS одиниць виміру (див. таблицю 9.1). Також можна вказати наступні значення:

- **left** — фонове зображення притискається до лівого краю елемента Web-сторінки (це звичайна поведінка);
- **center** — розташовується по центру;
- **right** — притискається до правого краю.

Формат завдання вертикальної позиції фонового зображення такий:

```
<числове значення>|top|center|bottom
```

Числове значення вказує місце розташування фонового зображення в елементі Web-сторінки по вертикалі і може бути задане із застосуванням кожної з підтримуваних CSS одиниць виміру.

Також можливі такі значення:

- **top** — фонове зображення притискається до верхнього краю елемента Web-сторінки (це звичайна поведінка);
- **center** — розташовується по центру;
- **bottom** — притискається до нижнього краю.

Якщо для якого-небудь елемента Web-сторінки вказана тільки позиція фонового зображення по горизонталі, його вертикальна позиція приймається рівної **center**

Приклад:

```
TABLE.bgr (background-position:1cm top }
```

Цей стиль поміщає фонове зображення на відстані **1** см від лівого краю елемента Web-сторінки і притискає його до нижнього краю даного елемента.

А от стиль, що притискає фонове зображення до правого краю елемента Web-сторінки і розташовує його в центрі даного елемента по вертикалі:

```
TABLE.bgr (background-position:right}
```

Атрибут стилю `background-attachment` управляє фіксацією фона

Коли ми прокручуємо вміст Web-сторінки у вікні Браузера, разом з нею прокручується і фонове зображення (якщо воно є). Стандарт CSS пропонує забавну можливість — заборону прокручування графічного фона Web-сторінки і фіксація його на місці. Фіксацією фона управляє атрибут стилю **background-attachment**:

```
background-attachment:scroll|fixed;
```

Значення **scroll** змушує Браузер прокручувати фон разом із вмістом Web-сторінки (це поведінка за замовчуванням). Значення **fixed** фіксує фон на місці, і він не буде прокручуватися.

Контейнери. Вбудовані контейнери

Усі розглянуті нами атрибути стилів можна вказувати для будь-яких елементів Web-сторінок: і блокових, і вбудованих. Виходить, ми можемо задати розмір шрифту і для абзацу (блокового тегу **<P>**), і для важливого тексту (вбудованих тегів **<STRONG>** і **<EM>**).

Але що робити, якщо нам знадобилося застосувати який-небудь стиль до довільного фрагмента тексту, не позначаючи його ніяким тегом?

Наприклад, нам потрібно виділити напівжирним шрифтом фрагмент абзацу, але ми не хочемо помістити його в тег **<STRONG>**. Чи може CSS нам в цьому допомогти?

CSS не може. Зате може HTML. Він спеціально для таких випадків дає особливі елементи Web-сторінки — контейнери — і, звичайно відповідні теги. Про контейнери зараз і піде розмова.

Контейнер — елемент Web-сторінки, призначений тільки для виділення якого-небудь її фрагмента. Таким фрагментом може бути частина блокового елемента (абзацу, заголовка, цитати, тексту фіксованого форматування та ін.), блоковий елемент або відразу кілька блокових елементів. Браузер ніяк не виділяє контейнер на Web-сторінці.

Контейнер служить двом цілям. По-перше, з його допомогою ми можемо прив'язати до певного елемента або елементів Web-сторінки потрібний стиль; для цього досить вкласти даний елемент або елементи в контейнер і прив'язати стиль до нього. По-друге, він може забезпечувати прив'язку поведінки до елемента або елементів Web-сторінки; виконується це в такий же спосіб, що і у випадку стилю.

Контейнери бувають блокові і вбудовані. Розмову про блокові контейнери ми відкладемо до розгляду контейнерного Web-дизайну. Поговоримо про вбудовані контейнери.

Уже по визначенню ясно, що вбудований контейнер є частиною блокового елемента Web-сторінки. Так, блоковим контейнером може стати фрагмент абзацу або цитати, графічне зображення, поміщене в абзац, та ін.

Вбудований контейнер створюється за допомогою парного тегу **<SPAN>.** Фрагмент блокового елемента, який потрібно перетворити у вміст вбудованого контейнера, поміщають в цей тег:

```
<P><SPAN>Представлення</SPAN> створюється за допомогою стилів CSS.</P>
```

Тут ми помістили у вбудований контейнер фрагмент абзацу.

Толку від нашого першого вбудованого контейнера ніякого. Тому давайте прив'яжемо до нього який-небудь стиль (лістинг 9.1).

Лістинг 9.1

```
.bolded {font-weight: bold}
......
<P><SPAN CLASS="bolded">Представлення</SPAN> створюється за допомогою стилів CSS.</P>
```

Тепер слово **"Представлення"** буде набрано напівжирним шрифтом.

Представлення для нашого Web-сайту, частина 1

Почнемо створювати представлення для нашого Web-сайту.

Всі стилі, які ми застосуємо до Web-сторінкам, помістимо в зовнішню таблицю стилів **main.css**. Створимо її та розташуємо в кореневій папці Web-сайту. Після чого прив'яжемо її до всіх Web-сторінкам, що входять у нього. Це виконується за допомогою тегу **<LINK>.**

Для головної Web-сторінки **index.htm** цей тег буде виглядати так:

```
<LINK REL="stylesheet" HREF="main.css" TYPE="text/css">
```

Для всіх Web-сторінок, що зберігаються в папці **tags**, він буде таким:

```
<LINK REL="stylesheet" HREF="../main.css" TYPE="text/css">
```

Внесемо в HTML-код всіх Web-сторінок цей тег і збережемо їх.

У тегах **<LINK>**, що прив'язують таблицю стилів до Web-сторінок, ми вказали відносні інтернет-адреси. Це дозволить нам переглядати Web-сторінки, відкриваючи їх у Браузері без використання Web-сервера.

Тепер наповнимо нашу таблицю стилів потрібними стилями. Але спочатку визначимося, що ми прагнемо створити. Опишемо параметри шрифту і фона наших Web-сторінок звичайною, "людською" мовою.

Концепція Web 2.0 визначає специфічні вимоги до шрифту і фону Web-сторінок. Це тонкі і досить великі шрифти без зарубок, приглушені кольори, звичайний (неграфічний) або однотонний графічний фон. Інші характеристики, як говориться, по смакові.

Виходячи із цього, для Web-сторінки ми задамо такі параметри.

Шрифт абзаців — **Verdana**. Якщо такий відсутній на клієнтському комп'ютері, застосуємо шрифт **Arial.**

- Шрифт заголовків — **Arial**. Це дозволить нам додатково виділити заголовки, зробити їх відмінними від звичайних абзаців.
- Розмір шрифту абзаців — **12** пунктів.
- Розмір шрифту заголовків першого рівня — **20** пунктів.
- Розмір шрифту заголовків другого рівня — **18** пунктів.
- Розмір шрифту заголовків шостого рівня — **12** пунктів.
- Шрифт заголовків всіх рівнів — звичайної насиченості.
- Розмір шрифту в таблицях — **10** пунктів. Нехай таблиці також будуть відрізнятися від звичайного тексту.

- Розмір шрифту великих цитат — **10** пунктів. Нехай і цитати виглядають по-іншому.
- Шрифт великих цитат — курсивний.
- Розмір шрифту тегу адреси — **10** пунктів. Відомості про авторські права теж повинні відрізнятися. До того ж їх традиційно пишуть дрібним шрифтом.
- Шрифт тегу адреси — курсивний.
- Колір тексту — **#3B4043** (дуже-дуже темний, майже чорний).
- Колір фона **#E8E8E8** (дуже-дуже світлий, майже білий).

Для підбора кольорів можна порекомендувати бібліотеки колірних тем Web-сайтів, доступних по інтернет-адресам http://www.tarusa.ru/~golovan і http://avy.ru.

Залишилося написати CSS-код відповідно до викладеного (лістинг 9.2).

Лістинг 9.2

```
BODY        {color:#3B4043;
             background-color:#F8F8F8;
             font-family: Verdana, Arial, sans-serif}
P           {font-size:12pt}
H1, H2, H6 {font-weight:normal;
             font-family:Arial, sans-serif}
H1          {font-size:20pt}
H2          {font-size:18pt}
H6          {font-size:12pt}
TABLE       {font-size:10pt}
BLOCKQUOTE P,
ADDRESS     {font-size:10pt;
             font-style:italic }
```

Таблиця стилів готова. Послідовно розглянемо всі створені в ній стилі.

Першим іде стиль перевизначення тегу **<BODY>.** Він задає параметри, загальні для всієї Web-сторінки: шрифт для звичайного тексту (абзаців, цитат і вмісту таблиць), колір тексту і колір фона. Всі елементи Web-сторінки будуть використовувати дані параметри, якщо, звичайно, ми не перевизначимо їх далі, у більш конкретних тегах:

```
BODY {color:#3B4043;
      background-color:#F8F8F8;
```

```
font-family: Verdana, Arial, sans-serif}
```

Наступним іде стиль перевизначення тегу **<P>.** Він задає розмір шрифту для тексту абзаців. Фактично він доповнює стиль перевизначення тегу **<BODY>**, створений раніше:

```
P {font-size:12pt}
```

Ми створили три однакові стилі, що перевизначають теги заголовків **<H1>, <H2>** і **<H6>:**

```
H1, H2, H6 {font-weight:normal;
            font-family:Arial, sans-serif}
```

Вони задають параметри, загальні для всіх заголовків: шрифт і його "жирність" (точніше, відсутність "жирності"). Оскільки ці стилі більш конкретні, чим створений раніше стиль перевизначення тегу **<BODY>**, задані в них параметри будуть мати більший пріоритет. Отже, заголовки будуть набрані шрифтом, який ми вказали в цих стилях, а не тим, що вказаний в стилі перевизначення тегу **<BODY>.**

Далі ми створили три стилі перевизначення тегів **<H1>, <H2>** і **<H6>**, що задають різні розміри шрифту для заголовків різного рівня. Ці стилі доповнять ті, що ми створили трохи раніше. В результаті заголовки різного рівня будуть набрані шрифтом різного розміру:

```
H1 {font-size:20pt}
H2 {font-size:18pt}
H6 {font-size:12pt}
```

Наступним іде стиль перевизначення тегу **<TABLE>**, що задає розмір шрифту і доповнює створений раніше стиль перевизначення **тегу <BODY>.** Шрифтом даного розміру буде набраний текст у всіх елементах таблиці (звичайних комірках, комірках заголовка і заголовку таблиці):

```
TABLE {font-size:10pt}
```

Останніми ми визначили два однакові стилі: комбінований стиль **BLOCKQUOTE P** і стиль перевизначення тегу **<ADDRESS>**:

```
BLOCKQUOTE P, ADDRESS{font-size:10pt;
                      font-style:italic }
```

Вони задають однакові параметри для шрифту великої цитати і тегу адреси. Оскільки для створення великої цитати ми використовували тег **<P>,** вкладений у тег **<BLOCKQUOTE>,** то параметри тексту цитати ми визначили за допомогою комбінованого стилю **BLOCKQUOTE P**. Обидва ці стилі доповнюють створений на самому початку стиль перевизначення тегу **<BODY>.**

Стилі об'єднуються один з одним, перевизначаючи задані в них параметри, згідно із пріоритетом. А пріоритет залежить від конкретності даного стилю, від "близькості" його до тегу.

Збережемо таблицю стилів і відкриємо в Браузері Web-сторінку **index.htm**. От що можна зробити з Web-сторінкою за допомогою стилів CSS. І адже нам зовсім не довелося правити її HTML-код (якщо не вважати внесення тегу **<LINK>,** що виконує прив'язку таблиці стилів).

Давайте трохи розрідимо текст заголовків, щоб зробити їх більш помітними. Для цього досить додати до початкової відстані між його символами додаткову, рівну 1 мм.

Але куди помістити відповідний атрибут стилю? В CSS-код, що створює три однакові стилі перевизначення тегів **<H1>, <H2>** і **<H6>.** От він:

```
H1, H2, H6 {font-weight:normal;
            font-family:Arial, sans-serif }
```

А так він буде виглядати після відповідного виправлення:

```
H1, H2, H6 {font-weight:normal;
           font-family:Arial,      sans-serif;      letter-
           spacing:1mm}
```

Збережемо таблицю стилів, вибравши кодування UTF-8, і обновимо Web-сторінку **index.htm**, відкриту в Браузері, натиснувши клавішу **<F5>.**

Давайте задіємо можливості CSS 3 і створимо для тексту заголовків тінь. Додамо відповідний атрибут стилю, знову ж, в CSS-код, що створює три однакові стилі перевизначення тегів **<H1>, <H2>** і **<H6>** (лістинг 9.3).

Лістинг 9.3

```
H1, H2, H6 {font-weight:normal;
            font-family:Arial, sans-serif;
            letter-spacing:1mm;
            text-shadow:#CDD9DB 1px 1px }
```

Для тіні ми задали колір **:#CDD9DB** (ясно-синій) і зовсім невеликі відступи, рівні 1 мм. Така тінь буде ненав'язливою, але симпатичною.

Знову збережемо таблицю стилів і обновимо Web-сторінку.

КОНТРОЛЬНІ ПИТАННЯ

1. Які параметри шрифту ви знаєте?
2. Назвіть основні параметри шрифту.
3. Як змінити одночасно відразу кілька параметрів шрифту?
4. Назвіть параметри, що управляють розривом рядків.
5. Назвіть параметри вертикального вирівнювання.
6. Які параметри тіні в тексті ви знаєте?
7. Які параметри фона ви знаєте?
8. Які параметри управляють завданням фонового зображення?
9. Що таке контейнери і вбудовані контейнери? Для чого вони використовуються?
10. Які види контейнерів ви знаєте? Як створити вбудований контейнер?

ТЕМА 10. ПАРАМЕТРИ АБЗАЦІВ, СПИСКІВ І ВІДОБРАЖЕННЯ

***Основна мета*:** вивчити атрибути стилю, що задають параметри абзаців, списків і відображення елементів Web-сторінки.

.

ПАРАМЕТРИ АБЗАЦІВ, СПИСКІВ І ВІДОБРАЖЕННЯ

Параметри виведення тексту

Атрибут стилю **`text-align`** *- горизонтальне вирівнювання тексту*

Атрибут стилю **`text-indent`** *- відступ для " червоного рядка"*

Параметри списків

Атрибут стилю **`list-style-type`** *- вигляд маркерів або нумерації в пунктів списку*

Атрибут стилю **`list-style-image`** *для створення графічного маркера*

Атрибут стилю **`list-style-position`** *для вказівки місця розташування маркера або нумерації в пункті списку*

Параметри відображення

Атрибут стилю **`visibility`** для вказівки чи буде елемент відображатися

Атрибут стилю **`display`** *для задання вигляду елемента Web-сторінки: буде він блоковим, вбудованим або пунктом списку*

Представлення для нашого Web-сайту, частина 2

Створення смуги навігації

Параметри курсору

КОНТРОЛЬНІ ПИТАННЯ

Ми познайомилися із стилями і таблицями стилів CSS, за допомогою яких створюється представлення Web-сторінок: чотири різновиди стилів і два різновиди таблиць стилів.

Ми почали вивчати можливості мови CSS. Ми вже розглянули атрибути стилів, що задають параметри шрифту і фона елементів Web-сторінок. Тепер розглянемо параметри абзаців, списків і відображення.

ПАРАМЕТРИ АБЗАЦІВ, СПИСКІВ І ВІДОБРАЖЕННЯ

Параметри виведення тексту

Почнемо ми з атрибутів стилю, керуючих виведенням тексту в блокових елементах, що структурують текст. Їхнім зовсім мало. І всі вони застосовні тільки до блокових елементів.

Атрибут стилю `text-align` - горизонтальне вирівнювання тексту:

Атрибут стилю **`text-align`** задає горизонтальне вирівнювання тексту:

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

Тут доступні значення

- **`left`** (вирівнювання по лівому краю; звичайна поведінка Браузера),
- **`right`** (по правому краю),
- **`center`** (по центру) і
- **`justify`** (повне вирівнювання).

Приклади:

```
P {text-align:justify}
H1 {text-align:center}
```

Атрибут стилю `text-indent` - відступ для " червоного рядка"

Атрибут стилю **`text-indent`** задає відступ для " червоного рядка":

```
text-indent:<відступ " червоного рядка">
```

Тут допускаються абсолютні і відносні (щодо ширини абзацу) величини відступу. За замовчуванням відступ "червоного рядка" дорівнює нулю. Відзначимо, що атрибут стилю **`text-indent`** не підтримує значення **`inherit`**.

Приклад:

```
P {text-indent:5mm}
```

От тепер абзаци будуть мати "червоний рядок".

Параметри списків

Списки серед блокових елементів стоять особняком. В основному, через те, що, по-перше, містять в собі інші блокові елементи (окремі пункти), а по-друге, включають маркери і нумерацію, які розставляє сам Браузер. От про маркери і нумерації, а точніше, про атрибути стилю, призначені для завдання їх параметрів, ми зараз і поговоримо.

Атрибут стилю `list-style-type` - вигляд маркерів або нумерації в пунктів списку

Атрибут стилю **`list-style-type`** задає вигляд маркерів або нумерації в пунктів списку:

```
list-style-type:disc|circle|square|decimal|
decimal-leading-zero|lower-roman|upper-roman|
lower-greek|lower-alpha|lower-latin|upper-alpha|
upper-latin|armenian|georgian|none|inherit
```

Як бачимо, доступних значень у цього атрибута стилю дуже багато. Вони позначають як різні види маркерів, так і різні способи нумерації.

- **`disc`** — маркер у вигляді чорного кружка (звичайна поведінка для маркірованих списків).
- **`circle`** — маркер у вигляді світлого кружка.
- **`square`** — маркер у вигляді квадратика. Він може бути світлим або темним, залежно від Браузера.
- **`decimal`** — нумерація арабськими цифрами (звичайна поведінка для нумерованих списків).
- **`decimal-leading-zero`** — нумерація арабськими цифрами від 01 до 99 з початковим нулем.
- **`lower-roman`** — нумерація маленькими римськими цифрами.
- **`upper-roman`** — нумерація більшими римськими цифрами.
- **`lower-greek`** — нумерація маленькими грецькими буквами.
- **`lower-alpha`** і **`lower-latin`** — нумерація маленькими латинськими буквами.
- **`upper-alpha`** і **`upper-latin`** — нумерація великими латинськими буквами.
- **`armenian`** — нумерація традиційними вірменськими цифрами.
- **`georgian`** — нумерація традиційними грузинськими цифрами.

- **none** — маркер і нумерація відсутні (звичайна поведінка для не-списків).

Атрибут стилю **list-style-type** можна задавати як для самих списків, так і для окремих пунктів списків. В останньому випадку створюється список, в якому пункти мають різні маркери або нумерацію. Іноді це може пригодитися.

От визначення маркірованого списку з маркером у вигляді квадратика:

```
UL {list-style-type:square}
```

А в лістингу 10.1 ми задали такий же маркер для одного з пунктів маркірованого списку.

Лістинг 10.1

```
.squared { list-style-type: square} <UL>
<LI>nepbbm nyhkt</LI>
<LI CLASS="squared">Другий пункт (з іншим Маркером)</LI>
<LI>Tpetkm nyhkt</LI> </UL>
```

Атрибут стилю `list-style-image` для створення графічного маркера

Атрибут стилю **list-style-image** дозволяє задати як маркер пунктів списку яке-небудь графічне зображення (створити *графічний маркер)*. В цьому випадку значення атрибута стилю **list-style-image**, якщо такий заданий, ігнорується:

```
list-style-image:none|<інтернет-адреса файлу зображення>|
inherit
```

Інтернет-адреса файлу із графічним маркером задається в такому ж форматі, що і інтернет-адреса файлу фонового зображення:

```
UL {list-style-image:url(/markers/dot.gif)}
```

Значення **none** забирає графічний маркер і повертає простий, неграфічний. Це звичайна поведінка.

Атрибут стилю **list-style-image** також можна задавати як для самих списків, так і для окремих їхніх пунктів.

Атрибут стилю `list-style-position` для вказівки місця розташування маркера або нумерації в пункті списку

Атрибут стилю **`list-style-position`** дозволяє вказати місце розташування маркера або нумерації в пункті списку:

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

Доступних значень тут два:

- **`inside`** — маркер або нумерація знаходяться як би усередині пункту списку, є його частиною;
- **`outside`** — маркер або нумерація винесені за межі пункту списку (це звичайна поведінка).

Приклад:

```
OL {list-style-position:inside}
```

Параметри відображення

Ще одна група атрибутів стилю управляє тем, як елемент буде відображатися на Web-сторінці, тобто буде він блоковим або вбудованим, і чи буде він відображатися взагалі. Ці атрибути стилю застосовні до будь-яких елементів Web-сторінок.

Атрибут стилю `visibility` для вказівки чи буде елемент відображатися

Атрибут стилю **`visibility`** дозволяє вказати, чи буде елемент відображатися на Web-сторінці:

```
visibility:visible|hidden|collapse|inherit
```

Він може приймати три значення:

- **`visible`** — елемент відображається на Web-сторінці (це звичайна поведінка);
- **`hidden`** — елемент не відображається на Web-сторінці, однак під нього все ще виділене на ній місце. Інакше кажучи, замість елемента на Web-сторінці знахлдиться порожня "діра";
- **`collapse`** — застосований тільки до рядків, секцій, колонок і груп колонок таблиці. Елемент не відображається на Web-сторінці, і під нього не виділяється місце на ній (тобто ніяких "дір"). Однак Браузер вважає,

що даний елемент Web-сторінки все ще на ній присутній. Дане значення підтримують не всі Браузери Браузер.

Атрибут стилю **visibility** застосовується досить рідко і тільки для створення спеціальних ефектів, на зразок елемента Web-сторінки, що зникає або з'являється.

Атрибут стилю `display` *для задання вигляду елемента Web-сторінки: буде він блоковим, вбудованим або пунктом списку*

Атрибут стилю **display** досить примітний. Він дозволяє задати вигляд елемента Web-сторінки: буде він блоковим, вбудованим або взагалі пунктом списку.

Приклад:

```
display:none|inline|block|inline-block|list-item|
run-in|table|inline-table|table-caption|table-column|
table-column-group|table-header-group|table-row-group|
table-footer-group|table-row|table-cell|inherit
```

Доступних значень у цього атрибута стилю дуже багато.

- **none** — елемент взагалі не буде відображатися на Web-сторінці, немов він і не заданий в її HTML-коді.
- **inline** — вбудований елемент.
- **block** — блоковий елемент.
- **inline-block** — блоковий елемент, який буде "обтекатися" вмістом сусідніх блокових елементів.
- **list-item** — пункт списку.
- **run-in** — елемент, що *вбудовується*. Якщо за таким елементом розташований блоковий елемент, він стає частиною даного блокового елемента (фактично — вбудованим в нього елементом), а якщо ні, то він сам стає блоковим елементом.
- **table** — таблиця.
- **inline-table** — таблиця, відформатована як вбудований елемент. (Виявляється, ми все-таки можемо помістити таблицю в абзац!)
- **table-caption** — заголовок таблиці.
- **table-column** — колонка таблиці.
- **table-column-group** — група колонок таблиці.

- **table-header-group** — секція заголовка таблиці.
- **table-row-group** — секція тіла таблиці.
- **table-footer-group** — секція завершення таблиці.
- **table-row** — рядок таблиці.
- **table-cell** — комірка таблиці.

Увага! Деякі значення атрибута стилю **display** деякі Браузери можуть не підтримувати.

Значення за замовчуванням атрибута стилю **display** залежить від конкретного елемента Web-сторінки. Так, для абзацу значенням за замовчуванням буде **block**, а для графічного зображення (яке є вбудованим елементом) — **inline.**

От **приклад** комбінованого стилю, що дозволяє графічним зображенням відображатися як блокові елементи:

```
IMG.block {display:block}
```

А от **приклад** стилю, після застосування якого пункти списків стануть вбудованими елементами і будуть виводитися в один рядок:

```
LI {display:inline}
```

Ще один **приклад**:

```
.hidden {display:none}
```

Застосування до елемента Web-сторінки даного стилю робить елемент невидимим. А на самій Web-сторінці навіть не залишиться ніякої ознаки того, що даний елемент на ній був присутній.

У більшості практичних випадків досить значень **none**, **block** і **inline** атрибута стилю **display**. Інші значення занадто специфічні, щоб часто їх застосовувати.

Представлення для нашого Web-сайту, частина 2

Що ж, продовжимо займатися представленням для Web-сторінок нашого Web-сайту. Задамо для них параметри абзаців і списків.

Знову сформулюємо список параметрів, які ми застосуємо до Web-сторінок.

- Вирівнювання тексту в абзацах — повне.

- Вирівнювання тексту в заголовках — по лівому краю. Вирівняні по лівому краю заголовки читаються краще вирівняних по центру.
- Вирівнювання тексту в великих цитатах — по лівому краю. Так ми додатково виділимо цитати.
- Вирівнювання тексту в тегу адреси — по правому краю. Відомості про авторські права найчастіше вирівнюють саме так.
- Маркери в пунктах списків — білий кружок. Додамо стильності нашим спискам.

Виходячи із цього, внесемо у відповідні стилі, визначені в таблиці стилів **main.css** такі зміни.

```
P {font-size:12pt;
   text-align:justify}
```

Фактично ми змінили тільки стиль перевизначення тегу **<P>,** додавши в нього атрибут стилю, що задає повне вирівнювання. Інші стилі залишаться без змін, і у відповідних їм елементах Web-сторінок (в тому числі в заголовках і великих цитатах) вирівнювання буде звичайним — по лівому краю.

Після цього додамо в кінець таблиці стилів наступний CSS-код:

```
ADDRESS {text-align:right}
UL      {list-style-type:circle}
```

Тут ми створили стилі перевизначення тегів **<address>** і **<ul>.** Перший об'єднається з аналогічним стилем, створеним нами в *темі 8,* і доповнить його параметрами вирівнювання (по правому краю). Другий стиль задасть вигляд маркера для пунктів маркірованого списку — білий кружок.

Збережемо таблицю стилів і відкриємо Web-сторінку **index.htm** в Браузері. Зміни Web-сторінці явно пішли на користь.

Створення смуги навігації

Наприкінці створимо для наших Web-сторінок нормальну смугу навігації. Зараз вона в нас занадто простенька.

Смуга навігації може бути горизонтальною або вертикальною, може формуватися в одному абзаці, за допомогою набору абзаців, списку або таблиці. Для набору абзаців кожне гіперпосилання смуги навігації є одним абзацом, для списку — окремим його пунктом, а для таблиці — окремою її коміркою. В цей момент наша смуга навігації є одним абзацом.

Давайте виберемо для формування смуги навігації список — так зараз роблять дуже часто. І відповідно переробимо HTML-код, що формує смугу навігації.

Створюючи першу смугу навігації, ми не додали в неї гіперпосилання, що вказує на Web-сторінку **index.htm**, яка містить власне довідник по HTML. Але правила гарного тону Web-дизайну вимагають, щоб смуга навігації містила гіперпосилання на всі Web-сторінки Web-сайту або, принаймні, на найважливіші. Тому додамо відповідне гіперпосилання в смугу навігації; зробимо це самостійно.

Лістинг 10.2 містить фрагмент HTML-коду Web-сторінки **index.htm**, що формує нову смугу навігації

Лістинг 10.2

```
<UL>
<LI><A HREF="index.htm">HTML</A></LI>
<LI><A HREF="css_index.htm">CSS</A></LI>
<LI><A HREF="samples_index.htm">Приклади</A></LI>
<LI><A HREF="about.htm">0 розроблювачах</A></LI>
 </UL>
```

Одне з найважливіших правил Web-дизайну — смуга навігації повинна відрізнятися від звичайного тексту. А смуга навігації, сформована на основі списку, повинна відрізнятися від звичайного списку. Як це зробити? За допомогою відповідних стилів.

Сформулюємо список параметрів для нашої нової смуги навігації.

- Шрифт — **Arial.** Нехай смуга навігації буде відмінна від звичайного тексту.
- Розмір шрифту — **14** пунктів. Смуга навігації повинна бути помітна.
- Символи шрифту — тільки прописні. Так смуга навігації стане ще помітніше.
- Маркер — відсутній. У смузі навігації маркери не потрібні.

Оскільки смуга навігації буде присутня на кожній Web-сторінці в єдиному екземплярі, для її оформлення ми застосуємо іменований стиль. Дамо йому ім'я **navbar.** CSS-код наведено в лістингу 10.3.

Лістинг 10.3

```
#navbar {font-family:Arial, sans-serif;
         font-size:14pt;
```

```
text-transform:uppercase;
list-style-type:none}
```

Помістимо цей код в самому кінці таблиці стилів **main.css**.

Щоб прив'язати іменований стиль до тегу, слід указати його ім'я як значення атрибута тегу **id**:

```
<UL ID="navbar">
```

Залишилося зберегти Web-сторінку і таблицю стилів та перевірити результат, що вийшов, в Браузері. Що ж, нова смуга навігації помітно краще старої.

Параметри курсору

CSS дає нам одну дуже цікаву можливість — вказівка вигляду курсору миші, який він прийме при наведенні на даний елемент Web-сторінки. Це може бути корисно при створенні спеціальних ефектів.

Атрибут стилю **cursor** встановлює форму курсору миші при наведенні його на даний елемент Web-сторінки. Даний атрибут можна застосувати до будь-якого елемента Web-сторінки, як блокового, так і вбудованого:

```
cursor:auto|default|none|context-menu|help|pointer|
progress|wait|cell|crosshair|text|vertical-text|alias|
copy|move|no-drop|not-allowed|e-resize|n-resize|
ne-resize|nw-resize|s-resize|se-resize|sw-resize|
w-resize|ew-resize|ns-resize|nesw-resize|nwse-resize|
col-resize|row-resize|all-scroll|inherit
```

Як бачимо, можливих значень у атрибута **cursor** дуже багато, до того ж багато з них на практиці застосовуються вкрай рідко. Тому ми розглянемо тільки самі необхідні.

- **auto** — Браузер сам управляє формою курсору миші. Це звичайна поведінка.
- **default** — курсор за замовчуванням, звичайно стрілка.
- **none** — курсор миші взагалі не відображається.
- **help** — стрілка зі знаком питання.
- **pointer** — "перст, що вказує". Звичайна поведінка при наведенні курсору миші на гіперпосилання.

- **`progress`** — стрілка з невеликим пісковим годинником. Позначає, що в цей момент працює якийсь фоновий процес.
- **`wait`** — пісковий годинник. Позначає, що в цей момент Браузер зайнятий.
- **`text`** — текстовий курсор. Звичайна поведінка при наведенні курсору миші на фрагмент тексту.

Повний список значень атрибута стилю **`cursor`** і опис відповідної їм форми курсору миші ви можете знайти на Web-сторінці https://developer.mozilla.org/en/CSS/cursor**.**

От **приклад** завдання курсору миші у вигляді "перста, що вказує" для пунктів списку, що формує тільки що створену смугу навігації:

```
#navbar LI {cursor:pointer}
```

Контрольні питання

1. Як задати горизонтальне вирівнювання тексту?
2. Як задати відступ для "червоного рядка"?
3. Які параметри списків ви знаєте?
4. Які види маркерів для списків ви знаєте? Як вони задаються?
5. Що таке параметри відображення?
6. Які параметри відображення ви знаєте?
7. Які значення атрибута стилю **`display`** найчастіше використовуються?
8. Як створити смугу навігації?
9. Які параметри курсору ви знаєте?
10. Як задати виду курсору миші, який він прийме при наведенні на даний елемент Web-сторінки?

ТЕМА 11. КОНТЕЙНЕРНИЙ WEB-ДИЗАЙН

***Основна мета*:** отримати знання про різні контейнери: блокові контейнери, контейнери, що плавають, контейнери із прокручуванням, а також знання по контейнерному Web-дизайнну і атрибутам стилю, що задають різні параметри контейнерів.

КОНТЕЙНЕРНИЙ WEB-ДИЗАЙН

Блокові контейнери

Як зрозуміло із назви, *блоковий* контейнер може зберігати тільки блокові елементи: абзаци, заголовки, великі цитати, теги адреси, текст фіксованого форматування, таблиці, аудіо- і відеоролики. Блоковий контейнер може містити як один блоковий елемент, так і дещоекілька.

Блоковий контейнер формується за допомогою парного тегу **<DIV>,** усередині якого поміщають HTML-код, що формує вміст контейнера (лістинги 11.1—11.3).

В лістингу 11.1 ми помістили в блоковий контейнер заголовок і два абзаци.

Лістинг 11.1

```
<DIV>
```

```
   <H3>Це заголовок</H3>
   <P>Це перший абзац</P>
   <P>Це другий абзац</P>
</DIV>
```

Лістинг 11.2 ілюструє блоковий контейнер, що містить таблицю.

Лістинг 11.2

```
<DIV>
   <TABLE>
      <CAPTION>Це таблиця</CAPTION>
      <TR>
         <TH>Це перша колонка</TH>
         <TH>Це друга колонка</TH>
      </TR>

      ..................
   </TABLE>
</DIV>
```

А блоковий контейнер, приведений в лістингу 11.3, може похвастатися самим "багатим" вмістом.

Лістинг 11.3

```
<DIV STYLE="text-align: center">
   <VIDEO SRC="film.ogg" CONTROLS>
   </VIDEO>
   <P>Клацніть кнопку відтворення, щоб переглянути фільм</P>
</DIV>
```

По-перше, ми помістили в нього відеоролик і абзац із текстом, що пропонує почати перегляд цього відеоролика. По-друге, ми прив'язали до блокового контейнера вбудований стиль, що задає вирівнювання тексту по центру. Відзначимо при цьому, що по центру також буде вирівняний і сам відеоролик.

Блокові контейнери застосовують значно частіше, чим вбудовані. Саме на блокових контейнерах заснований контейнерний Web-дизайн, про який зараз піде розмова.

Основи контейнерного Web-дизайну

Контейнерний Web-дизайн відомий уже досить давно. В теперешній час він поступово витісняє більш старі різновиди Web-дизайну. І тому є багато причин.

Старі різновиди Web-дизайну, їх переваги переваги і недоліки

Раніше в Інтернеті панували три різновиди Web-дизайну: текстовий, фреймовий і табличний. Кожний спосіб мав свої переваги і недоліки. Але всі тією чи іншою мірою програють четвертому різновиду Web-дизайну — контейнерному.

Першим з'явився, мабуть, *текстовий Web*-дизайн. Виконані в такий спосіб Web-сторінки були звичайними текстовими документами: набір абзаців, заголовків, великих цитат, тексту фіксованого форматування і таблиць, що ідуть один за одним. Класичний приклад подібного Web-дизайну — створені нами до даного моменту Web-сторінки. Відкрийте в Браузері, скажемо, Web-сторінку **index.htm** — і ви побачите текстовий Web-дизайн.

Перевага текстового Web-дизайну всього одна — виняткова простота HTML-коду. Дійсно, код таких Web-сторінок містить один тільки текст і, можливо, зображення і таблиці. Ніяких специфічних елементів, що формують власно Web-дизайн там немає.

Недоліків же у текстового Web-дизайну багато. По-перше, створені на його основі Web-сторінки виглядають занадто невибагливо. По-друге, практично відсутні засоби довільного розміщення елементів на Web-сторінці — вони можуть тільки розміщатися один за одним зверху вниз. По-третє... а от про це слід поговорити докладніше.

Інформацію, представлену на Web-сторінці, можна грубо розділити на чотири фрагменти: заголовок Web-сайту, смугу навігації, *основний вміст* (інформація, унікальна для конкретної Web-сторінки, та, заради якої саме ця Web-сторінка і створювалася) і відомості про авторські права. Як правило, ці фрагменти візуально відділені друг від друга, так що знайти їх не важко.

На всіх Web-сторінках, що складають Web-сайт, заголовок Web-сайту, смуга навігації і відомості про авторські права однакові. І тільки основний вміст у кожної Web-сторінки унікально по визначенню.

Але ж і заголовок Web-сайту, і смуга навігації, і відомості про авторські права визначаються в HTML-коді кожної Web-сторінки. І код цей може бути дуже об'ємним. І що виходить? Значна частина HTML-коду кожної Web-сторінки визначає елементи, які не змінюються від однієї Web-сторінки до іншої!

Чим об'ємніше HTML-код Web-сторінки, тим більше файл, в якому вона зберігається. Чим значніше розмір файлу, тем довше він завантажується. Чим довше файл завантажується, тим більше доведеться відвідувачу чекати, поки запитана Web- сторінка з'явиться на екрані.

Чи немає способу завантажити не всю Web-сторінку цілком, а тільки її частину — власне основний вміст? На жаль, текстовий Web-дизайн такого способу не пропонує...

Але вихід, проте, був знайдений у вигляді "нестандартного" розширення HTML — фреймів. *Фрейм* — це особливий елемент Web-сторінки, який виводить в указаному її місці вміст довільної Web-сторінки, інтернет-адреса якої задається в його параметрах. Крім того, були розширені можливості гіперпосилань — тепер вони могли виводити цільову Web-сторінку в указаному фреймі.

Головна Web-сторінка Web-сайту в такому варіанті являла собою набір фреймів. Окремі фрагменти її вмісту виносилися в окремі Web-сторінки, інтернет-адреси яких вказувалися в параметрах відповідних їм фреймів. Інші Web-сторінки містили в собі тільки основний вміст. А в параметрах гіперпосилань смуги навігації вказувалося, в якому фреймі повинні завантажуватися цільові Web-сторінки.

Подібний Web-дизайн отримав назву *фреймового.* Він мав незаперечну перевагу — різке збільшення швидкості завантаження Web-сторінок. І тому широко застосовувався багато років.

Однак у фреймів є істотні недоліки. По-перше, фрейми так і не були стандартизовані комітетом W3C, тому кожний Браузер обробляє їх по-своєму, не в цілому, звичайно, а в нюансах, які, проте, можуть бути істотними. По-друге, фрейми — дуже негнучкий елемент Web-сторінки; їхню структуру неможливо поміняти.

Конкурентом фреймового Web-дизайну став *табличний*, що з'явився трохи пізніше. Для формування Web-сторінки використовувалася велика таблиця HTML, у різні комірки якої поміщали заголовок Web-сайту, смугу навігації, різні фрагменти основного вмісту і відомості про авторські права. Поступово табличний Web-дизайн витіснив фреймовий; дотепер це самий популярний спосіб створення Web-сайтів.

Переваги табличного Web-дизайну:

- Таблиці — стандартна частина мови HTML, і виходить, можна добитися того, щоб засновані на них Web-сторінки відображалися однаково у всіх Браузерах.

- Таблиці HTML можна робити як завгодно складними, об'єднуючи їх комірки і вкладаючи одні таблиці в інші. Це дозволяє робити дуже складні Web-сторінки, що містять різнорідні фрагменти вмісту, які мають декілька колонок тексту і більше схожі на газети.
- Таблиці і їх окремі комірки можна легко форматувати за допомогою стилів CSS, задаючи для них рамки, відступи, фон, вирівнювання та інші параметри.

Однак табличний Web-дизайн має і безлічагато недоліків:

- Все та ж "монолітність" Web-сторінок, що і у випадку текстового Web-дизайну. Кожна Web-сторінка Web-сайту містить і його заголовок, і смугу навігації, і основний вміст, і відомості про авторські права, що не кращим чином позначається на її розмірах та на швидкості її завантаження.
- Для формування складних таблиць застосовується надзвичайно громіздкий і заплутаний HTML-код.
- Старі версії Браузерів не дуже вдало реалізовували обробку таблиць: вони спочатку завантажували таблицю цілком, а вже потім виводили її на екран. Враховуючи, що таблиці, за допомогою яких формувалися Web-сторінки, дуже великі, завантаження таких Web-сторінок забирала багато часу. Сучасні Браузери можуть виводити таблицю на екран в процесі її завантаження. Так що остання проблема відпала.

Як бачимо, всі три старі різновиди Web-дизайну, поряд з перевагами, мають і серйозні недоліки. Тому в цей час вони повільно, але вірно відступають під тиском контейнерного Web-дизайну/

Сутність контейнерного Web-дизайну

Контейнерний Web-дизайн для розміщення окремих фрагментів вмісту Web-сторінок використовує блокові контейнери, з якими ми познайомилися на початку цієї теми. Окремі контейнери створюються для заголовка Web-сайту, смуги навігації, основного вмісту і відомостей про авторські права. Якщо основний вміст має складну структуру і складається багатьох незалежних частин, для кожної з них створюють окремий контейнер.

Для завдання різних параметрів блокових контейнерів передбачені спеціальні атрибути стилю CSS. До таких параметрів належать розміри (ширина і висота), місце розташування контейнерів та їх поведінка при переповненні. Також ми можемо задати для контейнерів колір фона, створити відступи і рамки, щоб їх виділити.

Розглянемо недоліки трьох старих принципів Web-дизайну та з'ясуємо, чи зможе контейнерний Web-дизайн їх усунути.

- Невибагливий вид і лінійне представлення Web-сторінок — у текстового Web-дизайну. Ми можемо розташувати контейнери на Web-сторінці практично як завгодно і помістити в них довільний вміст: текст, таблиці, зображення, аудіо- і відеоролики та навіть інші контейнери. А CSS дозволить нам задати для них практично будь-яке представлення.
- "Монолітність" Web-сторінок — у текстового і табличного Web-дизайну. Сучасні Браузери дозволяють за допомогою спеціально створеної поведінки завантажити в контейнер Web-сторінку, збережену в окремому файлі, тобто організувати підвантажуваний вміст.
- "Нестандартність" фреймів — у фреймового Web-дизайну. Контейнери і відповідні теги офіційно стандартизовані комітетом W3C і обробляються всіма Браузерами однаково.
- Громіздкість HTML-коду — у табличного Web-дизайну. HTML-код, що формує контейнери, винятково компактний. Як ми вже знаємо, блоковий контейнер формується всього одним парним тегом <DIV>.
- Повільне завантаження Web-сторінок — у табличного Web-дизайну. Всі Браузери відображають вміст контейнерів прямо в процесі завантаження, так що Web-сторінки візуально завантажуються дуже швидко.

Недолік контейнерного Web-дизайну:

Контейнерний Web-дизайн програє табличному в можливості реалізації складного дизайну Web-сторінок. Таблиця дозволяє створити на Web-сторінці багато колонок різної ширини, що містять різний вміст. Щоб зробити це за допомогою контейнерів, швидше за все, доведеться застосовувати вкладені друг у друга контейнери, складні стилі і, можливо, поведінку, яка вже після закінчення завантаження Web-сторінки розташовує контейнери в потрібних місцях. Це, мабуть, єдиний недолік контейнерного Web-дизайну.

Переробимо наші Web-сторінки із застосуванням контейнерного Web-ди- зайну.

Представлення для нашого Web-сайту, частина 3

Спочатку розробимо схему розташування на Web-сторінках відповідних контейнерів. Найкраще намалювати її на папері або в програмі графічного редактора.

Класична схема Web-дизайну (і не тільки контейнерного) показано на рис. 11.1. Як бачимо, в самому верху знаходиться заголовок Web-сайту, нижче його в одну лінію по горизонталі розташувалися смуга навігації і основний вміст, а під ними прилаштувалися відомості про авторські права. Давайте використовуємо саме цю схему.

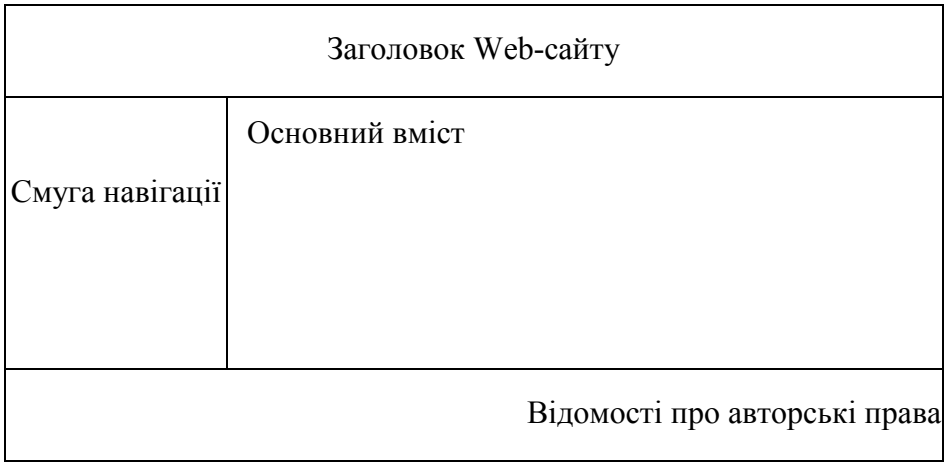


Рис. 11.1. **Класична схема Web-дизайну**

Відкриємо в Блокноті Web-сторінку **index.htm**. Знайдемо в її HTML-коді чотири важливі фрагменти будь-якої Web-сторінки: заголовок Web-сайту, смугу навігації, основний вміст і відомості про авторські права. Помістимо їх у блокові контейнери.

На рис. 11.1 показано, що заголовок Web-сайту передує смузі навігації, а не навпаки. Тому поміняємо місцями фрагменти HTML-коду, що створюють ці контейнери та їх вміст.

Згодом ми прив'яжемо до створених контейнерів стилі, що задають їхні розміри та місце розташування на Web-сторінці. Оскільки кожний із цих контейнерів присутній на кожній Web-сторінці в єдиному екземплярі, застосуємо для цього іменовані стилі. Дамо їм такі імена:

- **`cheader`** — стиль для контейнера із заголовком Web-сайту;
- **`cnav`** — стиль для контейнера із смугою навігації;
- **`cmain`** — стиль для контейнера з основним вмістом;
- **`ccopyright`** — стиль для контейнера з відомостями про авторські права.

Тут буква "**c**" позначає "**container**" — контейнер. Так ми відразу зрозуміємо, що дані стилі застосовуються саме до контейнерів.

Прив'язка іменованого стилю до тегу виконується шляхом вказівки імені цього стилю як значення атрибута тегу **`ID`**. Зробимо це для всіх контейнерів.

В лістингу 11.4 приведений фрагмент HTML-коду Web-сторінки **index.htm** з усіма потрібними виправленнями.

Лістинг 11.4

```
<DIV ID="cheader">
   <H1>Довідник по HTML і CSS</H1>
</DIV>
<DIV ID="cnavbar">
   <UL ID="navbar">
      <LI><A HREF="index.htm">HTML</A></LI>
      <LI><A HREF="css_index.htm">CSS</A></LI>
      <LI><A HREF="samples_index.htm">Приклади</A></LI>
      <LI><A HREF="about.htm"> Про розроблювачів</A></LI>
   </UL>
</DIV>
<DIV ID="cmain">
   <P>Вітаємо на нашім Web-сайті всіх, хто займається
   Web-дизайном! Тут ви зможете знайти інформацію про всі
   інтернет-технології, застосовувані при створенні Web-
   сторінок. А саме, про мови <DFN>HTML</DFN> і
   <DFN>CSS</DFN>.</P>
   <HR>
   <H2>Теги HTML</H2> <P><CODE>!DOCTYPE</CODE>,
   <CODE><A HREF="tags/t_audio.htm">AUDIO</A></CODE>,
   <CODE>BODY</CODE>,                    <CODE>EM</CODE>,
   <CODE>HEAD</CODE>, <CODE>HTML</CODE>,
   <CODE><A HREF="tags/t_img.htm">IMG</A></CODE>,
   <CODE>META</CODE>, <CODE>P</CODE>,
   <CODE>STRONG</CODE>,
   <CODE><A HREF="tags/t_title.htm">TITLE</A></CODE>,
   <CODE><A HREF="tags/t_video.htm">VIDEO</A></CODE></P>
</DIV>
<DIV ID="ccopyright">
  <ADDRESS>Все права захищені<BR>&#0169;
  <A HREF="mailto:user@mailserver.ru">студенти</A>, 2019
   рік</ADDRESS>
</DIV>
```

Збережемо Web-сторінку **index.htm** і відкриємо її в Браузері. Нічого не змінилося в порівнянні з тим, що було раніше.

Стилі, що задають параметри контейнерів

Контейнери, і блокові, і вбудовані, ніяк не відображаються в Браузері.

Щоб відчути користь від контейнерів, ми повинні застосувати до них стилі. Саме для цього контейнери і призначені. Тому зараз ми займемося атрибутами стилю, що задають параметри контейнерів. Атрибути стилю і параметри, що задаються ними діляться на дві групи.

Параметри розмірів

Атрибути стилю першої групи задають розміри контейнерів. Власне, їх можна застосовувати не тільки в контейнерах, але і в будь-яких інших блокових елементах.

Атрибути стилю **width i height** дозволяють задати, відповідно, ширину і висоту елемента Web-сторінки:

```
width:auto|<ширина>|inherit
height:auto|<висота>|inherit
```

Ми можемо вказати абсолютне значення розміру, скориставшись кожної з доступних в CSS одиниць виміру. Тоді розмір елемента буде незмінним:

```
#cnavbar {width:100px}
```

Можна задати відносне значення розміру у відсотках від відповідного розміру батьківського елемента. При цьому розмір елемента буде змінюватися при зміні розмірів вікна Браузера:

```
#cheader {height:10%}
```

Значення **auto** віддає керування цим розміром на відкуп Браузеру (звичайна поведінка). В останньому випадку Браузер установить такі розміри елемента Web-сторінки, щоб в ньому повністю помістився його вміст, і не більше.

Якщо ми призначимо для якого-небудь елемента Web-сторінки відносну ширину або висоту, нам можуть пригодитися атрибути стилю, що вказують мінімальні і максимальні можливі розміри для цього елемента. Зрозуміло, що при їхньому завданні значення розміру не зможе перевищити максимальне і стати менше мінімального.

Атрибути стилю **min-width i min-height** дозволяють визначити мінімальну ширину і висоту елемента Web-сторінки:

```
min-width: <ширина>
```

min-height: *<висота>*

Значення ширини або висоти може бути як абсолютним, так і відносним.

Приклад:

```
TABLE {min-width:500px;
       min-height:300px}

TABLE {min-width:90%;
       min-height:60%}
```

Аналогічні атрибути стилю **max-width** і **max-height** дозволяють задати максимальну, відповідно, ширину і висоту елемента Web-іторінки:

max-width:*<ширина>*
max-height:*<висота>*

І тут значення ширини або висоти може бути як абсолютним, так і відносним:

```
TABLE {max-width:90%;
       max-height:60%}
```

Параметри розміщення. Плаваючі контейнери

Місце розташування блокових контейнерів (і будь-яких інших блокових елементів) на Web-сторінці визначають два досить важливі атрибути стилю.

Початково блокові елементи Web-сторінки розташовуються на ній по вертикалі, строго один за одним, в тому порядку, в якому вони визначені в HTML-коді. Саме так розташовуються блокові контейнери, абзаци і заголовки на всіх створених нами Web-сторінках.

Однак ми можемо встановити для блокового елемента вирівнювання по лівому або правому краю батьківського елемента (блокового контейнера, в який він вкладен, або самої Web-сторінки). При цьому блоковий елемент буде притискатися до відповідного краю батька, а решта вмісту буде обтікати його із протилежної сторони.

Атрибут стилю **float** саме і задає, по якому краю батьківського елемента буде вирівнюватися даний елемент Web-сторінки:

float:left|right|none|inherit

Можливі три значення:

- **left** — елемент Web-сторінки вирівнюється по лівому краю батьківського елемента, а решта вмісту обтікає його праворуч;
- **right** — елемент Web-сторінки вирівнюється по правому краю батьківського елемента, а решта вмісту обтікає його ліворуч;
- **none** — звичайна поведінка елемента Web-сторінки, коли він іде за своїм попередником і розташовується нижче його.

Приклад:

```
TABLE.left-aligned {float:left }
```

Після застосування даного стилю до таблиці вона буде вирівняна по лівому краю батьківського елемента, а решта вмісту буде обтікати її праворуч.

А що буде, якщо ми задамо однакове значення атрибута стилю float для декількох розташованих один за одним блокових елементів? Вони розташуються по горизонталі один за одним в тому порядку, в якому вони визначені в HTML-коді, і будуть вирівняні по зазначеному краю батьківського елемента.

Даний атрибут стилю звичайно застосовують для блокових контейнерів, що формують дизайн Web-сторінок. Такі контейнери називають *плаваючими.*

При створенні контейнерного Web-дизайну, заснованого на плаваючих контейнерах, часто приходиться поміщати які-небудь контейнери нижче тих, що були вирівняні по лівому або правому краю батьківського елемента. Якщо просто видалити в них атрибут стилю float або задати для нього значення none, результат буде непередбаченим.

Тому CSS надає можливість однозначно вказати, що даний блоковий елемент обов'язково повинен розташовуватися нижче плаваючих контейнерів, що передують йому. Для цього служить **атрибут стилю `clear`:**

```
clear: left|right|both|none|inherit
```

Як бачимо, доступних значень тут чотири:

- **left** — елемент Web-сторінки повинен розташовуватися нижче всіх елементів, для яких у атрибута стилю float задане значення left;
- **right** — елемент Web-сторінки повинен розташовуватися нижче всіх елементів, для яких у атрибута стилю float задане значення right;
- **both** — елемент Web-сторінки повинен розташовуватися нижче всіх елементів, для яких у атрибута стилю float задане значення left або right;

- **none** — звичайна поведінка. Якщо контейнеру, для якого указаний даний атрибут стилю, передують плаваючі контейнер, задавати це значення не рекомендується.

Приклад:

```
#ccopyright {clear:both}
```

Тут ми задали для іменованого стилю **ccopyright** (він застосовується до контейнера, що містить відомості про авторські права) розташування нижче всіх контейнерів, вирівняних по лівому або правому краю батьківського елемента.

Представлення для нашого Web-сайту, частина 4

Отриманих нами знань уже достатньо для того, щоб створити контейнерний дизайн для нашого Web-сайту. Давайте займемося цим.

Як звичайно, випишемо список параметрів для всіх створених раніше контейнерів.

Для контейнера із заголовком Web-сайту (**cheader**):

- ширина — 100% (все вікно Браузера).

Для контейнера зі смугою навігації (**cnavbar**):

- ширина — 30% (приблизно третина ширини вікна Браузера);
- мінімальна ширина — 240 пікселів (це значення приблизно дорівнює ширині смуги навігації);
- вирівнювання — по лівому краю (тобто це буде плаваючий контейнер).

Для контейнера з основним умістом (**cmain**):

- ширина — 70% (приблизно дві третини ширини вікна Браузера);
- вирівнювання — по лівому краю.

Для контейнера з відомостями про авторські права (**ccopyright**):

- ширина — 100% (все вікно Браузера);
- розташування — нижче всіх плаваючих контейнерів, вирівняних по лівому і правому краям.

Як бачимо, у жодного контейнера явно не задана висота. Браузер сам встановить її такою, щоб контейнер при указаній ширині повністю вмістив свій вміст.

На основі перерахованих вимог напишемо CSS-код, що визначає потрібні стилі (лістинг 11.5).

Лістинг 11.5

```
#cheader {width:100%}
#cnavbar {width:30%;
          min-width:240px;
          float:left}
#cmain   {width:70%;
          float: left }
#ccopyright {width:100%;
          clear:both}
```

Помістимо цей код в самий кінець таблиці стилів **main.css**, після чого збережемо її. Відкриємо Web-сторінку **index.htm** в Браузері та подивимося, що вийшло (рис. 11.2).

От вона — наша Web-сторінка, виконана по канонах сучасного Web-дизайну.

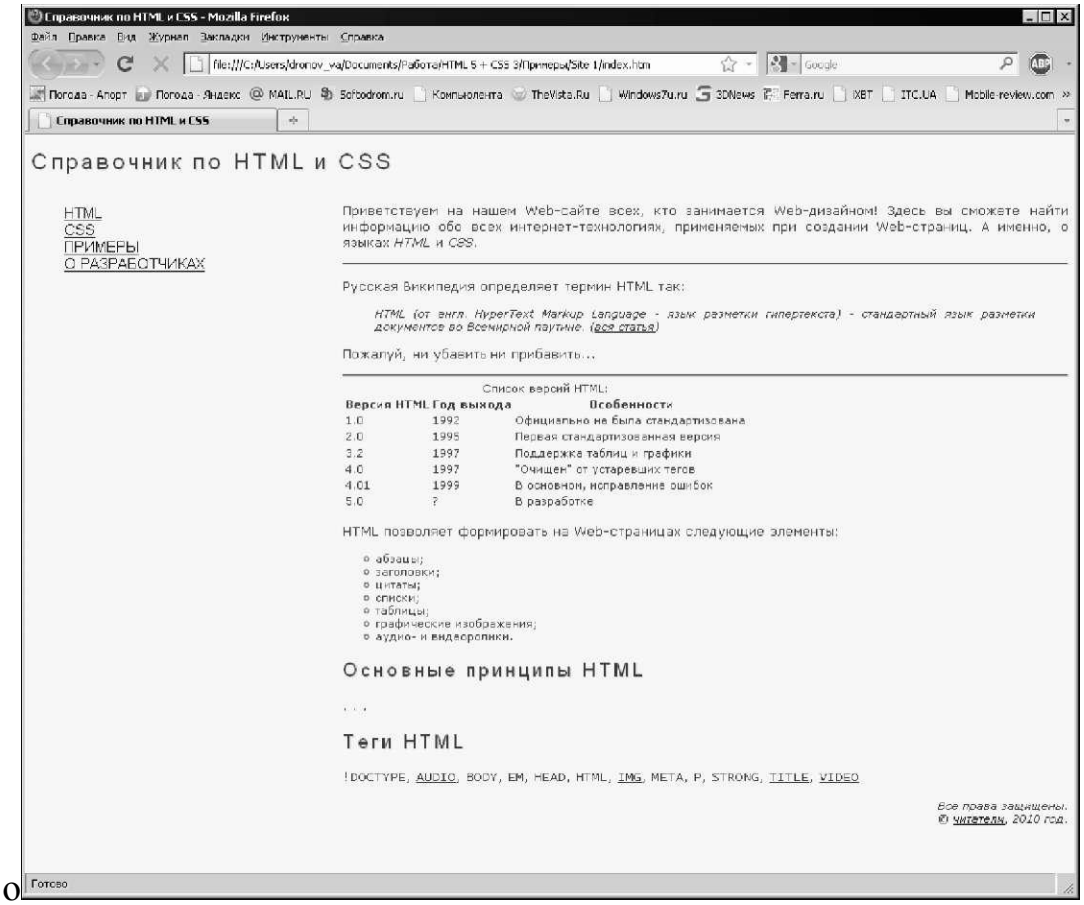


Рис. 11.2. **Web-сторінка index.htm, виконана на основі контейнерного Web-дизайну, в Браузері**

Параметри переповнення. Контейнери із прокручуванням

У жодного з контейнерів, що формують дизайн нашої Web-Сторінки, ми не задали явну висоту. Браузер сам встановить для контейнерів такі значення висоти, щоб вони вмістили свій вміст повністю.

Але що трапиться, якщо ми задамо для контейнера висоту? Тоді може вийти так, що вміст контейнера не поміститься в ньому, і виникне *переповнення контейнера.* Поведінка контейнера залежить від параметрів, які ми задамо для нього.

Атрибут стилю **overflow** саме і задає поведінку контейнера при переповненні:

overflow:visible|hidden|scroll|auto|inherit

Тут доступні чотири значення:

- **visible** — висота контейнера збільшиться, щоб повністю вмістити весь вміст (звичайна поведінка);
- **hidden** — вміст, що не поміщається в контейнер, буде обрізано. Контейнер збереже свої розміри;
- **scroll** — в контейнері з'являться смуги прокручування, за допомогою яких можна переглянути частину вмісту, що не поміщається. Ці смуги прокручування будуть присутні в контейнері завжди, навіть якщо в них немає потреби;
- **auto** — смуги прокручування з'являться в контейнері, тільки якщо в них виникне необхідність.

Приклад:

```
#cmain {overflow:auto}
```

Ми задали для контейнера **cmain** таку поведінку, коли при виході вмісту за межі контейнера в ньому з'являться смуги прокручування.

Тут потрібно сказати наступне. Атрибут стилю **overflow** має сенс тільки в тому випадку, якщо ми задамо для висоти контейнера абсолютне значення. При завданні відносного значення висоти контейнера він завжди буде збільшуватися в розмірах для того, щоб вмістити весь вміст, начебто для атрибута стилю **overflow** задане значення **visible**:

```
#cmain {height:500px;
        overflow:auto}
```

От тепер контейнер **cmain** при необхідності обзаведеться смугами прокручування.

А в наступному прикладі атрибут стилю **overflow** можна не вказувати — контейнер **cmain** завжди буде поводитися так, начебто для згаданого атрибута стилю задане значення **visible:**

```
#cmain {height:50%;
        overflow:auto}
```

Атрибути стилю **overflow-x i overflow-y** задають поведінку контейнера при виході вмісту за межі його границь, відповідно, по горизонталі і вертикалі. Доступні значення в них ті ж, що і в атрибута стилю **overflow:**

```
#cnavbar {width:270px;
          overflow-x:hidden }
```

Користуючись тільки що вивченими атрибутами стилю, ми можемо створити на Web-сторінці *контейнери із прокручуванням.* Це звичайні контейнери з великим вмістом, для перегляду якого передбачені смуги прокручування. Їхня перевага в тому, що відвідувач, прокручуючи вміст такого контейнера, не торкається всіх інших фрагментів Web-сторінки (заголовок Web-сайту, смуга навігації та відомості про авторські права). Досить зручно.

Представлення для нашого Web-сайту, частина 5

Давайте створимо контейнер із прокручуванням для виведення основного вмісту Web-сторінки. Точніше, переробимо вже створений контейнер **cmain.**

Що нам для цього буде потрібно? По-перше, задати для даного контейнера абсолютне значення висоти, а краще — і ширини, і висоти. По-друге, визначити відповідну поведінку при переповненні.

При завданні абсолютного значення ширини для контейнера **cmain** потрібно встановити абсолютне значення ширини і у контейнера **cnavbar.** Змішувати абсолютні та відносні значення ширини в плаваючих контейнерів не рекомендується — результат буде непередбаченим.

Випишемо список параметрів контейнерів cnavbar і **cmain.**

Для контейнера **cnavbar:**

- ширина — 240 пікселів;
- висота — 620 пікселів;
- вирівнювання — по лівому краю.

Для контейнера **cmain:**

- ширина — 780 пікселів;
- висота — 620 пікселів;
- вирівнювання — по лівому краю;
- поведінка при переповненні — поява смуг прокручування.

Розміри контейнерів потрібно вибрати такі, щоб простір Web-сторінки було зайнято максимально повно, а смуги прокручування у самої Web-сторінки були відсутні.

Крім того, ми задали однакову висоту в обох контейнерів — і **cnavbar**, і **cmain.** Це потрібно для того, щоб виключити деякі небажані ефекти, які можуть виникнути, якщо ми створимо у контейнерів рамки або змінимо колір фона.

У лістингу 11.6 приведений виправлений фрагмент таблиці стилів **main.css, що** створює стилі, відповідні до контейнерів **cnavbar** і **cmain.**

Лістинг 11.6

```
#cnavbar  {width:240px;
           height:620px;
           float:left }
#cmain    {width:780px;
           height:620px;
           float:left;
           overflow:auto}
```

Внесемо ці виправлення в таблицю стилів, збережемо її й перевіримо, що вийшло. А вийшло в нас те, що показано на рис. 11.3.

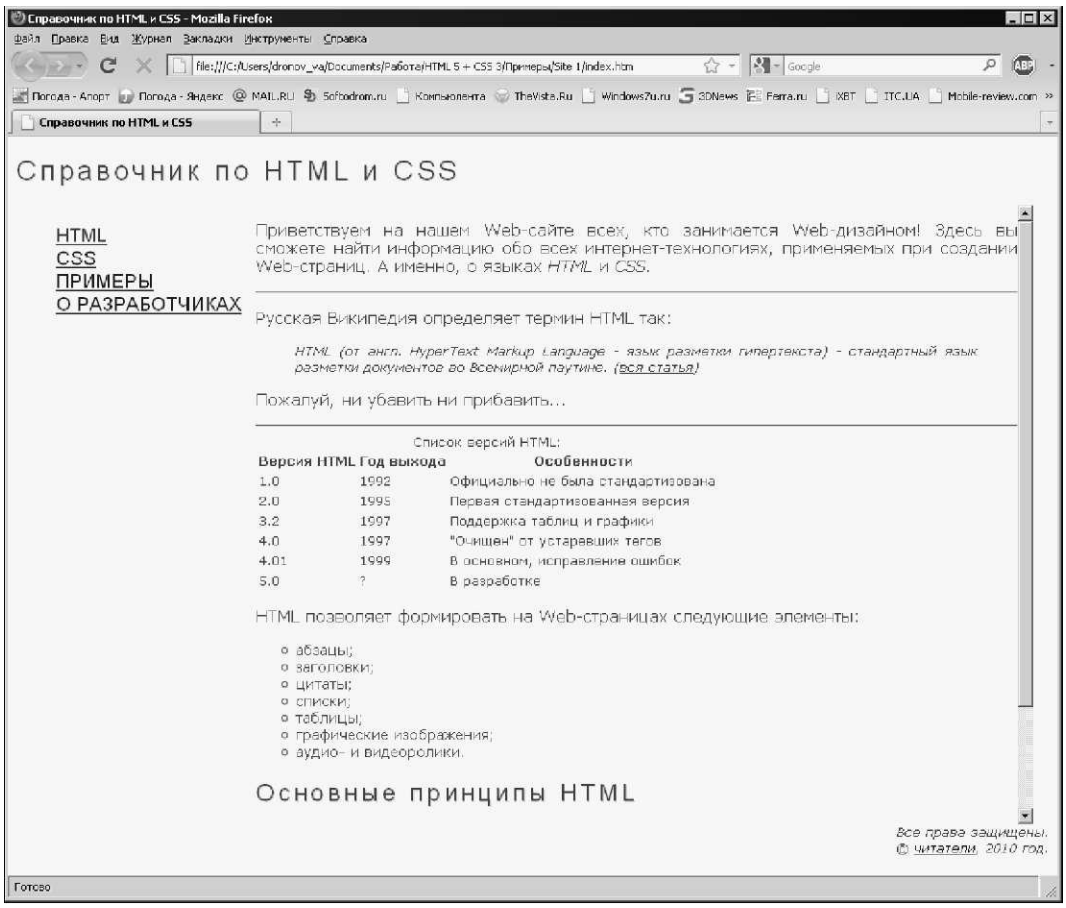


Рис.11.3. **Web-сторінка index.htm, що містить контейнер із прокручуванням, в Браузері**

КОНТРОЛЬНІ ПИТАННЯ

1. Що таке блокові контейнери?
2. Текстовий, фреймовий і табличний Web-дизайн, їх переваги і недоліки.
3. Сутність контейнерного Web-дизайну.
4. Порівняйте табличний і контейнерний Web-дизайни.
5. Дві групи стилів, що задають параметри контейнерів.
6. Параметри розмірів для контейнерів.
7. Параметри розміщення для контейнерів. Плаваючі контейнери.
8. Параметри переповнення. Контейнери із прокручуванням.
9. Як створити смугу навігації?
10. Наведіть класичну схему Web-дизайну.

ТЕМА 12. ВІДСТУПИ, РАМКИ І ВИДІЛЕННЯ

***Основна мета*:** навчитися створювати відступи рамки і виділення елементів Web-сторінки.

ВІДСТУПИ, РАМКИ І ВИДІЛЕННЯ

Розберемося з атрибутами стилю, за допомогою яких задають параметри відступів і рамок. І доробимо нарешті нашу Web-сторінку.

Параметри відступів

Стандарт CSS пропонує засоби для створення відступів двох видів.

1. Відступ між уявлюваною границею елемента Web-сторінки і його вмістом — *внутрішній* відступ. Такий відступ належить даному елементу Web-сторінки, знаходиться всередині нього.
2. Відступ між уявлюваною границею даного елемента Web-сторінки і уявлюваними границями сусідніх елементів Web-сторінки — *зовнішній* відступ. Такий відступ не належить даному елементу Web-сторінки, знаходиться поза ним.

Щоб краще зрозуміти різницю між внутрішнім і зовнішнім відступами, давайте розглянемо комірку таблиці. Комірка наповнена вмістом, скажемо, текстом, має уявлювану границю і оточена іншими комірками.

- Внутрішній відступ — це відступ між границею комірки і текстом, що міститься в ній.
- Зовнішній відступ — це відступ між границями окремих комірок таблиці.

Атрибути стилю **`padding-left`**, **`padding-top`**, **`padding-right`** і **`padding-bottom`** дозволяють задати величини внутрішніх відступів, відповідно, ліворуч, зверху, праворуч і знизу елемента Web-сторінки:

padding-left|padding-top|padding-right|padding-bottom: <*відступ*>|auto|inherit

Ми можемо вказати як величину відступу абсолютне або відносне значення. Значення **auto** задає величину відступу за замовчуванням, звичайно воно дорівнює нулю.

В лістингу 12.1 ми вказали внутрішній відступ для комірок таблиці, рівний двом пікселам з усіх боків.

Лістинг 12.1

```
TD, TH {padding-left:2px;
        padding-top:2px;
        padding-right:2px;
        padding-bottom:2px}
```

А от стиль, що створює внутрішні відступи, рівні двом сантиметрам ліворуч і праворуч:

```
.indented {padding-left:2cm;
           padding-right:2cm}
```

Ми можемо прив'язати такий стиль до абзацу і подивитися, що вийде.

Атрибут стилю **padding** дозволяє відразу вказати величини внутрішніх відступів з усіх боків елемента Web-сторінки:

padding:<*відступ 1*> [<*відступ 2*> [<*відступ 3*> [<*відступ 4*>]]]

- Якщо вказано одне значення, воно задасть величину відступу з усіх боків елемента Web-сторінки.
- Якщо вказано два значення, перше установить величину відступу зверху і знизу, а друге — ліворуч і праворуч.
- Якщо вказано три значення, перше визначить величину відступу зверху, друге — ліворуч і праворуч, а третє — знизу.
- Якщо вказано чотири значення, перше задасть величину відступу зверху, друге — праворуч, третє — знизу, а четверте — ліворуч.

Приклад:

```
TD, TH {padding:2px}
.indented {padding:0cm 2cm 0cm 2cm}
```

Тут ми просто переписали визначення наведених раніше стилів з використанням атрибута стилю **padding**.

Атрибути стилю **margin-left, margin-top, margin-right** і **margin-bottom** дозволяють задати величини зовнішніх відступів, відповідно, ліворуч, зверху, праворуч і знизу:

margin-left|margin-top|margin-right|margin-bottom: *<відступ>*|auto|inherit

Тут також доступні абсолютні і відносні значення. Значення **auto** задає величину відступу за замовчуванням, як правило, рівне нулю.

Приклад:

```
H1 {margin-top:5mm}
```

Цей стиль створить у всіх заголовків першого рівня відступ зверху 5 мм. Як значення зовнішніх відступів припустимі від'ємні величини:

```
UL {margin-left:-20px }
```

В цьому випадку Браузер створить "від'ємний" відступ. Такий прийом дозволяє видалити відступи, створювані Браузером за замовчуванням, наприклад, відступи ліворуч у великих цитат і списків.

Зовнішні відступи ми також можемо вказати за допомогою атрибута стилю **margin.** Він задає величини відступу одночасно з усіх боків елемента Web-сторінки:

margin:*<відступ 1>* [*<відступ 2>* [*<відступ 3>* [*<відступ 4>*]]]

Цей атрибут стилю поводиться так само, як **padding**.

Приклад:

```
H1 {margin:5mm 0mm}
```

Однак ми не можемо використовувати атрибути стилю **margin-left, margin-top, margin-right, margin-bottom** і **margin** для завдання зовнішніх відступів у комірок таблиць (тобто відстані між комірками) — вони просто не будуть діяти. Замість цього слід застосувати атрибут стилю **border-spacing**:

border-spacing:<*відступ 1*> [<*відступ 2*>]

Відступи можуть бути задані тільки у вигляді абсолютних значень.

- Якщо вказано одне значення, воно задасть величину відступу з усіх боків комірки таблиці.
- Якщо вказано два значення, перше задасть величину відступу ліворуч і праворуч, а друге — зверху і знизу.

Атрибут стилю застосовується тільки до таблиць (тегу **<TABLE>**):

```
TABLE {border-spacing:1px}
```

Тут ми задали відступи між комірками таблиці, рівні одному пікселу.

Увага. Задаючи відступи, внутрішні або зовнішні, потрібно пам'ятати, що вони збільшують розміри елемента Web-сторінки. Тому, якщо ми застосуємо відступи до блокових контейнерів, що формують дизайн Web-сторінки, то повинні будемо відповідно змінити розміри цих контейнерів, інакше вони змістяться, і дизайн буде порушений.

Також потрібно знати, що при застосуванні відступів до елемента Web-сторінки з розмірами, заданими у вигляді відносних величин, Браузер спочатку обчислює абсолютний розмір елемента, а потім до нього додає величини відступів. Так, якщо ми задамо ширину контейнера в 100%, а потім укажемо для нього відступи, то Браузер спочатку обчислить його абсолютну ширину, ґрунтуючись на розмірах вікна Браузера, а потім додасть до неї величину відступів. В результаті ширина контейнера стане більше, ніж ширина вікна Браузера, і у вікні з'являться смуги прокручування.

Параметри рамки

CSS також дозволяє нам створити суцільну, пунктирну або крапкову рамку по уявлюваній границі елемента Web-сторінки.

Атрибути стилю **border-left-width, border-top-width, border-right-width** і **border-bottom-width** задають товщину, відповідно, лівої, верхньої, правої і нижньої сторін рамки:

border-left-width|border-top-width|border-right-width|
border-bottom-width:thin|medium|thick|<*товщина*>|inherit

Ми можемо вказати або абсолютне, або відносне числове значення товщини рамки, або одне з визначених значень: **thin** (тонка), **medium** (середня) або **thick** (товста). В останньому випадку реальна товщина рамки

залежить від Браузера. Значення товщини за замовчуванням також залежить від Браузера, тому її завжди краще встановлювати явно.

У лістингу 12.2 ми вказали товщину рамки у комірок таблиці, рівну одному пікселу.

Лістинг 12.2

```
TD, TH {border-left-width:thin;
        border-top-width:thin;
        border-right-width:thin;
        border-bottom-width:thin}
```

А от стиль, що створює у всіх заголовків першого рівня рамку з однієї тільки нижньої сторони товщиною 5 пікселів:

```
H1 {border-bottom-width:5px}
```

Фактично всі заголовки першого рівня виявляться підкресленими.

Атрибут стилю **border-width** дозволяє вказати значення товщини відразу для всіх сторін рамки:

border-width:<товщина 1> [<товщина 2> [<товщина 3> [<товщина 4>]]]

- Якщо вказано одне значення, воно задасть товщину всіх сторін рамки.
- Якщо вказано два значення, перше задасть товщину верхньої нижньої, а друге — лівої і правої сторін рамки. і
- Якщо вказано три значення, перше задасть товщину верхньої, друге — лівої і правої, а третє — нижньої сторін рамки.
- Якщо в иимказано чотири значення, перше задасть товщину верхньої, друге — правої, третє — нижньої, а четверте — лівої сторін рамки.

Приклад:

```
TD, TH {border-width: }
```

Атрибути стилю **border-left-color, border-top-color, border-right-color** і **border-bottom-color** задають колір, відповідно, лівої, верхньої, правої і нижньої сторін рамки:

border-left-color|border-top-color|border-right-color|
border-bottom-color:transparent|<*колір*>|inherit

Значення **transparent** задає "прозорий" колір, крізь який буде "просвічувати" фон батьківського елемента.

Увага. Колір рамки завжди слід задавати явно — а якщо ні, то вона може бути не намальована.

Приклад:

```
H1 {border-bottom-width:5px;
    border-bottom-color:red}
```

Атрибут стилю **border-color** дозволяє вказати колір відразу для всіх сторін рамки:

border-color:<*колір 1*> [<*колір 2*> [<*колір 3*> [<*колір 4*>]]]

Він поводиться так само, як і аналогічний атрибут стилю **border-width**:

Приклад:

```
TD, TH {border-width:thin;
        border-color:black}
```

Атрибути стилю **border-left-style, border-top-style, border-right-style** і **border-bottom-style** задають стиль ліній, якими буде намальована, відповідно, ліва, верхня, права і нижня сторона рамки:

border-left-style|border-top-style|border-right-style|
border-bottom-style:none|hidden|dotted|dashed|solid|
double|groove|ridge|inset|outset|inherit

Тут доступні наступні значення:

- **none** і **hidden** — рамка відсутня (звичайна поведінка);
- **dotted** — пунктирна лінія;
- **dashed** — штрихова лінія;
- **solid** — суцільна лінія;
- **double** — подвійна лінія;
- **groove** — "втиснена" тривимірна лінія;
- **ridge** — "опукла" тривимірна лінія;
- **inset** — тривимірна "опуклість";
- **outset** — тривимірне "поглиблення".

Приклад:

```
H1 {border-bottom-width:5px;
    border-bottom-color:red;
    border-bottom-style:double}
```

Атрибут стилю **border-style** дозволяє вказати стиль відразу для всіх сторін рамки:

border-style:<*стиль 1*> [<*стиль 2*> [<*стиль 3*> [<*стиль 4*>]]]

Він поводиться так само, як і аналогічні атрибути стилю **border-width** і **border-color**.

Приклад:

```
TD, TH {border-width:thin;
        border-color:black;
        border-style:dotted}
```

Атрибути **стилю border-left, border-top, border-right** і **border-bottom** дозволяють задати всі параметри для, відповідно, лівої, верхньої, правої й нижньої сторони рамки:

border-left|border-top|border-right|border-bottom:
<*товщина*> <*стиль*> <*колір*>|inherit

У багатьох випадках ці атрибути стилю виявляються переважніше:

```
H1 {border-bottom:5px double red}
```

Атрибут стилю **border** дозволяє задати всі параметри відразу для всіх сторін рамки:

border:<*товщина*> <*стиль*> <*колір*>|inherit

Приклад:

```
TD, TH {border:thin dotted black}
```

Увага. Рамки також збільшують розміри елемента Web-сторінки. Тому, якщо ми створимо рамки в блокових контейнерів, що формують дизайн Web-сторінки, то повинні будемо відповідно змінити розміри цих контейнерів, інакше вони змістяться, і дизайн буде порушений.

Представлення для нашого Web-сайту, частина 6

Насамперед, зробимо відступи між контейнерами, що формують дизайн наших Web-сторінок, і між границями цих контейнерів і їх вмістом.

- Зовнішній відступ між краями вікна Браузера і імістом Web-сторінки дорівнює нулю. Нехай простір у вікні Браузера використовується максимально повно. (За замовчуванням кожний Браузер створює свої відступи між краями свого вікна і вмістом Web-сторінки).
- Внутрішні відступи в контейнері із заголовком Web-сайту (**cheader**) ліворуч і праворуч — по 20 пікселів. Нам доведеться відсунути текст заголовка від країв вікна Браузера, інакше заголовок буде виглядати некрасиво.
- Зовнішній відступ між контейнерами зі смугою навігації (**cnavbar**) і з основним вмістом (**cmain**) — 10 пікселів.
- Внутрішні відступи в контейнера з основним вмістом (**cmain**) з усіх боків — по 5 пікселів.
- Внутрішній відступ у контейнера з основним вмістом (**cmain**) зверху — 10 пікселів. Так ми відокремимо його від контейнерів **cnavbar** і **ccopyright**.
- Внутрішні відступи в контейнері з відомостями про авторські права (**ccopyright**) ліворуч і праворуч — по 20 пікселів. Текст відомостей про авторські права також слід відсунути від країв вікна Браузера.

Тепер розділимо всі чотири контейнери рамками.

- □ Контейнер **cheader** отримає рамку з однієї нижньою стороною.
- □ Контейнер **cmain** — рамку з однієї лівою стороною.
- □ Контейнер **ccopyright** — рамку з однієї верхньою стороною.

Рамки ми зробимо тонкими і точковими. Кольор задамо для них **#BIBEC6** (ясно-синій).

В лістингу 12.3 наведений виправлений фрагмент CSS-коду таблиці стилів **main.css**, що реалізує вибрані нами параметри відступів і рамок.

Лістинг 12.3

```
BODY    {color:#3B4043;
         background-color:#F8F8F8;
         font-family:Verdana, Arial, sans-serif;
         margin:0px}

…… •
```

```
#cheader  {width:1010px;
           padding:0 20px;
           border-bottom:thin dotted #B1BEC6}
#cnavbar  {width:250px;
           height:570px;
           float:left;
           margin-right:10px}
#cmain    {width:760px;
           height:570px;
           float:left;
           overflow:auto;
           padding:5px;
           border-left:thin dotted #B1BEC6}
#ccopyright {width:1010px;
           clear:both;
           padding:10px 20px 0px 20px;
           border-top:thin dotted #BIBEC6}
```

Давайте розберемо його.

Щоб усунути відступ між границями вікна Браузера і вмістом Web-сторінки, ми використали атрибут стилю **margin.** Його ми помістили в стиль перевизначення тегу **<BODY>** і дали йому значення **0** пікселів:

```
BODY    {color:#3B4043;
         background-color:#F8F8F8;
         font-family:Verdana, Arial, sans-serif;
         margin:0px}
```

В іменованому стилі **cheader**, прив'язаному до однойменного контейнера, ми задали внутрішні відступи ліворуч і праворуч, рівні 20 пікселам, і рамку з однієї тільки нижньою стороною. Ця рамка відокремить даний контейнер від розташованих нижче:

```
#cheader  {width:1010px;
           padding:0 20px;
           border-bottom:thin dotted #B1BEC6}
```

Крім того, ми задали як ширину цього контейнера абсолютне значення. Згадаємо: при виведенні на екран контейнера з відносною шириною Браузер спочатку обчислить абсолютне значення його ширини, а потім додасть до нього величину відступів. В результаті чого контейнер стане ширше, чим вікно

Браузера, і у вікні з'являться смуги прокручування, що нам зовсім не потрібно. Тому для ширини контейнера краще задати абсолютне значення, підібравши його так, щоб контейнер гарантовано помістився у вікно Браузера по ширині.

В іменованому стилі **cnavbar** ми вказали зовнішній відступ праворуч 10 пікселів — він візуально відокремить контейнер **cnavbar** від контейнера **cmain**:

```
#cnavbar  {width:250px;
           height:570px;
           float:left;
```

В іменованому стилі **cmain** ми задали внутрішні відступи і рамку з однієї лівою стороною — вона відокремить контейнер **cmain** від контейнера **cnavbar**:

```
#cmain    {width:760px;
           height:570px;
           float:left;
           overflow:auto;
           padding:5px;
           border-left:thin dotted #B1BEC6}
```

В іменованому стилі **ccopyright** ми задаємо відступи ліворуч і праворуч по 20 пікселів, а зверху — 10 пікселів. Крім того, ми створюємо рамку з однієї верхньою стороною, яка відокремить контейнер **ccopyright** від розташованих вище "сусідів":

```
#ccopyright {width:1010px;
           clear:both;
           padding:10px 20px 0px 20px;
           border-top:thin dotted #BIBEC6}
```

От і все. Збережемо таблицю стилів **main.css** і відкриємо Web-сторінку **index.htm** в Браузері.

Подивимося тепер на смугу навігації. Невиразні гіперпосилання скупчилися у верхній частині контейнера **cnavbar**. Давайте їх зрушимо вліво, трохи "розрідимо", створивши відступи, і заодно вкладемо кожне з них в рамки.

Як ми пам'ятаємо, наша смуга навігації являє собою список, а окремі її гіперпосилання — пункти даного списку.

От необхідні зміни:

- Список, що формує смугу гіперпосилань, зрушити вліво на 30 пікселів. Так ми ліквідуємо вільний простір, який утворився після видалення маркерів, ліворуч, що виглядає некрасиво.
- Зовнішні відступи у пунктів списку зверху і знизу — 10 пікселів. Так ми відокремимо гіперпосилання одне від іншого.
- Рамка пунктів списку — тонка, суцільна, колір **#BIBEC6.**
- Внутрішні відступи пунктів списку: зверху і знизу — по 5 пікселів, ліворуч і праворуч — по 10 пікселів. Так ми відокремимо текст пунктів від рамок.

Залишилося тільки відповідно виправити CSS-код таблиці стилів **main.css** (лістинг 12.4).

Лістинг 12.4

```
#navbar   {font-family:Arial, sans-serif;
           font-size:14pt;
           text-transform:uppercase;
           list-style-type:none;
           margin-left:-30px}

.....................................................
#navbar LI {padding:5px 10px;
            margin:10px 0px;
            border:thin solid #B1BEC6}
```

Розглянемо їх докладніше.

В іменований стиль **navbar,** прив'язаний до тегу списку, який формує смугу навігації, ми додали відступ ліворуч, рівний -30 пікселів. Завдяки цьому список зміститься вліво, заповнюючи порожній простір:

```
#navbar   {font-family:Arial, sans-serif;
           font-size:14pt;
           text-transform:uppercase;
           list-style-type:none;
           margin-left:-30px }
```

Створений комбінований стиль створить у пунктів списку, що формує смугу навігації, рамку і задасть відповідні відступи:

```
#navbar LI {padding:5px 10px;
```

```
margin:10px 0px;
border:thin solid #BICBEC6 }
```

Збережемо виправлену таблицю стилів і обновимо відкриту в Браузері Web-сторінку **index.htm,** натиснувши клавішу <F5> (рис. 12.1).

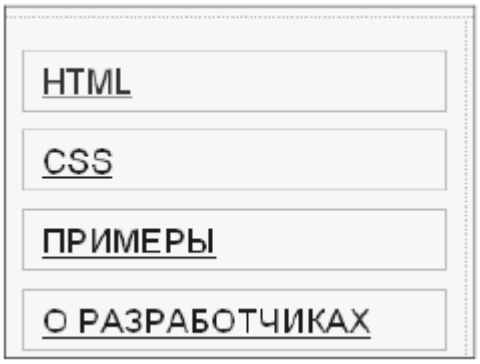


Рис. 12.1. **Смуга навігації на Web-сторінці index.htm після застосування до неї стилів**

Тепер трохи відволічемося від CSS і займемося HTML. Є в наших Web-сторінок вада, непрощенний для гарного Web-дизайнера.

Повна смуга навігації

Правила гарного тону Web-дизайну наказують, щоб у смузі навігації знаходилися гіперпосилання, що вказують на всі Web-сторінки Web-сайту. А в нас частина гіперпосилань знаходиться саме там, а інші скупчені в останньому абзаці основного вмісту.

Створимо нову, повну смугу навігації.

Помістимо гіперпосилання, що вказують на Web-сторінки, які описують різні теги HTML, нижче гіперпосилання, що вказують на Web-сторінку **index.htm** (вона присвячена самому HTML). Зробимо у цих гіперпосилань невеликий відступ зліва — так ми покажемо, що вони відносяться до теми "HTML". Те ж саме проробимо і з гіперпосиланнями, що вказують на Web-сторінки, які описують атрибути стилю CSS і приклади.

Наша смуга навігації — суть список. Її гіперпосилання — пункти цього списку. Тому для гіперпосилань, що вказують на Web-сторінки — опису тегів, логічно створити вкладений список. HTML дозволяє робити вкладені списки.

Відкриємо в Блокноті Web-сторінку **index.htm** і знайдемо HTML-код, що створює смугу навігації (лістинг 12.5).

Лістинг 12.5

```
<UL ID="navbar">
    <LI><A HREF="index.htm">HTML</A></LI>
    <LI><A HREF="css_index.htm">CSS</A></LI>
    <LI><A HREF="samples_index.htm">Приклади</A></LI>
    <LI><A HREF="about.htm">0 розробниках</A></LI>
</UL>
```

Доповнимо його кодом, що створюють вкладений список. Лістинг 12.6 містить виправлений код.

Лістинг 12.6

```
<UL ID="navbar">
    <LI><A HREF="index.htm">HTML</A>
      <UL>
      </UL>
    </LI>
    <LI><A HREF="css_index.htm">CSS</A></LI>
    <LI><A HREF="samples_index.htm">Приклади</A></LI>
    <LI><A HREF="about.htm">0 розробниках</A></LI>
</UL>
```

Ми помістили тег **<UL>**, що створює вкладений список, в розрив між текстом першого пункту "зовнішнього" списку і його закриваючим тегом **</LI>.** Тоді вкладений список буде розташовано нижче першого пункту "зовнішнього" списку.

Після цього знайдемо HTML-код, що створює гіперпосилання на Web-сторінки — описи тегів. Він знаходиться в самому кінці коду Web-сторінки, в окремому абзаці контейнера **cmain**. Запозичимо звідти фрагменти коду, які створюють окремі гіперпосилання, що вказують на Web-сторінки — описи тегів, і створимо на їхній основі пункти вкладеного списку (лістинг 12.7).

Лістинг 12.7

```
<UL ID="navbar">
   <LI><A HREF="index.htm">HTML</A>
     <UL>
        <LI><CODE>!DOCTYPE</CODE></LI>
```

```
          <LI><CODE><A
          HREF="tags/t_audio.htm">AUDIO</A></CODE></LI>
           <LI><CODE>BODY</CODE></LI>

           ……….

           <LI><CODE><A
           HREF="tags/t_video.htm">VIDEO</A></CODE></LI>
       </UL>
    </LI>
    <LI><A HREF="css index.htm">CSS</A></LI>
    <LI><A HREF="s amples_index.htm">Приклади</A></LI>
    <LI><A HREF="about.htm">0 розробниках</A></LI>
</UL>
```

Видалимо з контейнера **cmain** код, який створює абзац із гіперпосиланнями, що вказують на Web-сторінки — описи тегів і заголовок, що відноситься до нього. Потреби в них більше немає.

Збережемо виправлену Web-сторінку **index.htm** і відкриємо її в Браузері. Нижче пункту "HTML" "зовнішнього" списку, що формує смугу навігації, ми побачимо пункти тільки що створеного вкладеного списку.

Вкладений список в смузі навігації виглядає погано. Тому приступимо до створення для нього представлення.

Оскільки ми суттєво збільшили розмір смуги навігації, може наступити такий момент, коли вона стане більше контейнера **cnavbar.** Так що першою справою задамо для цього контейнера підходящу поведінку при переповненні, доповнивши відповідний йому іменований стиль (лістинг 12.8).

Лістинг 12.8

```
#cnavbar    {width:250px;
             height:570px;
             float:left;
             overflow:auto;
             margin-right:10px}
```

Тепер сформулюємо вимоги до вкладеного списку і його пунктів.

- Маркер пунктів вкладеного списку — відсутній.

- Вкладений список, що формує смугу гіперпосилань, зрушити вліво на 30 пікселів. Так ми ліквідуємо вільний простір, що утворився після видалення маркерів.
- Зовнішній відступ у вкладеного списку зверху — 10 пікселів. Так ми відокремимо його від пункту "зовнішнього" списку, в який він вкладений.
- Розмір шрифту у пунктів вкладеного списку — 10 пунктів. Якщо вкладений список є як би "підпорядкованим", те нехай його пункти будуть набрані шрифтом меншого розміру.
- Відступи і рамки у пунктів вкладеного списку відсутні. Так ми зробимо вкладений список компактніше.

Щоб реалізувати ці вимоги, допишемо в кінець таблиці стилів **main.css** CSS-код лістингу 12.9.

Лістинг 12.9

```
#navbar LI UL    {list-style-type: none;
                  margin-left:-20px;
                  margin-top:10px}
#navbar LI UL LI {font-size:12pt;
                  padding:0px;
                  margin:0px;
                  border-style:none}
```

Ми створили два комбіновані стилі. Перший задає параметри вкладеного списку.

Другий комбінований стиль задає параметри пунктів вкладеного списку. Відзначимо, що в ньому ми явно задали величини зовнішніх і внутрішніх відступів рівними нулю, і відсутність рамки. Якщо ми цього не зробимо, до пунктів вкладеного списку будуть застосовані параметри, які ми задали для пунктів "зовнішнього" списку, і у пунктів вкладеного списку також з'являться відступи і рамки. Що нам зовсім не потрібно.

Збережемо таблицю стилів **main.css** і обновимо відкриту в Браузері Web-сторінку **index.htm**, натиснувши клавішу <**F5**>. Смуга навігації повинна виглядати так, як показано на рис. 12.2.

Рис. 12.2. **Повна смуга навігації на Web-сторінці index.htm**

Параметри виділення

Ми знаємо безлічагато способів привернути увагу відвідувача до певних елементів Web-сторінок, використавши теги HTML або атрибути стилю CSS. Але CSS 3 пропонує нам ще один спосіб зробити це — так зване ***виділення***.

- Виділення CSS 3 є рамкою, яка оточується даний елемент Web-сторінки.
- Ми можемо задавати параметри виділення: товщину, колір і стиль.
- Виділення, на відміну від знайомої нам рамки CSS, не збільшує розміри елемента Web-сторінки. Так що можна спокійно застосовувати виділення, не побоюючись, що воно порушить створений нами контейнерний дизайн.

Для завдання параметрів виділення CSS 3 призначено чотири спеціальні атрибути стилю.

Атрибут стилю **`outline-width`** задає товщину рамки виділення:

`outline-width: thin|medium|thick|<товщина>|inherit`

Тут доступні ті ж значення, що і для знайомого нам атрибута стилю **`border- width`**.

Приклад:

```
DFN {outline-width:thin}
```

Тут ми задали для вмісту тегу **<DFN>** тонку рамку виділення.

Атрибут стилю **outline-color** задає колір рамки виділення:

```
outline-color: <колір>|inherit
```

Увага. Колір рамки виділення завжди слід задавати явно — а якщо ні, то вона може бути не намальована.

Приклад:

```
DFN {outline-width:thin;
     outline-color:black}
```

Тепер виділення вмісту тегу **<DFN>** буде виведено чорним кольором.

Атрибут стилю **outline-style** задає стиль ліній, якими буде намальована рамка виділення:

```
outline-style:none|dotted|dashed|solid|double|groove|
ridge|inset|outset|inherit
```

Значення тут доступні ті ж, що і для атрибута стилю **border-style**.

Приклад:

```
DFN {outline-width:thin;
     outline-color:black;
     outline-style:dotted}
```

Атрибут стилю **outline** дозволяє задати відразу всі параметри для рамки виділення:

```
outline:<товщина> <стиль> <колір>|inherit
```

Приклад:

```
DFN {outline:thin dotted black}
```

Давайте додамо в нашу таблицю стилів **main.css** от такий стиль:

```
DFN {outline:thin dotted #B1BEC6}
```

Після цього всі нові терміни (тобто вміст тегів **<DFN>**) на наших Web-сторінках будуть виділені тонкою точковою рамкою ясно-синього кольору.

Контрольні питання

1. Назвіть параметри відступів.
2. Які два види параметрів відступів ви знаєте? Як їх задати?
3. Які атрибути дозволяють задати величини внутрішніх відступів?
4. Які атрибути дозволяють задати величини зовнішніх відступів?
5. Які параметри рамки ви знаєте? Як їх задати?
6. Який атрибут стилю дозволяє задати всі параметри відразу для всіх сторін рамки?
7. Як створити повну смугу навігації?
8. Що таке виділення в CSS 3? Назвіть параметри виділення.
9. Які перametри можна задати? Які атрибути при цьому використовують?
10. Який атрибут дозволяє задати всі параметри виділення?

ТЕМА 13. ПАРАМЕТРИ ТАБЛИЦЬ

***Основна мета*:** навчитися задавати параметри таблиць, використовуючи вивчені раніше атрибути стилю, а також вивчити специфічні саме для таблиць атрибути стилю CSS.

ПАРАМЕТРИ ТАБЛИЦЬ

Розглянемо атрибути стилю CSS для завдання різних параметрів таблиць. Частину з них ми вже вивчили, а деякі вивчимо зараз.

Параметри вирівнювання

Для вирівнювання вмісту комірок таблиці по горизонталі ми застосуємо атрибут стилю **text-align**, описаний в темі 9:

```
TD, TH {text-align:center}
```

Цей же атрибут стилю використовується для вирівнювання тексту в заголовку таблиці (тегу **<CAPTION>**):

```
CAPTION {text-align:left}
```

Вміст комірок таблиць по вертикалі ми вирівняємо за допомогою атрибута стилю **vertical-align**:

```
vertical-align:baseline|sub|super|top|text-top|middle|
 bottom|text-bottom|<проміжок між базовими лініями>
 |inherit
```

Стосовно до інших елементів Web-сторінок він був описано в темі 9, але у випадку комірок таблиць поводиться трохи по-іншому.

- **top** — вирівнює вміст комірки по її верхньому краю (звичайна поведінка).
- **middle** — вирівнює вміст комірки по її центру.
- **bottom** — вирівнює вміст комірки по її нижньому краю.

Інші значення цього атрибута стилю діють так само, як і для інших елементів Web-сторінок:

```
TD, TH {vertical-align:middle}
```

Параметри відступів і рамок

Для завдання відступів ми можемо користуватися атрибутами стилю, знайомими нам по темі 12.

- Для завдання внутрішніх відступів між вмістом комірки і її границею — атрибутами стилю **padding-left, padding-top, padding-right, padding-bottom** і **padding**.
- Для завдання зовнішніх відступів між границями сусідніх комірок — атрибутом стилю **border-spacing**.

Параметри рамок задамо через відповідні атрибути стилю, які також знайомі нам по темі 12 (лістинг 13.1).

Лістинг 13.1

```
TABLE  {align:center;
        border:medium solid black;
        border-spacing:1px}
TD, TH {border:thin dotted black;
        padding:2px}
```

Тут ми призначили для самої таблиці середньої товщини суцільну чорну рамку та відступ між комірками, рівний одному пікселу, а для комірок цієї таблиці — тонку точкову чорну рамку і відступ між границею комірки та її вмістом, рівний двом пікселам.

Якщо ми задамо рамки навколо комірок таблиці, Браузер намалює рамку навколо кожної комірки. Така таблиця буде виглядати як набір прямокутників - комірок, що містяться у великому прямокутнику-таблиці (рис. 13.1).

Версия HTML	Год выхода	Особенности
1.0	1992	Официально не была стандартизована
2.0	1995	Первая стандартизованная версия
3.2	1997	Поддержка таблиц и графики
4.0	1997	"Очищен" от устаревших тегов
4.01	1999	В основном, исправление ошибок
5.0	?	В разработке

Рис. 13.1. **Звичайна поведінка Браузера — рамки рисуються навколо кожної комірки таблиці**

Однак у друкованих виданнях набагато частіше зустрічаються таблиці іншого вигляду. У них рамки присутні тільки між комірками (рис. 13.2).

Версия HTML	Год выхода	Особенности
1.0	1992	Официально не была стандартизована
2.0	1995	Первая стандартизованная версия
3.2	1997	Поддержка таблиц и графики
4.0	1997	"Очищен" от устаревших тегов
4.01	1999	В основном, исправление ошибок
5.0	?	В разработке

Рис. 13.2. **Таблиця, у якій рисуються тільки рамки, поділяючі комірки**

Атрибут стилю **border-collapse** вказує Браузеру, як будуть рисуватися рамки комірок в таблиці:

border-collapse:collapse|separate|inherit

- **separate** — кожна комірка таблиці міститься в окрему рамку (див. рис. 13.1). Це звичайна поведінка.
- **collapse** — рисуються рамки, що розділяють комірки таблиці (див. рис. 13.2).
- **inherit** — рамок немає.

Даний атрибут стилю застосовується тільки до самих таблиць (тегів **<TABLE>**).

Приклад:

```
TABLE {border-collapse:collapse}
```

Параметри розмірів

Для завдання розмірів — ширини та висоти — таблиць і їх комірок підійдуть атрибути стилю **width** і **height**, описані в темі 12.

- Якщо необхідно задати ширину або висоту всієї таблиці, потрібний атрибут стилю вказують саме для неї:

```
TABLE {width:100%;
       height:300px}
```

- Якщо потрібно задати ширину колонки, атрибут стилю **width** вказують для першої комірки, що входить в цю колонку (лістинг 13.2).

Лістинг13.2

```
<TABLE> <TR>
<TH>Перша колонка</TH>
<TH STYLE="width:40px">Друга колонка шириною в 40 пікселів</TH>
<TH>Третя колонка</TH>
</TR>

.....
</TABLE>
```

- Якщо потрібно задати висоту рядка, атрибут стилю **height** вказують для першої комірки цього рядка (лістинг 13.3).

Лістинг 13.3

```
<TABLE>

.....
    <TR>
       <TD STYLE="height:30px">Рядок висотою в 30 пікселів</TD>

       .....
    </TR>
```

```
.....
</TABLE>
```

Звичайно всі розміри, які ми задамо для таблиці і її комірок, — не більш ніж рекомендація для Браузера. Якщо вміст таблиці не буде в ній поміщатися, Браузер збільшить ширину або висоту таблиці. Найчастіше це може бути неприйнятно, тому стандарт CSS передбачає засоби, що дозволяють змінити таку поведінку Браузера.

Атрибут стилю **table-layout** дозволяє вказати, як Браузер буде трактувати розміри, задані нами для таблиці і її комірок:

```
table-layout : auto|fixed|inherit
```

- **auto** — Браузер може змінити розміри таблиці і її комірок, якщо вміст в них не поміщається. Це звичайна поведінка.
- **fixed** — розміри таблиці і її комірок в жодному разі змінюватися не будуть. Якщо вміст у них не поміщається, виникне переповнення, параметри якого ми можемо задавати за допомогою атрибутів стилю **overflow, overflow-x** і **overflow-y** (див. тему 11).
  Даний атрибут стилю застосовується до самої таблиці (тега **<TABLE>**).

  Приклад:

```
TABLE {table-layout:fixed;
overflow:auto}
```

Інші параметри

І ще кілька корисних атрибутів стилю.

Атрибут стилю **caption-side** вказує місце розташування заголовка таблиці щодо самої таблиці:

```
caption-side:top|bottom|inherit
```

- **top** — заголовок розташовується над таблицею (звичайна поведінка).
- **bottom** — заголовок розташовується під таблицею.
  Даний атрибут стилю застосовується до самої таблиці (тега **<TABLE>**).

  Приклад:

```
TABLE {caption-side:bottom}
```

Атрибут стилю **empty-cells** вказує, як Браузер повинен виводити на екран порожні (що не мають вмісту) комірки:

```
empty-cells:show|hide|inherit
```

- **show** — порожні комірки будуть виводитися на екран. Якщо для них було задано інший фон, на екран буде виведений фон, а якщо задані рамки, будуть виведені рамки.
- **hide** — порожні комірки не будуть виводитися на екран.

Звичайна поведінка залежить від Браузера, так що, якщо це критично, краще явно задати потрібне значення атрибута стилю **empty-cells**.

Атрибут стилю **empty-cells** також застосовується до самої таблиці (тега **<TABLE>**).

Приклад:

```
TABLE {empty-cells:hide}
```

Представлення для нашого Web-сайту, частина 7

От, власне, і всі атрибути стилю, що задають параметри таблиць. Для практики давайте попрацюємо над нашою єдиною таблицею — списком версій HTML. Зробимо її більш красивою.

Спочатку, як звичайно, сформулюємо перелік її параметрів.

- Зовнішні відступи зверху і знизу таблиці — 10 пікселів. Нехай таблиця буде візуально відділена від "сусідів".
- Рамка навколо самої таблиці — тонка, суцільна, колір **#B1BEC6**.
- Будуть виводитися тільки рамки, що розділяють комірки. Таблиці з такими рамками більш звичні.
- Внутрішні відступи в комірках — 2 піксела.
- Рамки комірок — тонкі, точкові, колір **#B1BEC6**.
- Вирівнювання тексту заголовка таблиці — по лівому краю.

Лістинг 13.4 містить виправлений фрагмент таблиці стилів **main.css**.

Лістинг 13.4

```
TABLE   {font-size:10pt;
         margin:10px 0px;
         border:thin solid #B1BEC6;
         border-collapse:collapse}
```

```
.....................
TD, TH  {padding: 2px;
         border:thin dotted #B1BEC6}
CAPTION {text-align:left}
```

Тут ми доповнили стиль перевизначення тегу **<TABLE>** і створили стилі перевизначення тегів **<TD>, <TH>** і **<CAPTION>**.

Збережемо таблицю стилів **main.css** і відкриємо Web-сторінку **index.htm** у Браузері. Рамки і відступи поліпшили її.

КОНТРОЛЬНІ ПИТАННЯ

1. Параметри таблиць: параметри вирівнювання.
2. Назвіть атрибути для вирівнювання вмісту комірок таблиці по двом напрямкам.
3. Параметри таблиць: параметри відступів.
4. Параметри таблиць: параметри рамок.
5. Які різновиди рамок навколо комірок таблиці існують? Як вони задаються?
6. Параметри таблиць: параметри розмірів.
7. Які атрибути стилю використовуються для завдання розмірів — ширини і висоти — таблиць і їх комірок?
8. Який атрибут стилю дозволяє вказати, чи буде Браузер при необхідності міняти розміри, задані нами для таблиці і її комірок?
9. Параметри таблиць: параметр, що вказує місце розташування заголовка таблиці щодо самої таблиці.
10. Параметри таблиць: параметр, що вказує як виводити на екран порожні комірки.

ТЕМА 14. СПЕЦІАЛЬНІ СЕЛЕКТОРИ

***Основна мета*:** вивчити спеціальні селектори и навчитися їх застосовувати при оформленні сайтів.

СПЕЦІАЛЬНІ СЕЛЕКТОРИ

Ми вивчили атрибути стилів CSS. Їх дуже багато; одні задають параметри шрифту, інші — параметри фона, треті — параметри відступів та ін. Можна сказати, що немає такого параметра, що впливає на представлення елементів Web-сторінки, який ми не могли б задати, користуючись засобами CSS.

Розглянемо селектори стилів. Із селекторами ми познайомилися ще в темі 8: стилі перевизначення тегів, стильові класи, іменовані та комбіновані стилі.

Розглянемо спеціальні селектори.

Спеціальний селектор — це селектор, що неявно прив'язує стиль до елемента Web-сторінки відповідно до більш складного, чим ім'я тегу, критерія. Таким критерієм може бути, наприклад, порядковий номер елемента в батьківському елементі, вказівка на певну частину вмісту абзацу (перший рядок або першу букву), стан гіперпосилання (чи відвідано воно, чи ні) та ін. Звичайно спеціальні селектори використовують у комбінованих стилях, щоб зробити їх більш конкретними.

Існує кілька різновидів спеціальних селекторів.

Комбінатори

Комбінатори — різновид спеціальних селекторів, що прив'язує стиль до елемента Web-сторінки на підставі його місця розташування щодо інших елементів.

Комбінатор **+** дозволяє прив'язати стиль до елемента Web-сторінки, що безпосередньо знаходиться за іншим елементом. При цьому обидва дочірніх елемента повинні бути вкладеними в той самий батьківський:

<елемент 1>+<елемент 2> {<стиль>}

Стиль буде прив'язано до **елемента 2**, який повинен безпосередньо іти за **елементом 1**.

Лістинг 14.1

```
H6+PRE {font-size:smaller}

.....................
<H6>Це заголовок</H6>
<PRE>Цей текст буде набраний зменшеним шрифтом</PRE>
<P>А цей — звичайним шрифтом</P>
<H6>Це заголовок</H6>
<P>И цей — звичайним шрифтом</P>
<PRE>И цей — звичайним шрифтом</PRE>
```

Стиль, описаний у лістингу 14.1, буде застосований тільки до першого тексту фіксованого форматування, тому що він безпосередньо іде за заголовком шостого рівня.

Комбінатор **~** (тильда) дозволяє прив'язати стиль до елемента Web-сторінки, що розташований за іншим елементом і, можливо, відділений від нього іншими елементами. При цьому обидва дочірніх елемента повинні бути вкладеними в той самий батьківський:

<елемент 1>~<елемент 2> {<стиль>}

Стиль буде прив'язано до **елемента 2**, який повинен іти за **елементом 1**. При цьому **елемент 2** може бути відділено від **елемента 1** іншими елементами.

Лістинг 14.2

```
H6~PRE {font-size:smaller}
.....
<H6>Це заголовок</H6>
<PRE>Цей текст буде набраний зменшеним шрифтом</PRE>
<P>А цей — звичайним шрифтом</P>
<H6>Це заголовок</H6>
<P>И цей — звичайним шрифтом</P>
<PRE>А цей — зменшеним шрифтом</PRE>
```

Стиль із лістингу 14.2 буде застосований до обох текстів фіксованого формату: і до того, що безпосередньо іде за заголовком шостого рівня, і до того, який відділений від заголовка абзацом.

Комбінатор **>** дозволяє прив'язати стиль до елемента Web-сторінки, безпосередньо вкладеного в інший елемент:

<елемент 1>><елемент 2> {<стиль>}

Стиль буде прив'язано до **елемента 2**, який безпосередньо вкладено в **елемент 1**.

Лістинг 14.3

```
BLOCKQUOTE>P {font-style:italic}
.....
<BLOCKQUOTE>
   <P>Цей абзац буде набраний курсивом</P>
   <DIV>
      <P>А цей абзац — звичайним шрифтом</P>
   </DIV>
</BLOCKQUOTE>
```

Стиль із лістингу 14.3 буде застосований тільки до абзацу, безпосередньо вкладеного в велику цитату. На другий абзац, послідовно вкладений у велику цитату і блоковий контейнер, цей стиль діяти не буде.

Комбінатор <**пробіл**> дозволяє прив'язати стиль до елемента Web-сторінки, вкладеного в інший елемент, причому не обов'язково безпосередньо:

```
<елемент 1> <елемент 2> {<стиль>}
```

Стиль буде прив'язано до **елемента 2**, який так чи інакше вкладено в **елемент 1**. При цьому **елемент 2** може бути вкладений в інший елемент, вкладений в **елемент 1**, або навіть у кілька таких елементів послідовно.

Лістинг 14.4

```
BLOCKQUOTE>P {font-style:italic}
.....
<BLOCKQUOTE>
   <P>Цей абзац буде набраний курсивом</P>
   <DIV>
      <P> Цей абзац — теж.</P>
   </DIV>
</BLOCKQUOTE>
```

Стиль із лістингу 14.4 буде застосований до обох абзаців: і до вкладеного безпосередньо в велику цитату, і до того, що послідовно вкладений у велику цитату та блоковий контейнер.

Стиль, приведений у лістингу 14.4, ми вивчали як комбінований стиль ще в темі 8.

Селектори по атрибутах тегу

Селектори по атрибутах тегу — це спеціальні селектори, що прив'язують стиль до тегу на підставі, чи є присутнім в ньому певний атрибут або чи має він певне значення.

Селектори по атрибутах тегу використовують не самі по собі, а тільки в сукупності зі стилями перевизначення тегу або комбінованими стилями. Їх записують відразу після основного селектора без усяких пробілів і беруть у квадратні дужки.

Селектор виду

```
<основний селектор>[<ім'я атрибута тегу>] {<стиль>}
```

прив'язує **стиль** до елементів, теги яких мають атрибут із вказаним **ім'ям**.

Приклад:

```
TD[ROWSPAN] {background-color:grey}
```

Цей стиль буде прив'язаний до комірок таблиці, теги яких мають атрибут **ROWSPAN**, тобто до комірок, об'єднаних по горизонталі.

Селектор виду

<основний селектор>[<ім'я атрибута тегу>=<значення>] {<стиль>}

прив'язує **стиль** до елементів, теги яких мають атрибут із вказаними **ім'ям** і **значенням**.

Приклад:

```
TD[ROWSPAN=2] {background-color:grey}
```

Даний стиль буде прив'язаний до комірок таблиці, теги яких мають атрибут **ROWSPAN** зі значенням **2**, тобто до подвійних комірок, об'єднаних по горизонталі.

Селектори виду

<основний селектор>[<ім'я атрибута тегу>~=
<список значень, розділених пробілами>] {<стиль>}

и

<основний селектор>[<ім'я атрибута тегу>|=
<список значень, розділених комами>] {<стиль>}

прив'язує **стиль** до елементів, теги яких мають атрибут із вказаним **ім'ям** і **значенням**, що збігаються з одним із вказаних у **списку**.

Приклад:

```
TD[ROWSPAN~=2 3] {background-color:grey}
TD[ROWSPAN|=2,3] {border:thin dotted black}
```

Ці стилі будуть прив'язані до комірок таблиці, теги яких мають атрибут **ROWSPAN** зі значеннями **2** і **3**, тобто до подвійних і потрійних комірок, об'єднаних по горизонталі.

Селектор виду

<основний селектор>[<ім'я атрибута тегу>^=<підрядок>]{<стиль>

прив'язує **стиль** до елементів, теги яких мають атрибут із вказаним **ім'ям** і **значенням**, що починаються із вказаного **підрядка.**

Приклад:

IMG[SRC^=http://www.pictures.ru]{margin:5px}

Цей стиль буде прив'язаний до графічних зображень, теги яких мають атрибут **SRC** зі значенням, що починаються з підрядка "http://www.pictures.ua", тобто. до зображень, взятих з Web-сайту http://www.pictures.ua.

Селектор виду

<основний селектор>[<ім'я атрибута тегу>$=<підрядок>]{<стиль>}

прив'язує **стиль** до елементів, теги яких мають атрибут із вказаним **ім'ям** і **значенням**, що закінчуються вказаним **підрядком.**

Приклад:

IMG[SRC$=gif] {margin:10px}

Даний стиль буде прив'язаний до графічних зображень, теги яких мають атрибут **SRC** зі значенням, що закінчуються підрядком **"gif"**, тобто до зображень формату GIF.

Селектор виду

<основний селектор>[<ім'я атрибута тегу>*=<підрядок>]{<стиль>}

прив'язує **стиль** до елементів, теги яких мають атрибут із вказаним **ім'ям** і **значенням**, що включають вказаний **підрядок**.

Приклад:

IMG[SRC*=/picts/] {margin:10px}

Цей стиль буде прив'язаний до графічних зображень, теги яких мають атрибут **SRC** зі значенням, що включають підрядок **"/picts/",** тобто до зображень, взятих з папки **picts** Web-сайту, звідки вони завантажені.

Псевдоелементи

Псевдоелементи — різновид спеціальних селекторів, що прив'язують стиль до певного фрагмента елемента Web-сторінки. Таким фрагментом може бути перший символ або перший рядок в абзаці.

Псевдоэлементы використовуються не самі по собі, а тільки в сукупності з іншими стилями. Їх записують відразу після основного селектора без усяких пробілів:

```
<основний селектор><псевдоелемент> {<стиль>}
```

Псевдоелемент **::first-letter** прив'язує стиль до першої букви тексту в елементі Web-сторінки, якщо їй не передує вбудований елемент, що не є текстом, наприклад, зображення.

Приклад:

```
P::first-letter{font-size:larger}
```

Цей стиль буде прив'язаний до першої букви абзацу.

Псевдоелемент **::first-line** прив'язує стиль до першого рядка тексту в елементі Web-сторінки:

```
P::first-line {text-transform:uppercase}
```

Даний стиль буде прив'язаний до першого рядка абзацу.

Псевдокласи

Псевдокласи — самий потужний різновид спеціальних селекторів. Вони дозволяють прив'язати стиль до елементів Web-сторінки на основі їх стану (чи наведений на них курсор миші, чи ні) і місця розташування в батьківському елементі.

Псевдокласи також використовують не самі по собі, а тільки в сукупності з іншими стилями. Їх записують відразу після основного селектора без всяких пробілів:

```
<основний селектор><псевдоклас>{<стиль>}
```

Псевдокласи, у свою чергу, самі діляться на групи.

Псевдокласи гіперпосилань

Псевдокласи гіперпосилань служать для прив'язки стилів до гіперпосилань на основі їх стану: є гіперпосилання відвіданим або невідвіданим, клацає на ньому відвідувач мишею або тільки навів на нього курсор миші та ін.

Псевдокласи гіперпосилань, доступні в стандарті CSS 3:

- **`:link`** — невідвідане гіперпосилання;
- **`:visited`** — відвідане гіперпосилання;
- **`:active`** — гіперпосилання, на якому відвідувач в цей момент клацає мишею;
- **`:focus`** — гіперпосилання, що має фокус введення;
- **`:hover`** — гіперпосилання, на яке наведений курсор миші.

Псевдокласи гіперпосилань застосовуються спільно зі стилями, що задають параметри для гіперпосилань. Це можуть бути стилі перевизначення тегу **<A>** або комбіновані стилі, створені на основі стилю перевизначення тегу **<A>** (лістинг 14.5).

Лістинг 14.5

```
A:link    {text-decoration:none}
A:visited {text-decoration:overline}
A:active  {text-decoration:underline}
A:focus   {text-decoration:underline}
A:hover   {text-decoration:underline}
```

В лістингу 14.5 псевдокласи гіперпосилань діють спільно зі стилями перевизначення тегу **<A>.** В результаті приведені стилі будуть застосовані до всіх гіперпосилань на Web-сторінці.

Лістинг 14.6

```
A.special:link    {color:darkred}
A.special:visited {color:darkviolet}
A.special:active  {color:red}
A.special:focus   {color:red}
A.special:hover   {color:red}
```

В лістингу 14.6 псевдокласи гіперпосилань сполучені з комбінованими стилями, що поєднують стиль перевизначення тегу **<A>** і стильовий клас **special.** Вони будуть застосовані тільки до тих гіперпосилань, до яких був прив'язаний вказаний стильовий клас.

Псевдокласи гіперпосилань можна комбінувати, записуючи їх один за одним:

```
A:visited:hover {text-decoration:underline}
```

Цей стиль буде застосований до відвіданого гіперпосилання, над яким знаходиться курсор миші.

Псевдоклассы гіперпосилань — єдиний засіб, що дозволяє вказати параметри для тексту гіперпосилань.

Структурні псевдокласи

Структурні псевдокласи дозволяють прив'язати стиль до елемента Web-сторінки на основі його місця розташування в батьківському елементі.

Псевдоклассы **:first-child** і **:last-child** прив'язують стиль до елемента Web-сторінки, який є, відповідно, першим і останнім дочірнім елементом свого батька:

```
UL:first-child {font-weight:bold}
```

Цей стиль буде застосований до пункту, що є першим дочірнім елементом свого батька-списку, тобто до першого пункту списку.

Приклад:

```
TD:last-child {color:green}
```

Даний стиль буде застосований до останнього дочірнього елемента кожного рядка таблиці — до її останньої комірки.

Лістинг 14.7 ілюструє більш цікавий випадок.

Лістинг 14.7

```
#cmain P:first-child {font-weight:bold}

.....
<DIV ID="cmain">
   <P>Цей абзац буде набраний зеленим кольором</P>
   <BLOCKQUOTE>
      <P>Цей абзац — теж.</P>
   </BLOCKQUOTE>
   <BLOCKQUOTE>
      <P>И цей — теж.</P>
      <P>А цей — ні.</P>
```

```
</BLOCKQUOTE> </DIV>
```

Стиль, приведений в лістингу 14.7, буде застосований до найпершого абзацу, єдиного абзацу в першій великій цитаті і першого абзацу в другій великій цитаті. Справа в тому, що останні два абзаци, до яких буде застосований стиль, відлічуються щодо своїх батьків — великих цитат — і в них є першими.

А якщо ми змінимо даний стиль от так:

```
#cmain>P:first-child {font-weight:bold}
```

він буде застосований тільки до найпершого абзацу, безпосередньо вкладеного в контейнер **cmain**. Адже ми вказали комбінатор **>**, який наказує, що елемент, до якого застосовується стиль, повинен бути безпосередньо вкладений в свого батька.

Псевдоклас **:only-of-type** прив'язує стиль до елемента Web-сторінки, який є єдиним дочірнім елементом, сформованим за допомогою даного тегу, в своєму батьку.

Лістинг 14.8 ілюструє приклад.

Лістинг 14.8

```
P:only-of-type {color: green}

.....
<BLOCKQUOTE>
   <P>Цей абзац буде набраний зеленим кольором</P>
</BLOCKQUOTE>
<BLOCKQUOTE>
   <P>И цей абзац буде набраний зеленим кольором</P>
   <ADDRESS>А цей текст — ні.</ADDRESS>
</BLOCKQUOTE>
<BLOCKQUOTE>
   <P>И цей — ні.</P>
   <P>И цей — ні.</P>
</BLOCKQUOTE>
```

Стиль із лістингу 14.8 буде застосований до абзаців, вкладених в першу і другу великі цитати, тому що ці абзаци є там єдиними елементами, сформованими за допомогою тегу **<P>.** До абзаців, вкладених в третю велику цитату, стиль застосований не буде.

Псевдоклас **:nth-child** дозволяє прив'язати стиль до елементів Web-сторінки, вибравши їх по порядкових номерах, під якими ці елементи визначені в своєму батьку:

<основний селектор>*:nth-child(<a>n+<b>) *{<стиль>}

Після імені даного псевдокласа в дужках указують формулу, по якій обчислюються номери елементів, до яких буде застосований стиль. Ця формула має два параметри, що задаються Web-дизайнером: **a** і **b**. Їхні значення повинні бути невід'ємними цілими числами.

Розглянемо, як виконується прив'язка стилю, що включає псевдоклас **:nth-child**.

Спочатку Браузер зчитує CSS-код стилю і, керуючись його селектором, знаходить елементи Web-сторінки, до яких, можливо, буде застосований даний стиль. Також Браузер визначає батька цих елементів.

Після цього Браузер розбиває всі знайдені елементи на групи. Кількість елементів в кожній групі задається параметром **a** приведеної формули. Після розбивки Браузер обчислює кількість груп, що вийшли.

Далі Браузер послідовно підставляє в формулу замість **n** номера груп, що вийшли, починаючи з нуля. В результаті кожного проходу обчислення виходить номер елемента, до якого застосовується стиль.

Для **прикладу** створимо таблицю з п'яти рядків і застосуємо до неї такий стиль:

```
TR:nth-child(2n+1) {text-align:center}
```

В стилі ми вказали, що група повинна містити два дочірні елементи. Тоді Браузер розіб'є рядки таблиці на дві групи, по два рядки в кожній.

- Браузер підставить у формулу **n** рівне **0**. Після обчислення вийде значення **1**. Браузер застосує даний стиль до першого рядка таблиці.
- Браузер підставить у формулу **n** рівне **1**. Після обчислення вийде значення **3**. Браузер застосує даний стиль до третього рядка таблиці.
- Браузер підставить у формулу **n** рівне **2**. Після обчислення вийде значення **5**. Браузер застосує даний стиль до п'ятого рядка таблиці. Оскільки 2 — загальна кількість груп, то на цьому обчислення закінчаться.

В результаті даний стиль буде застосований до кожного непарного рядка нашої таблиці.

Якщо ми створимо такий стиль:

```
TR:nth-child(2n+0) {text-align:center}
```

те він буде застосований до другого і четвертого (парних) рядків нашої таблиці. Ми можемо зробити запис трохи коротшим:

```
TR:nth-child(2n) {text-align:center}
```

Раніше ми розглядали приклади з розбивкою дочірніх елементів на групи. Але ми можемо відмінити розбивку, вказавши нульову кількість елементів в групі.В цьому випадку Браузер знайде **b**-й дочірній елемент і застосує стиль до нього.

Так, якщо ми створимо такий стиль:

```
TR:nth-child(0n+2) {text-align:center}
```

то Браузер застосує його до другого рядка таблиці.

Ми можемо записати даний стиль і так:

```
TR:nth-child(2) {text-align:center}
```

і Браузер правильно його обробить.

Замість вказівки формули в дужках ми можемо поставити там визначені значення **odd** і **even**. Перше прив'язує стиль до непарних дочірніх елементів, друге — до парних:

```
TR:nth-child(odd) {background-color:grey}
TR:nth-child(even) {background-color yellow}
```

Перший стиль буде застосований до непарних рядків таблиці, другий — до парних.

Псевдоклас **:nth-last-child** аналогічний розглянутому нами псевдокласу **:nth-child** за тим винятком, що відлік дочірніх елементів ведеться не з початку, а з кінця батьківського елемента.

Приклад:

```
TR:nth-last-child(1) {text-align:center}
```

Даний стиль буде застосований до останнього рядка таблиці.

Приклад:

```
#cmain P:nth-last-child(2) {font-weight:bold}
```

А цей стиль буде застосований до передостаннього (другому з кінця) абзацу в контейнері **cmain**.

Ще стандарт CSS описує псевдокласи **:nth-of-type** і **:nth-last-of-type**. Вони працюють так само, як і псевдокласи **:nth-child** і **:nth-last-child**.

Структурний псевдоклас **:empty** прив'язує стиль до елементів, що не мають дочірніх елементів.

Приклад:

```
P:empty {display:none}
```

Цей стиль буде прив'язаний до порожніх (тих, що не мають вмісту) абзаців.

Псевдокласи :not і *

Особливий псевдоклас **:not** дозволяє прив'язати стиль до будь-якого елемента Web-сторінки, що не задовольняє заданим умовам. Такою умовою може бути будь-який селектор:

```
<основний селектор>:not(<селектор вибору>) {<стиль>}
```

Стиль буде прив'язаний до елемента Web-сторінки, що задовольняє основному селектору, і не задовольняє селектору вибору.

Приклад:

```
DIV:not(#cmain) {background-color:yellow}
```

Тут ми вказали як основний селектор стиль перевизначення тегу **<DIV>**, а як селектор вибору — іменований стиль **cmain**. В результаті даний стиль буде застосований до всіх блокових контейнерів, за винятком **cmain**.

А от стиль, який буде застосований до всіх рядків таблиці, за винятком першого:

```
TR:not(:nth-child(1)) {background-color:grey}
```

Псевдоклас ***** ("зірочка") позначає будь-який елемент Web-сторінки.

Приклад:

```
#cmain>*:first-child {border-bottom: medium solid black}
```

Цей стиль буде застосований до першого елемента будь-якого типу, який безпосередньо вкладений в контейнер **cmain.**

Представлення для нашого Web-сайту, частина 8

От список параметрів Web-сторінок нашого Web-сайту, які ми задамо за допомогою спеціальних селекторів.

- Текст невідвіданих гіперпосилань не підкреслено, колір **#576C8C** (темно-синій).
- Текст невідвіданих і відвіданих гіперпосилань, розташованих в смузі навігації, — не підкреслено, колір **#576C8C**. Ми задали для невідвіданих і відвіданих гіперпосилань в смузі навігації однакові параметри — так прийнято.
- Текст відвіданих гіперпосилань, не розташованих в смузі навігації, — не підкреслено, колір **#A1AFBA** (синій).
- Текст активного гіперпосилання — підкреслено, колір **#576C8C.**
- Текст гіперпосилання, на яке наведений курсор миші, — підкреслено, колір **#576C8C.**
- Текст гіперпосилання, що має фокус введення, підкреслено, колір **#576C8C.**
- Шрифт першої букви першого абзацу в контейнері **amain** — 18 пунктів і напівжирний.
- Вирівнювання тексту в комірках першої та другої колонок таблиці — списку версій HTML — по центру. Ці комірки містять винятково числа, і центральне вирівнювання для них підходить більше.

Щоб втілити задумане, нам буде потрібно додати в таблицю стилів **main.css** кілька нових стилів (лістинг 14.9).

Лістинг 14.9

```
A:link                {color: #576C8C;
                       text-decoration:none}
A:visited             {color: #A1AFBA;
                       text-decoration:none}
A:focus, A:hover, A:active {color: #576C8C;
                       text-decoration:underline}
```

Стилі з лістингу 14.9 задають параметри гіперпосилань, розташованих поза смугою навігації. Тут ми активно використовуємо псевдокласи гіперпосилань.

Лістинг 14.10

```
#navbar A:link,
   #navbar A:visited    {color:#576C8C;
                         text-decoration:none}
```

```
#navbar A:focus,
   #navbar A:hover,
   #navbar A:active    {color:#576C8C;
                        text-decoration:underline}
```

Стилі з лістингу 14.10 задають параметри гіперпосилань смуги навігації. Відзначимо, що тут застосовуються комбіновані стилі, що містять вказівку на список **navbar**, який формує смугу навігації, — так простіше і наочніше.

От стиль, що задає параметри першої букви першого абзацу в контейнері **cmain** (тобто в основному вмісті Web-сторінки):

```
#cmain>P:first-child:first-letter {font-size:18pt;
                                  font-wight:bold}
```

Він є комбінованим стилем, що містить цілих три спеціальні селектори, і досить складний. Тому розглянемо його детальніше.

- **#cmain>P** — абзац повинен бути безпосередньо вкладений в контейнер **cmain**.
- **#cmain>P:first-child** — крім того, абзац повинен бути першим дочірнім елементом свого батька (даного контейнера).
- **#cmain>P:first-child:first-letter** — вказуємо на першу букву даного абзацу. Саме до неї буде прив'язаний цей стиль.

Зверніть увагу: — ми спеціально вказали, що абзац повинен бути безпосередньо вкладений у контейнер **cmain**. Якщо ми цього не зробимо, написавши так:

```
#cmain P:first-child:first-letter {font-size:18pt;
                                  font-weight:bold}
```

стиль буде застосований і до абзацу, який вкладений в тег великої цитати — адже цей абзац також буде першим дочірнім елементом свого батька. А нам це не потрібно.

От два однакові стилі, що задають вирівнювання тексту по центру для першої і другої комірок кожного рядка таблиці:

```
.table-html-versions TD:first-child,
.table-html-versions TD:nth-child(2) {text-align:center}
```

Як бачимо, вони являють собою комбінований стиль, що включає стильовий клас **table-html-versions**.

Щоб дані стилі почали діяти на таблицю, нам доведеться прив'язати цей стильовий клас до її тегу **<TABLE>:**

```
<TABLE CLASS="table-html-versions">
   <CAPTION>Список версій HTML:</CAPTION>
</TABLE>
```

Додамо приведені стилі в таблицю стилів **main.css**, внесемо виправлення в Web-сторінку **index.htm**, збережемо їх і відкриємо Web-сторінку **index.htm** в Браузері.

КОНТРОЛЬНІ ПИТАННЯ

11. Що таке спеціальні селектори?
12. Які різновидів спеціальних селекторів існують?
13. Що таке комбінатори? Які види комбінаторів існують?
14. Що таке селектори по атрибутах тегу? Які види селекторів по атрибутах тегу існують?
15. Що таке псевдоелементи? Які види псевдоелементів існують?
16. Що таке псевдокласи? Які види псевдокласів існують?
17. Розповідайте про псевдокласи гіперпосилань.
18. Що таке структурні псевдокласи?
19. Псевдоклас :**not.**
20. Псевдоклас *****

ТЕМА 15. РОЗШИРЕННЯ CSS

***Основна мета*:** вивчити розширення CSS для певних Браузерів і навчитися їх застосовувати.

РОЗШИРЕННЯ CSS

Розширення CSS — це атрибути стилю, які підтримуються певними Браузерами і, можна сказати, розширюють можливості поточної версії CSS. Ці атрибути стилю можуть бути включені в чорнову редакцію майбутньої версії CSS, а можуть взагалі бути відсутніми в будь-яких документах, що описують стандарт. В останньому випадку розширення CSS служать для підтримки специфічних можливостей того або іншого Браузера.

Стандарт CSS вимагає, щоб імена всіх розширень CSS починалися з особливого префікса, що позначає Браузер, який їх підтримує:

- **`-moz`** - позначає Mozilla Firefox;
- **`-o`** - позначає Opera;
- **`-webkit`** - позначає Браузери, засновані на програмному ядрі Webkit. Найвідоміші серед них — Google Chrome і Apple Safari.

Розширення CSS дозволять задіяти можливості Браузера, які не підтримуються стандартом CSS або з'являться тільки в майбутній його версії. Однак, по-перше, розширення CSS найчастіше підтримуються тільки одним Браузером; інші Браузери можуть і не "знати" про їхнє існування. По-друге, розроблювачі Браузера в будь-який момент можуть позбавити своє дітище підтримки тих або інших розширень CSS, порахувавши їх непотрібними.

В цій темі ми розглянемо деякі розширення CSS, самі корисні для Web-дизайнерів.

Багатобарвні рамки

Ми вже знаємо, що за допомогою особливих атрибутів стилю CSS можна створювати одноколірні рамки в будь-яких елементів Web-сторінки.

Однак для рамок, товщина яких перевищує один піксел, ми можемо задати відразу кілька кольорів. В цьому випадку рамка буде представлена Браузером як набір вкладених друг у друга рамок товщиною в один піксел. Перший із заданих нами кольорів буде застосований до самої зовнішньої рамки, другий — до вкладеної в неї рамки, третій — до рамки, вкладеної в ту, яка вкладена в зовнішню, і т.д.

Багатобарвні рамки створюють за допомогою розширень CSS **-moz-border-left-colors, -moz-border-top-colors, -moz-border-right-colors і -moz-border-bottom-colors.** Вони задають кольори, відповідно, для лівої, верхньої, правої і нижньої сторони рамок.

-moz-border-left-colors|-moz-border-top-colors|-moz-border-right-colors|-moz-border-border-colors:<*набір кольорів, розділених пробілами*>|none

Значення **none** забирає "багатоколірність" у відповідної сторони рамки. Це значення за замовчуванням.

Дані розширення CSS підтримуються тільки Firefox і не включені в стандарт CSS.

Приклад:

```
#cheader {width: 1010px;
        padding: 0 20px;
        border-bottom: medium dotted;
        -moz-border-bottom-colors: #B1BEC6 #F8F8F8 #B1BEC6}
```

Тут ми задаємо для контейнера **cheader** рамку, що складається з однієї нижньої сторони, середньої товщини, із трьома кольорами.

Рамки з округленими кутами

Рамки із прямокутними кутами зустрічаються дуже часто і вже встигнули намуляти очі. От рамки з округленими кутами — інша справа!

Розширення CSS **-moz-border-radius-topleft** (Firefox), **border-top-left-radius** (Opera і стандарт CSS 3) і **-webkit-border-top-left-radius** (Браузери на програмному ядрі Webkit) задають радіус округлення верхнього лівого кута рамки:

- **-moz-border-radius-topleft|border-top-left-radius| -webkit-border-top-left-radius: *<значення 1> [<значення 2>]***

Якщо вказано одне значення, воно задасть радіус округлення і по горизонталі, і по вертикалі. Якщо вказано два значення, то перше задасть радіус округлення по горизонталі, а друге — по вертикалі, створюючи тим самим округлення еліптичної форми. Радіус округлення може бути заданий в будь-якій одиниці виміру, підтримуванійї CSS.

Для вказівки радіуса округлення інших кутів рамки призначені такі розширення CSS:

- **-moz-border-radius-topright** (Firefox), **border-top-right-radius** (Opera і стандарт CSS 3) і **-webkit-border-top-right-radius** (Браузери на програмному ядрі Webkit) — радіус округлення верхнього правого кута рамки.
- **-moz-border-radius-bottomright** (Firefox), **border-bottom-right-radius** (Opera і стандарт CSS 3) і **-webkit-border-bottom-right-radius** (Браузери на програмному ядрі Webkit) — радіус округлення нижнього правого кута рамки.
- **-moz-border-radius-bottomleft** (Firefox), **border-bottom-left-radius** (Opera і стандарт CSS 3) і **-webkit-border-bottom-left-radius** (Браузери на програмному ядрі Webkit) — радіус округлення нижнього лівого кута рамки.

Формат їх використання такий же, як у розширень CSS, описаних на початку цієї теми (лістинг 15.1).

Лістинг 15.1

```
#cheader {width: 1010px;
        padding: 0 20px;
        border-bottom: medium dotted;
        -moz-border-radius-bottomleft:
        2px;
        -moz-border-radius-bottomright:
        2px;
        border-bottom-left-radius: 2px;
```

```
border-bottom-right-radius: 2px;
-webkit-border-bottom-left-radius:
2px;
-webkit-border-bottom-right-radius: 2px}
```

Тут ми задаємо для контейнера **cheader** рамку, що складається з однієї нижньої сторони, що має радіуси округлення нижнього лівого і нижнього правого кутів, рівні двом пікселам, відразу для всіх Браузерів, що підтримують цю можливість. Таким чином, і Firefox, і Opera, і Браузери на програмному ядрі Webkit виведуть ці кути рамки округленими.

Розширення CSS **-moz-border-radius** (Firefox), **border-radius** (Opera і стандарт CSS) і **-webkit-border-radius** (Браузери на програмному ядрі Webkit) дозволяють задати радіуси округлення відразу для всіх кутів рамки:

```
-moz-border-radius|border-radius|-webkit-border-radius:
<позиція 1 значення 1>[/<позиція 1 значення 2>]
[<позиція 2 значення 1>[/<позиція 2 значення 2>]
[<позиція 3 значення 1>[/<позиція 3 значення 2>]
[<позиція 4 значення 1>[/<позиція 4 значення 2>]]]]
```

Як бачимо, кожна позиція може містити як одне значення, так і пари значень, розділених слешем **/**. Якщо вона містить одне значення, це значення задасть радіус с округлення і по горизонталі, і по вертикалі. Якщо ж у позиції вказати два значення, розділивши їх слешем, перше задасть радіус округлення по горизонталі, друге — по вертикалі.

Крім того, можна вказати від однієї до чотирьох позицій.

- Якщо вказана одна позиція, вона задасть радіус округлення всіх кутів рамки.
- Якщо вказані дві позиції, перша задасть радіус округлення верхнього лівого і нижнього правого кутів рамки, а друга — верхнього правого і нижнього лівого кутів.
- Якщо вказані три позиції, перша задасть радіус округлення верхнього лівого кута рамки, друга — верхнього правого і нижнього лівого, а третя — нижнього правого кута.
- Якщо вказані чотири позиції, перша задасть радіус округлення верхнього лівого кута рамки, друга — верхнього правого, третя — нижнього правого, а четверта — нижнього лівого кута.

Приклад ілюструє лістинг 15.2.

Лістинг 15.2

```
#navbar LI {padding: 5px 10px;
        margin: 10px 0px;
        border: thin solid #B1BEC6;
        -moz-border-radius: 3px/1px 3px/1px 0px 0px;
        border-radius: 3px/1px 3px/1px 0px 0px;
        -webkit-border-radius: 3px/1px 3px/1px 0px
        0px;
        cursor: pointer}
```

Тут ми задаємо для пунктів "зовнішніх" списків, що формують смугу навігації, рамки з округленими верхніми кутами. Радіус округлення їх по горизонталі буде 3 піксела, а по вертикалі — 1 піксел.

Виділення з округленими кутами

Firefox, що дозволив створювати рамки з округленими кутами, дозволив округляти кути у виділення. Розширення CSS **-moz-outline-radiu-topleft** задає радіус округлення верхнього лівого кута виділення:

-moz-outline-radius-topleft: *<значення>*

Радіус округлення може бути заданий в будь-якій одиниці виміру, підтримуваній CSS.

Для вказівки радіуса округлення інших кутів виділення застосовуються такі розширення CSS:

- **-moz-outline-radius-topright** — радіус округлення верхнього правого кута виділення.
- **-moz-outline-radius-bottomright** — радіус округлення нижнього правого кута виділення.
- **-moz-outline-radius-bottomleft** — радіус округлення нижнього лівого кута виділення.

Формат їх використання такий же, як у розширень CSS, описаних на початку цієї теми.

Лістинг 15.3 ілюструє приклад.

Лістинг 15.3

```
DFN {outline: thin dotted #B1BEC6;
    -moz-outline-radius-topleft: 3px;
```

```
-moz-outline-radius-topright: 3px;
-moz-outline-radius-bottomright: 3px;
-moz-outline-radius-bottomleft: 3px}
```

Тут ми задаємо для всіх фрагментів тексту, позначених тегом **<DFN>,** виділення, усі кути якого мають радіус округлення 3 піксела.

Розширення CSS **-moz-outline-radius** дозволяє задати радіуси округлення відразу для всіх кутів виділення:

-moz-outline-radius: **<значення 1> [<значення 2> [<значення 3> [<значення 4>]]|*inherit***

Тут можна вказати від одного до чотирьох значень.

- Якщо вказане одне значення, воно задасть радіус округлення всіх кутів виділення.
- Якщо вказані два значення, перше задасть радіус округлення верхнього лівого і нижнього правого кутів виділення, а друге — верхнього правого і нижнього лівого кутів.
- Якщо вказані три значення, перше задасть радіус округлення верхнього лівого кута виділення, друге — верхнього правого і нижнього лівого, а третє — нижнього правого кута.
- Якщо вказані чотири значення, перше задасть радіус округлення верхнього лівого кута виділення, друге — верхнього правого, третє — нижнього правого, а четверте — нижнього лівого кута.

Ці розширення CSS підтримуються тільки Firefox і не включені в стандарт CSS 3.

Приклад:

```
DFN {outline: thin dotted #B1BEC6; -moz-outline-radius: 3px}
```

Багатоколоночна верстка

Багатоколоночна верстка — це те, чого давно не вистачало в Web-дизайні. Окремі ентузіасти вже давно реалізовували її за допомогою таблиць або контейнерів.

Але тепер у них є "законні" засоби розбити текст на довільне число колонок, скориставшись особливими розширеннями CSS.

Розширення CSS **-moz-column-count** (Firefox) і **-webkit-column-count** (Браузери на програмному ядрі Webkit) задають бажане число колонок для багатоколоночної верстки:

-moz-column-count|-webkit-column-count: <*число колонок*>|auto

Реальне число колонок, яке виведе Браузер, може бути іншим; наприклад, на Web-сторінці може не виявитися місця для вказаного числа колонок або для розміщення тексту може знадобитися менше колонок, чим було вказано.

Значення **auto** скасовує багатоколоночну верстку. Приклад:

```
#cmain {-moz-column-count: 2;
      -webkit-column-count: 2}
```

Тут ми задаємо для контейнера **cmain** дві колонки.

Розширення CSS **-moz-column-width** (Firefox) і **-webkit-column-width** (Браузери на програмному ядрі Webkit) задають бажану ширину колонок:

-moz-column-width|-webkit-column-width: *<ширина колонок>* auto

Реальна ширина колонок, встановлена Браузером, може бути більше або менше і буде залежати від ширини елемента Web-сторінки, вміст якого верстається в декілька колонок.

Значення **auto** повертає керування шириною колонок Браузеру.

Лістинг 15.4 ілюструє приклад.

Лістинг 15.4

```
#cmain {-moz-column-count: 2;
      -webkit-column-count: 2;
      -moz-column-width: 300px;

      -webkit-column-width: 300px}
```

Задаємо для контейнера **cmain** ширину колонок в 300 пікселів.

Розширення CSS **-moz-column-gap** (Firefox) і **-webkit-column-gap** (Браузери на програмному ядрі Webkit) задають ширину простору між колонками:

-moz-column-gap|-webkit-column-gap: *<ширина простору між колонками>* |normal

Значення **normal** задає ширину простору між колонками за замовчуванням. Її величина залежить від Браузера.

Лістинг 15.5 ілюструє приклад.

Лістинг 15.5

```
#cmain {-moz-column-count: 2;
      -webkit-column-count: 2;
      -moz-column-width: 300px;
      -webkit-column-width: 300px;
      -moz-column-gap: 5mm;
      -webkit-column-gap: 5mm}
```

В лістингу 15.5 задаємо для контейнера **cmain** ширину простору між колонками 5 мм.

Розширення CSS **-moz-column-rule-width** (Firefox) і **-webkit-column-rule-width** (Браузери на програмному ядрі Webkit) задають товщину ліній, якими колонки будуть відділятися друг від друга:

-moz-column-rule-width|-webkit-column-rule-width:
thin|medium| thick| *<товщина ліній між колонками>*

Тут доступні ті ж значення, що і для атрибутів стилю, що задають товщину ліній рамки і виділення.

Лістинг 15.6 ілюструє приклад.

Лістинг 15.6

```
#cmain {-moz-column-count: 2;
      -webkit-column-count: 2;
      -moz-column-width: 300px;
      -webkit-column-width: 300px;
      -moz-column-gap: 5mm;
      -webkit-column-gap: 5mm;
      -moz-column-rule-width: thin;
      -webkit-column-rule-width:
      thin}
```

Тепер між колонками в контейнері **cmain** будуть проведені тонкі лінії.

Розширення CSS **-moz-column-rule-style** (Firefox) і **-webkit-column-rule-style** (Браузери на програмному ядрі Webkit) задають стиль ліній, якими колонки будуть відділятися друг від друга:

-moz-column-rule-style|-webkit-column-rule-style: none|hidden|dotted|dashed|solid|double|groove|ridge|inset|outs

Тут доступні ті ж значення, що і для атрибутів стилю, що задають стиль ліній рамки і виділення.

Лістинг 15.7 ілюструє приклад.

Лістинг 15.7

```
#cmain { -moz-column-count: 2;
      -webkit-column-count: 2;
      -moz-column-width: 300px;
      -webkit-column-width: 300px;
      -moz-column-gap: 5mm;
      -webkit-column-gap: 5mm;
      -moz-column-rule-width: thin;
      -webkit-column-rule-width:
      thin; -moz-column-rule-style:
      dotted;
      -webkit-column-rule-style: dotted
      }
```

Тепер між колонками в контейнері **cmain** будуть проведені тонкі точкові лінії.

Розширення CSS **-moz-column-rule-color** (Firefox) і **-webkit-column-rule-color** (Браузери на програмному ядрі Webkit) задають колір ліній, якими колонки будуть відділятися друг від друга:

-moz-column-rule-color|-webkit-column-rule-color: ***<колір ліній між колонками>***

Лістинг 15.8 ілюструє приклад.

Лістинг 15.8

```
#cmain { -moz-column-count: 2;
      -webkit-column-count: 2;
      -moz-column-width: 300px;
```

```
-webkit-column-width: 300px;
-moz-column-gap: 5mm;
-webkit-column-gap: 5mm;
-moz-column-rule-width: thin;
-webkit-column-rule-width:
thin;
-moz-column-rule-style: dotted;
-webkit-column-rule-style: dotted;
-moz-column-rule-color: #B1BEC6;
-webkit-column-rule-color: #B1BEC6}
```

Тепер між колонками в контейнері **cmain** будуть проведені тонкі точкові лінії ясно-синього кольору.

Розширення CSS **-moz-column-rule** (Firefox) і **-webkit-column-rule** (Браузери на програмному ядрі Webkit) задають відразу всі параметри ліній, якими колонки будуть відділятися друг від друга:

-moz-column-rule|-webkit-column-rule: *<товщина> <стиль> <колір>*

Лістинг 15.9 ілюструє приклад.

Лістинг 15.9

```
#cmain {-moz-column-count: 2;
      -webkit-column-count: 2;
            -moz-column-width:
                        300px;
      -webkit-column-width: 300px;
      -moz-column-gap: 5mm;
      -webkit-column-gap: 5mm;
      -moz-column-rule: thin dotted #B1BEC6;
      -webkit-column-rule: thin dotted #B1BEC6}
```

На жаль, Opera на даний момент не підтримує багатоколоночну верстку.

Перетворення CSS

За допомогою особливих розширень CSS ми можемо змістити, повернути, розтягти або стиснути будь-який елемент Web-сторінки:.

Для вказівки, які саме перетворення слід виконати над елементом Web-сторінки, служать розширення CSS **-moz-transform** (Firefox), **-o-transform** (Opera) і **-webkit-transform** (Браузери на програмному ядрі Webkit):

-moz-transform|-o-transform|-webkit-transform: <*перетворення*>

Доступних перетворень не так вуж і багато. Зараз ми їх розглянемо.

Перетворення **rotate** дозволяє повернути елемент Web-сторінки на вказаний кут за годинниковою стрілкою:

***rotate*(<кут>)**

Значення *кута* може бути задане в трьох одиницях виміру: градусах, радіанах і градах. Якщо потрібно вказати кут в радіанах, після самого числового значення кута ставлять позначення **deg**, в радіанах — **rad**, в градах — **grad**. При цьому між числом і позначенням одиниці виміру кута не повинний бути пробілів.

Тут ми повертаємо контейнер **cheader** на 3,14 радіан (приблизно 180°):

```
#cheader {-moz-transform: rotate(3.14rad);
      -o-transform: rotate(3.14rad); -
      webkit-transform: rotate(3.14rad)}
```

А тут буде виконаний поворот контейнера **cheader** на 30°:

```
#cheader {-moz-transform: rotate(30deg);
      -o-transform: rotate(30deg); -
      webkit-transform: rotate(30deg)}
```

Перетворення **scale** дозволяє змінити масштаб елемента Web-сторінки по горизонталі і вертикалі, розтягши його або стиснувши:

***scale*(<масштаб *1*>[, <масштаб *2*>])**

Якщо вказано одне значення, воно задає масштаб і по горизонталі, і по вертикалі. Якщо вказані два значення, перше задає масштаб по горизонталі, друге — по вертикалі. Значення більше 1 задають збільшення масштабу (розтягання), а значення менше 1 — зменшення (стиск); значення 1 залишає масштаб без змін.

Тут збільшуємо масштаб контейнера **cheader** вдвічі по горизонталі і вертикалі, тим самим розтягуючи його:

```
#cheader {-moz-transform: scale(2);
        -o-transform: scale(2);
        -webkit-transform: scale(2)}
```

А тут збільшуємо масштаб контейнера **cheader** вдвічі по горизонталі і зменшуємо його вдвічі по вертикалі:

```
#cheader {-moz-transform: scale(2, 0.5);
        -o-transform: scale(2, 0.5);
        -webkit-transform: scale(2, 0.5)}
```

Перетворення **scaleX** і **scaleY** дозволяють змінити масштаб елемента Web-сторінки, відповідно, тільки по горизонталі і тільки по вертикалі:

```
scaleX|scaleY(<масштаб>)
#cheader {-moz-transform: scaleX(2);
        -o-transform: scaleX(2);
        -webkit-transform: scaleX(2)}
```

Перетворення **skew** дозволяє зрушити елемент Web-сторінки по горизонтальній і вертикальній осі на заданий кут, тим самим "скособочив" його:

```
skew(<кут 1> [, <кут 2>])
```

Якщо вказане одне значення, воно задає кут зрушення і по горизонтальній, і по вертикальній осі. Якщо вказано два значення, перше задає кут зрушення по горизонтальній, друге — по вертикальній осі. Кути можуть бути задані в будь-яких одиницях виміру.

Зрушуємо контейнер **cheader** по горизонтальній осі на 10°. По вертикальній осі він зрушений не буде, оскільки ми задали кут зрушення 0°:

```
#cheader {-moz-transform: skew(10deg, 0deg);
        -o-transform: skew(10deg, 0deg);
        -webkit-transform: skew(10deg, 0deg)}
```

Перетворення **skewX** і **skewY** дозволяють зрушити елемент Web-сторінки, відповідно, тільки по горизонтальній і тільки по вертикальній осі.

Приклад:

```
#cheader {-moz-transform: skew(10deg, 0deg);
        -o-transform: skew(10deg, 0deg);
        -webkit-transform: skew(10deg, 0deg)}
```

Перетворення **translate** дозволяє змістити елемент Web-сторінки по горизонтальній і вертикальній осі на задану відстань:

```
translate(<відстань 1>[, <відстань 2>])
```

Якщо вказане одне значення, воно задає відстань для зсуву і по горизонтальній, і по вертикальній осі. Якщо вказані два значення, перше задає відстань для зсуву по горизонтальній, друге — по вертикальній осі. Відстані можуть бути задані в будь-яких одиницях виміру, підтримуваних CSS.

Зміщаємо контейнер **cheader** на 5 мм по горизонтальній осі іна 2 мм по вертикальній:

```
#cheader {-moz-transform: translate(5mm, 2mm);
          -o-transform: translate(5mm, 2mm);
          -webkit-transform: translate(5mm, 2mm)}
```

Перетворення **translateX** і **translateY** дозволяють змістити елемент Web-сторінки, відповідно, тільки по горизонтальній і тільки по вертикальній осі.

Зміщаємо контейнер **cheader** на 2 мм по вертикальній осі:

```
translateX|translateY(<расстояние>)
#cheader {-moz-transform: translateY(2mm);
        -o-transform: translateY(2mm);
        -webkit-transform:
        translateY(2mm)}
```

Ми можемо піддати елемент Web-торінки відразу декільком перетворенням. Для цього слід записати ці перетворення одне за іншим, розділивши їх пробілами. Перетворення будуть застосовуватися до елемента в тому порядку, у якім вони записані.

Приклад:

```
#cheader {-moz-transform: skewX(10deg) translateY(2mm);
        -o-transform: skewX(10deg) translateY(2mm);
        -webkit-transform: skewX(10deg) translateY(2mm)}
```

Тут ми піддаємо контейнер **cheader** спочатку зрушенню по горизонтальній осі на 10°, а потім — зсуву по вертикальній осі на 2 мм.

За замовчуванням усі перетворення виконуються щодо центру елемента Web-сторінки. Так, якщо ми повернемо елемент на якийсь кут, він повернеться щодо свого центру. Але ми можемо вказати іншу точку, щодо якої будуть виконуватися всі наступні перетворення. Для цього служать розширення CSS **-moz-transform-origin** (Firefox), **-o-transform-origin** (Opera) і **-webkit-transform-origin** (Браузери на програмному ядрі Webkit):

```
-moz-transform-origin|-o-transform-origin|-webkit-
transform-origin: <координата>1left|center|right
[<координата>1top|center|bottom]
```

Якщо задано одне значення, воно вкаже координату точки, щодо якої будуть виконуватися перетворення, по горизонтальній осі; координатою цієї точки по вертикальній осі стане центр елемента Web-сторінки. Якщо вказно дваа значення, перше задасть координату точки по горизонтальній осі, друге — по вертикальній.

Саме значення координати може бути задане в будь-якій абсолютній або відносній одиниці виміру, підтримуванійї CSS. Також можна вказати значення

- **left** (лівий край елемента Web-Сторінки),
- **center** (центр),
- **right** (правий край),
- **top** (верхній край) і
- **bottom** (нижній край).

Лістинг 15.10 ілюструє приклад.

Лістинг 15.10

```
#cheader {-moz-transform: rotate(30deg);
        -o-transform: rotate(30deg);
        -webkit-transform: rotate(30deg);
        -moz-transform-origin: left;
        -o-transform-origin: left;
        -webkit-transform-origin: left}
```

Тут ми повертаємо контейнер **cheader** на 30° щодо точки, яка знаходиться в середині його лівого краю. Ми задали як координата точки, щодо якої будуть виконуватися перетворення (в тому числі і поворот), значення **left** (лівий край контейнера) — це її координата по горизонталі; як вертикальнаї координата, оскільки ми її не вказали, буде прийнята середина контейнера.

Ще один приклад ілюструє лістинг 15.11.

Лістинг 15.11

```
#cheader {-moz-transform: rotate(30deg);
        -o-transform: rotate(30deg);
        -webkit-transform: rotate(30deg);
        -moz-transform-origin: right bottom;
        -o-transform-origin: right bottom;
        -webkit-transform-origin: right bottom}
```

Тепер контейнер **cheader** буде повернений на 30° щодо свого нижнього правого кута.

КОНТРОЛЬНІ ПИТАННЯ

1. *Що таке розширення CSS?*
2. Що вимагає Стандарт CSS щодо імен усіх розширень CSS?
3. Як виглядають багатоколірні рамки? Який Браузер підтримує їхнє створення?
4. Створення багатоколірних рамок.
5. Створення рамок з округленими кутами.
6. Виділення з округленими кутами.
7. *Розширення CSS, що дозволяють здійснити* багатоколоночну верстку.
8. Які параметри задають багатоколоночну верстку?
9. Перетворення CSS.

ПРАКТИЧНІ ПОБОТИ

№ з/п	Найменування тем
	1. Вміст Web-сторінок, мова HTML 5: Вступ до сучасного Web-дизайну та мови опису гіпертекстів (HTML). Основні принципи створення Web-сторінок. Мова HTML5. Структурування та оформлення тексту. Створення таблиць. Графіка й мультимедиа. Засоби навігації.
1.	Створення Web-документу засобами HTML (створення Web-сторінок).
2.	Наповнення Web-документу засобами HTML (структурування тексту)
3.	Наповнення Web-документу засобами HTML (виділення тексту)
4.	Наповнення Web-документу засобами HTML (створення списків та таблиць)
5.	Наповнення Web-документу засобами HTML (вставка малюнків, діаграм, об'єктів мультимедіа)
6.	Наповнення Web-документу засобами HTML (створення Web-форм та елементів управління)
7.	Звязування Web-сторінок в єдиний Web-документ (застосування внутрішніх і зовнішніх гіперпосилань).
	2. Представлення Web-сторінок. Каскадні таблиці стилів: Введення в стилі CSS. Параметри шрифту та фона. Контейнери. Параметри абзаців, списків і відображення. Контейнерний Web-Дизайн. Відступи, рамки та виділення. Параметри таблиць. Спеціальні селектори
8.	Оформлення Web-документу засобами CSS. Створення таблиць стилів.
9.	Оформлення Web-документу засобами CSS (використання параметрів шрифту та фона).
10.	Оформлення Web-документу засобами CSS (використання параметрів абзаців, списків і відображення).
11.	Оформлення Web-документу засобами CSS (створення контейнерного Web-дизайну, полоси навігації).
12.	Оформлення Web-документу засобами CSS (використання параметрів відступів, рамок та виділення).
13.	Оформлення Web-документу засобами CSS (використання параметрів таблиць).
14.	Оформлення Web-документу засобами CSS (використання параметрів спеціальних селекторів).
15.	Оформлення Web-документу засобами CSS (використання розширень CSS).

ЗАВДАННЯ ДЛЯ ПРАКТИЧНОЇ РОБОТИ

Наповнення сайту:

На базі звіту зробити сайт. Сайт повинен складатися з декількох файлів. Зміст та файли пов'язані між собою гіперпосиланнями. Кожний файл, крім одного великого повинен відображатися на екрані комп'ютера без прокруток, цілком займая його. Для великого файлу використовувати внутрішні якоря в оби боки.

Сайт повинен містити нумеровані та маркіровані списки, складні таблиці, малюнки та графіки, об'єкти мультимедіа, розміщені засобами HTML5.

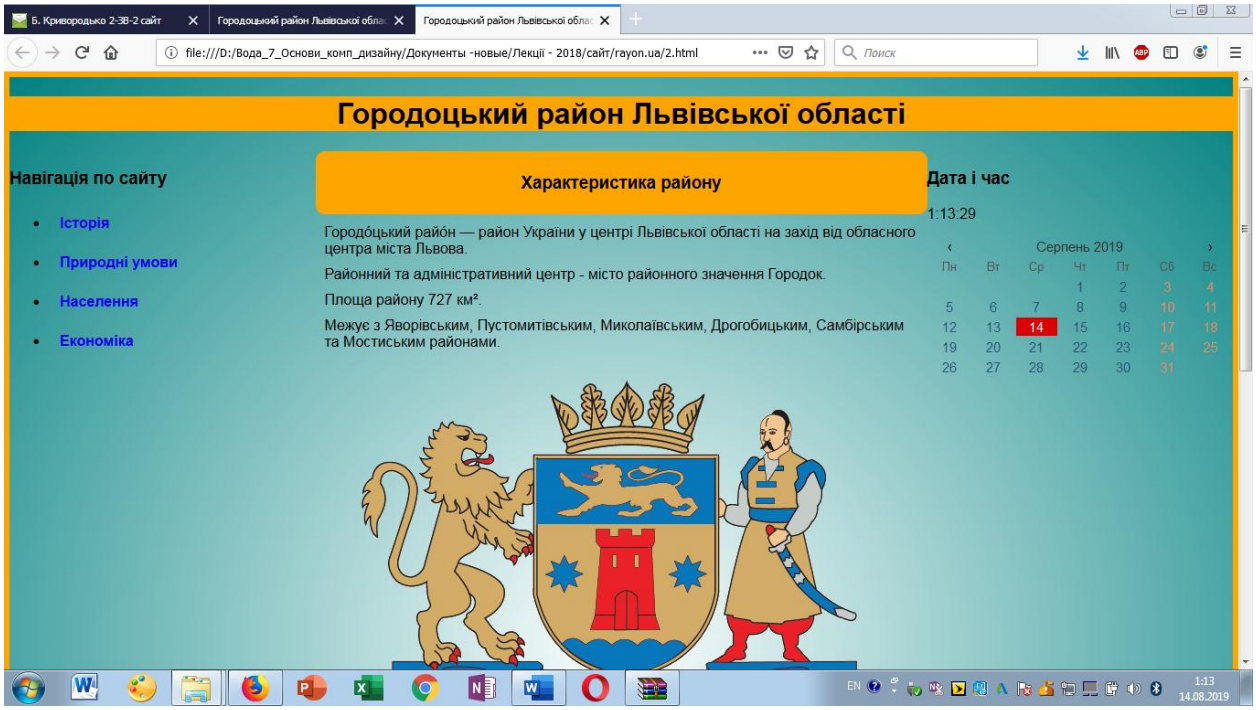
Оформлення сайту:

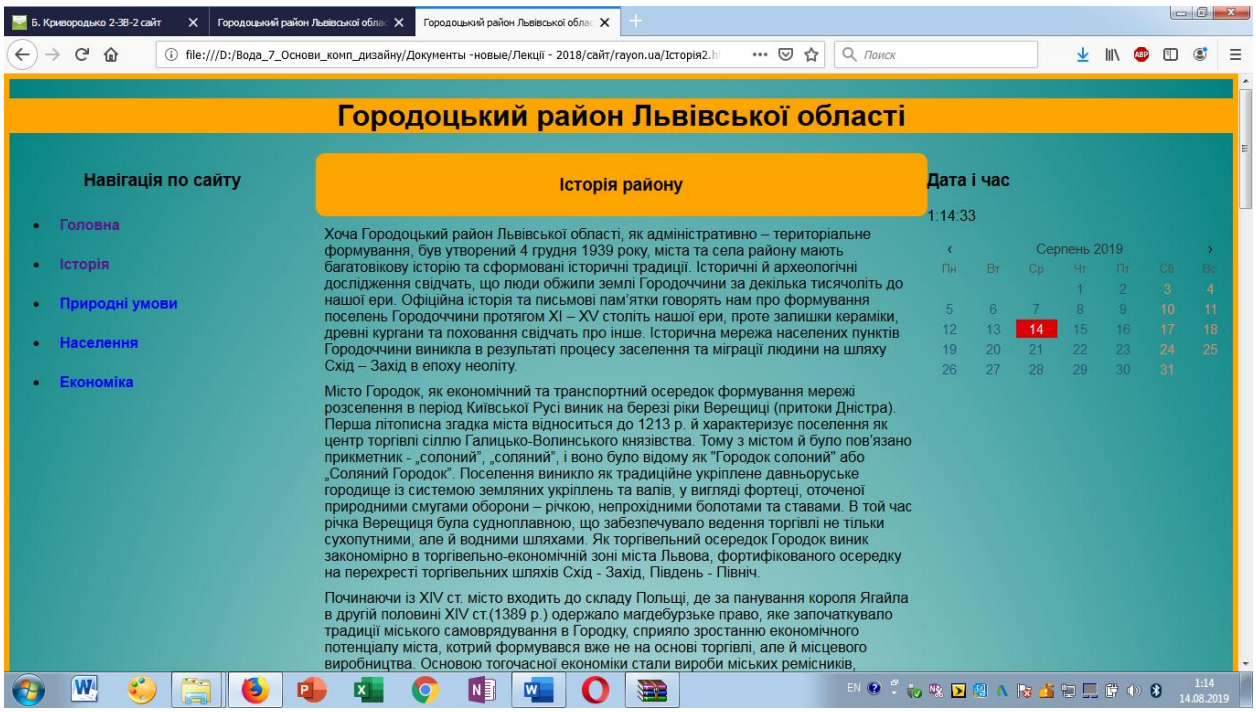
Для оформлення сайту використовувати каскадні таблиці стилів CSS.

Для розміщення інформації використовувати контейнерний Web-дизайн. Кожна сторінка сайту повинна містити заголовок, полосу навігації, в разі потреби відповідні блоки та підвал.

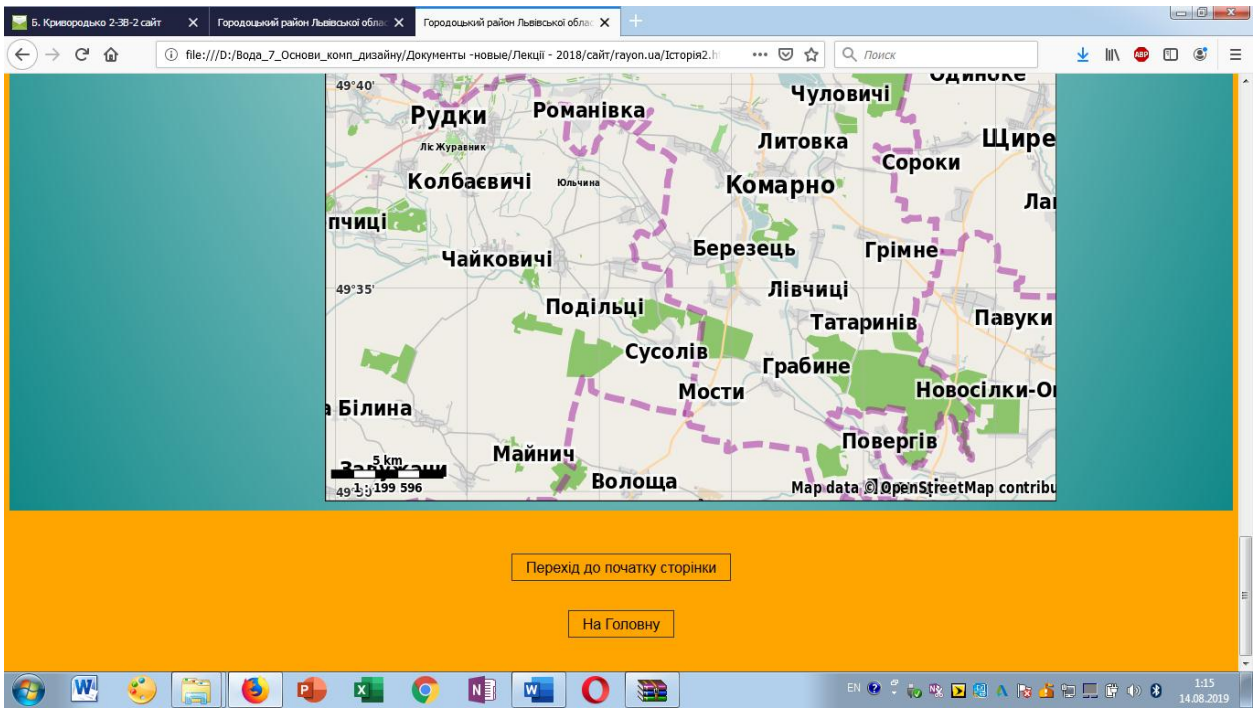
Для оформлення сторінок використовувати різний колір фону, тексту, виділення, горизонтальні лінії і т.і.

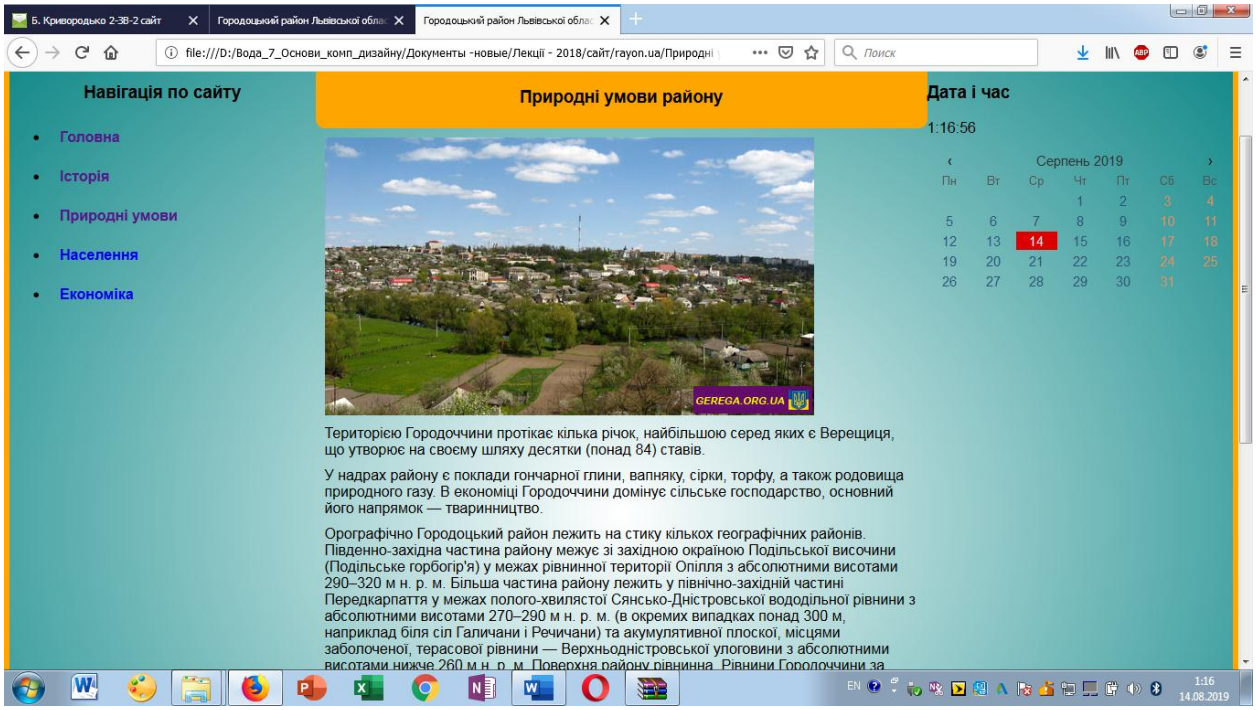
ПРИКЛАД ВИКОНАННЯ РОБОТИ

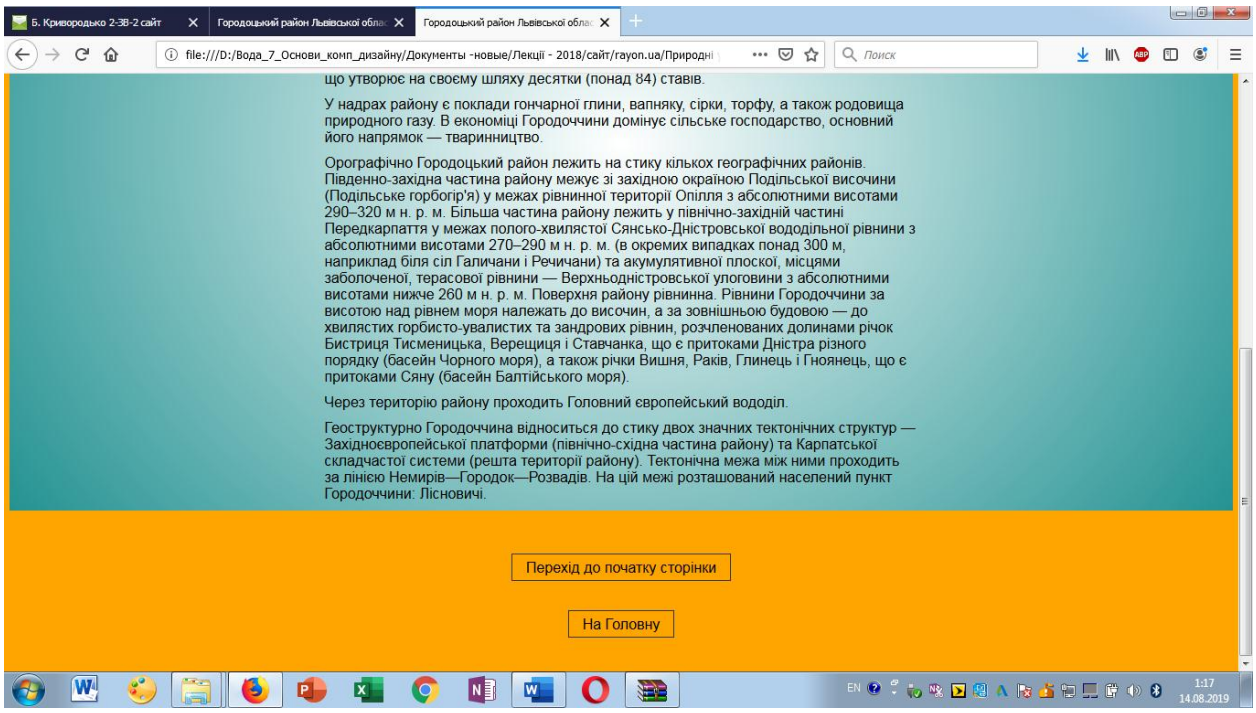


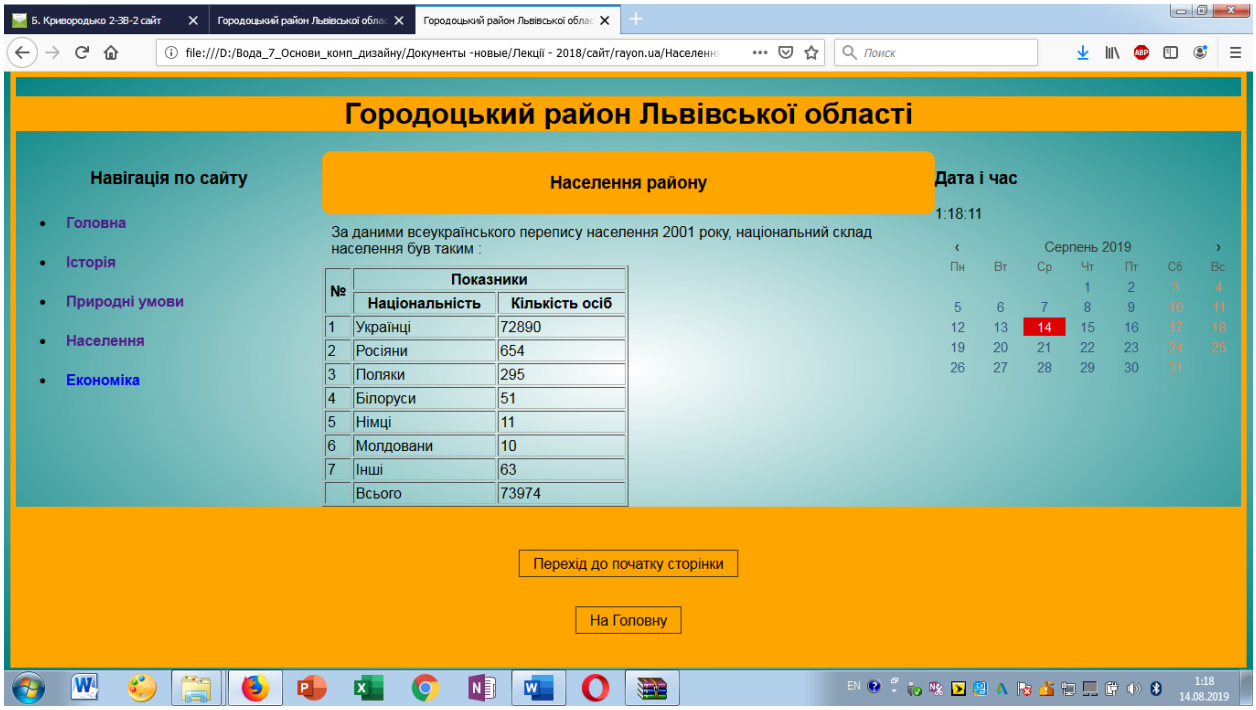


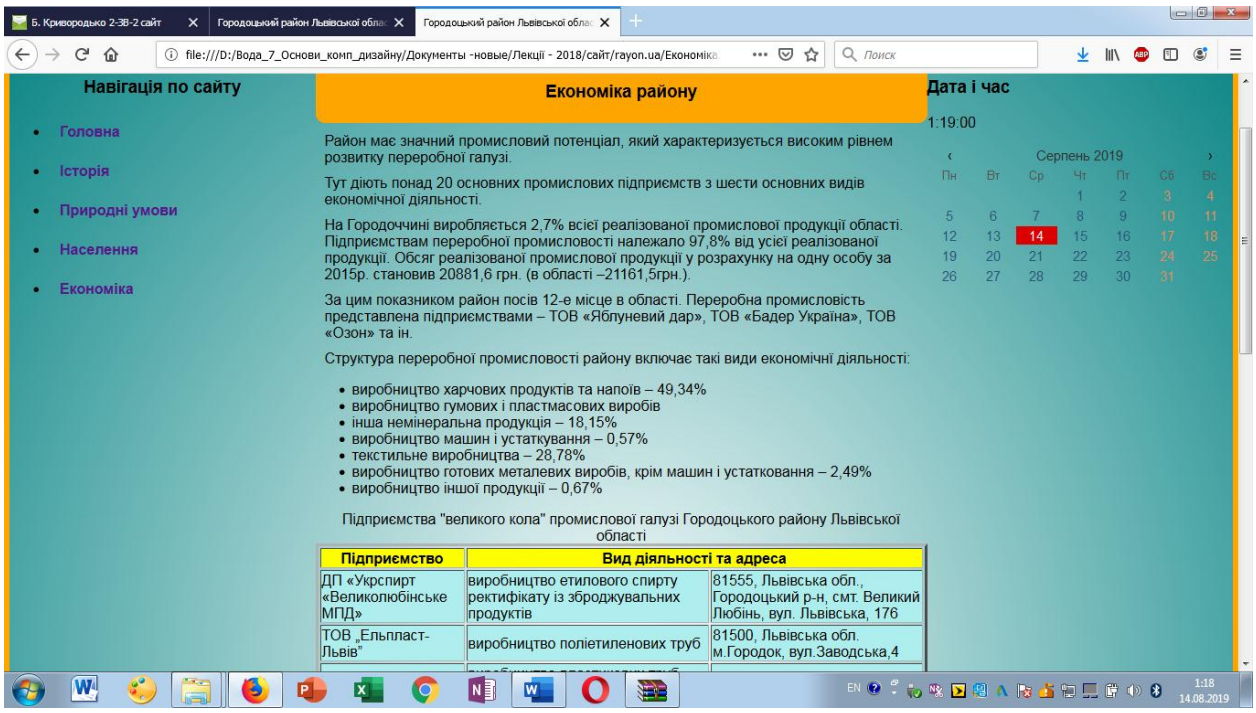
. . . . . .

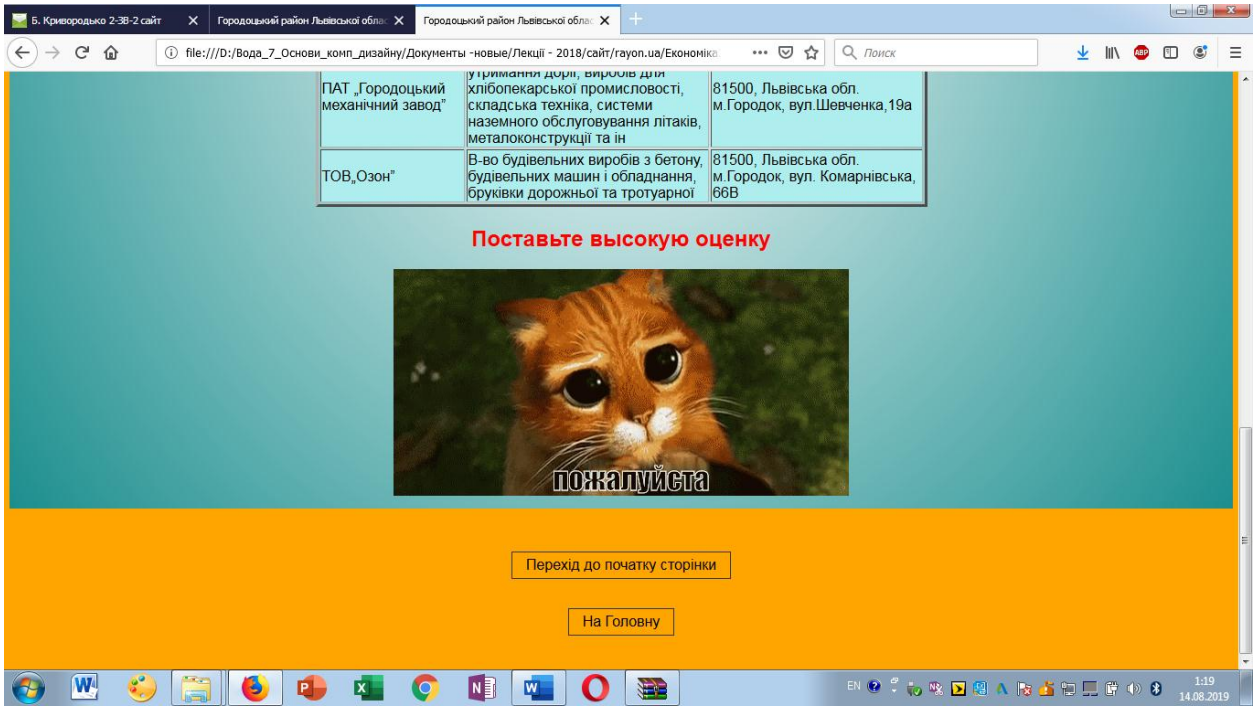

Рудки
Романівка
Чуловичі
Литовка
Сороки
Колбаєвичі
Комарно
Чайковичі
Березець
Грімне
Лівчиці
Подільці
Татаринів
Павуки
Сусолів
Грабине
Мости
Повергів
Майнич
Волоща
Map data © OpenStreetMap contribu
Перехід до початку сторінки
На Головну


Навігація по сайту
Головна
Історія
Природні умови
Населення
Економіка
Природні умови району
Дата і час
Серпень 2019
GEREGA.ORG.UA
Територією Городоччини протікає кілька річок, найбільшою серед яких є Верещиця, що утворює на своєму шляху десятки (понад 84) ставів.
У надрах району є поклади гончарної глини, вапняку, сірки, торфу, а також родовища природного газу. В економіці Городоччини домінує сільське господарство, основний його напрямок — тваринництво.
Орографічно Городоцький район лежить на стику кількох географічних районів. Південно-західна частина району межує зі західною окраїною Подільської височини (Подільське горбогір'я) у межах рівнинної території Опілля з абсолютними висотами 290–320 м н. р. м. Більша частина району лежить у північно-західній частині Передкарпаття у межах полого-хвилястої Сянсько-Дністровської вододільної рівнини з абсолютними висотами 270–290 м н. р. м. (в окремих випадках понад 300 м, наприклад біля сіл Галичани і Речичани) та акумулятивної плоскої, місцями заболоченої, терасової рівнини — Верхньодністровської улоговини з абсолютними


що утворює на своєму шляху десятки (понад 84) ставів.
У надрах району є поклади гончарної глини, вапняку, сірки, торфу, а також родовища природного газу. В економіці Городоччини домінує сільське господарство, основний його напрямок — тваринництво.
Орографічно Городоцький район лежить на стику кількох географічних районів. Південно-західна частина району межує зі західною окраїною Подільської височини (Подільське горбогір'я) у межах рівнинної території Опілля з абсолютними висотами 290–320 м н. р. м. Більша частина району лежить у північно-західній частині Передкарпаття у межах полого-хвилястої Сянсько-Дністровської вододільної рівнини з абсолютними висотами 270–290 м н. р. м. (в окремих випадках понад 300 м, наприклад біля сіл Галичани і Речичани) та акумулятивної плоскої, місцями заболоченої, терасової рівнини — Верхньодністровської улоговини з абсолютними висотами нижче 260 м н. р. м. Поверхня району рівнинна. Рівнини Городоччини за висотою над рівнем моря належать до височин, а за зовнішньою будовою — до хвилястих горбисто-увалистих та зандрових рівнин, розчленованих долинами річок Бистриця Тисменицька, Верещиця і Ставчанка, що є притоками Дністра різного порядку (басейн Чорного моря), а також річки Вишня, Раків, Глинець і Гноянець, що є притоками Сяну (басейн Балтійського моря).
Через територію району проходить Головний європейський вододіл.
Геоструктурно Городоччина відноситься до стику двох значних тектонічних структур — Західноєвропейської платформи (північно-східна частина району) та Карпатської складчастої системи (решта території району). Тектонічна межа між ними проходить за лінією Немирів—Городок—Розвадів. На цій межі розташований населений пункт Городоччини: Лісновичі.
Перехід до початку сторінки
На Головну


Городоцький район Львівської області
Навігація по сайту
Головна
Історія
Природні умови
Населення
Економіка
Населення району
За даними всеукраїнського перепису населення 2001 року, національний склад населення був таким :
№ | Показники | Національність | Кількість осіб
1 | Українці | 72890
2 | Росіяни | 654
3 | Поляки | 295
4 | Білоруси | 51
5 | Німці | 11
6 | Молдовани | 10
7 | Інші | 63
Всього | 73974
Дата і час
1:18:11
Серпень 2019
Перехід до початку сторінки
На Головну


Навігація по сайту
Головна
Історія
Природні умови
Населення
Економіка
Економіка району
Район має значний промисловий потенціал, який характеризується високим рівнем розвитку переробної галузі.
Тут діють понад 20 основних промислових підприємств з шести основних видів економічної діяльності.
На Городоччині виробляється 2,7% всієї реалізованої промислової продукції області. Підприємствам переробної промисловості належало 97,8% від усієї реалізованої продукції. Обсяг реалізованої промислової продукції у розрахунку на одну особу за 2015р. становив 20881,6 грн. (в області –21161,5грн.).
За цим показником район посів 12-е місце в області. Переробна промисловість представлена підприємствами – ТОВ «Яблуневий дар», ТОВ «Бадер Україна», ТОВ «Озон» та ін.
Структура переробної промисловості району включає такі види економічної діяльності:
виробництво харчових продуктів та напоїв – 49,34%
виробництво гумових і пластмасових виробів
інша немінеральна продукція – 18,15%
виробництво машин і устаткування – 0,57%
текстильне виробництва – 28,78%
виробництво готових металевих виробів, крім машин і устаткування – 2,49%
виробництво іншої продукції – 0,67%
Підприємства "великого кола" промислової галузі Городоцького району Львівської області
Підприємство
Вид діяльності та адреса
ДП «Укрспирт «Великолюбінське МПД»
виробництво етилового спирту ректифікату із зброджувальних продуктів
81555, Львівська обл., Городоцький р-н, смт. Великий Любінь, вул. Львівська, 176
ТОВ „Ельпласт-Львів"
виробництво поліетиленових труб
81500, Львівська обл. м.Городок, вул.Заводська,4
Дата і час
1:19:00
Серпень 2019
Пн Вт Ср Чт Пт Сб Вс
1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31


ПАТ „Городоцький механічний завод"
складська техніка, системи наземного обслуговування літаків, металоконструкції та ін
81500, Львівська обл. м.Городок, вул.Шевченка,19а
ТОВ„Озон"
В-во будівельних виробів з бетону, будівельних машин і обладнання, бруківки дорожньої та тротуарної
81500, Львівська обл. м.Городок, вул. Комарнівська, 66В
Поставьте высокую оценку
пожалуйста
Перехід до початку сторінки
На Головну

ПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ

1. World Wide Web та її призначення. Принципи сучасного Web-сайту.
2. Клієнти та сервери Інтернету. Інтернет-адреси.
3. Web-сайти й Web-сервери.
4. Мова HTML і її теги. Вкладеність тегів. Атрибути HTML-тегів.
5. Секції Web-сторінки.
6. Метадані та тип Web-сторінки.
7. Яку структуру має тіло складної Web-сторінки? Які теги використовуються для опису тіла складної Web-сторінки?
8. Як розбити текст на абзаци?
9. Як створити заголовки різного рівня?
10. Які бувають типи списків? Як їх створити?
11. Що таке «Текст фіксованого формату»? Як його вставити в документ і як він відобразиться?
12. Як вставити в документ цитату і як вона відобразиться?
13. Як вставити в текст горизонтальну лінію? Як вставити рядок, що рухається?
14. Як вставити в документ коментар і як він відобразиться?
15. Як вставити в документ адресу і як вона відобразиться?
16. Як можна виділити фрагменти тексту?
17. Як організувати розриви рядків?
18. Вставка неприпустимих символів. Літерали.
19. Таблиці. Як створювати таблиці?
20. Що таке заголовок і секції таблиці? Як їх створювати?
21. Як об'єднати комірки в таблиці по горизонталі та вертикалі?
22. Що таке «впроваджене зображення»?
23. Які формати інтернет-графіки ви знаєте?
24. Розкажіть про формат *GIF*.
25. Розкажіть про формат *JPEG.*
26. Розкажіть про формат *PNG.*
27. Як вставити графічне зображення на Web-сторінку?
28. Розкажіть про формати файлів і формати кодування.
29. Що таке типи MIME?
30. Вставка аудіоролика на Web-сторінку.
31. Вставка відеоролика на Web-сторінку.
32. Як вказати в одному тегу **<AUDIO>** або **<VIDEO>** відразу кілька мультимедійних файлів?
33. Як указати текст заміни, якщо браузер взагалі не підтримує теги **<AUDIO>** і **<VIDEO>?**

34. Що таке «Засоби навігації» Web-сторінок? Якими вони бувають?
35. Створення текстових гіперпосилань.
36. Інтернет-адреси в WWW.
37. Поштові гіперпосилання.
38. «Гарячі» клавіші для створення гіперпосилань.
39. Фокус введення та порядок обходу гіперпосилань.
40. Створення зображень-гіперпосилань.
41. Створення зображень-мап.
42. Що таке *Смуга навігації*?
43. Створення Якоря (внутрішнього гіперпосилання).
44. Для чого потрібні стилі CSS? Перелічіть способи створення стилів.
45. Розкажіть про стиль перевизначення тегу.
46. Розкажіть про стильові класи.
47. Розкажіть про іменований стиль.
48. Розкажіть про комбіновані стилі.
49. Розкажіть про вбудовані стилі.
50. Що таке зовнішні та внутрішні таблиці стилів.
51. Пріоритет стилів.
52. Що таке правила каскадності стилів?
53. Що таке Важливі атрибути стилів?
54. Які стилі в яких випадках слід застосовувати?
55. Як записуються коментарі CSS.
56. Які параметри шрифту ви знаєте?
57. Назвіть параметри, що управляють розривом рядків.
58. Назвіть параметри вертикального вирівнювання.
59. Які параметри тіні у текста ви знаєте?
60. Які параметри фона ви знаєте?
61. Що таке контейнери та вбудовані контейнери? Для чого вони використовуються?
62. Які види контейнерів ви знаєте? Як створити вбудований контейнер?
63. Як задати горизонтальне вирівнювання тексту?
64. Як задати відступ для "червоного рядка"?
65. Які параметри списків ви знаєте?
66. Які параметри відображення ви знаєте?
67. Які параметри курсору ви знаєте?
68. Що таке блокові контейнери?
69. Текстовий, фреймовий і табличний Web-дизайн, їх переваги та недоліки.
70. Сутність контейнерного Web-дизайну.
71. Дві групи стилів, що задають параметри контейнерів.

72. Параметри розмірів для контейнерів.
73. Параметри розміщення для контейнерів. Плаваючі контейнери.
74. Параметри переповнення. Контейнери із прокручуванням.
75. Параметри відступів.
76. Параметри рамки.
77. Як створити повну смугу навігації?
78. Виділення. Параметри виділення.
79. Параметри таблиць: параметри вирівнювання.
80. Параметри таблиць: параметри відступів і рамок.
81. Параметри таблиць: параметри розмірів.
82. Параметри таблиць: параметр, що вказує місце розташування заголовка таблиці щодо самої таблиці.
83. Параметри таблиць: параметр, що вказує як виводити на екран порожні комірки.
84. Що таке спеціальні селектори? Які різновидів спеціальних селекторів існують?
85. Що таке комбінатори? Які види комбінаторів існують?
86. Що таке селектори по атрибутах тегу? Які види селекторів по атрибутах тегу існують?
87. Що таке псевдоелементи? Які види псевдоелементів існують?
88. Що таке псевдокласи? Які види псевдокласів існують?
89. Розкажіть про псевдокласи гіперпосилань.
90. Структурні псевдокласи. Псевдокласи :not і *.

ЛІТЕРАТУРА

Базова:

1. Гоше Х. Д. HTML5. Для профессионалов. — СПб.: Питер, 2013. — 460 с.
2. Дронов В. А. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов. — СПб.: БХВ-Петербург, 2011. — 416 с.
3. Дунаев В. В. Основы Web-дизайна. Самоучитель. —. — СПб.: БХВ-Петербург, 2012. — 480 с.
4. Кантор И. Язык JavaScript. – 634 с.
5. Квинт И. HTML, XHTML и CSS на 100 %. — СПб.: Питер, 2010. — 384 с.
6. Клименко Р. А. Веб-мастеринг на 100%. — СПб.: Питер, 2013. — 512 с.
7. Макфарланд Д. С. Большая книга CSS3. — СПб.: Питер, 2014. — 608 с.
8. Русаков М. Создание сайта от начала до конца. – 172 с.
9. Хоган Б. HTML5 и CSS3. Веб-разработка по стандартам нового поколения. -. СПб.: Питер, 2014. — 320 с. — (Серия «Библиотека программиста»).
10. Херман Д. Сила JavaScript. 68 способов эффективного использования JS. — СПб.: Питер, 2013. — 288 с. — (Серия «Библиотека специалиста»).
11. Хоган Б., Уоррен К., Уэбер М., Джонсон К., Годин А. Книга веб-программиста: секреты профессиональной разработки веб-сайтов. - СПб.: Питер, 2013. — 288 с.

Допоміжна:

1. Джилленуотер З. Сила CSS3. Освой новейший стандарт веб-разработок! — СПб.: Питер. 2012. — 304 с.
2. Кит Дж. HTML5 для веб-дизайнеров. - *Текст предоставлен издательством http ://www. litres. ru/pages/biblio_book/?art=4570175 HTML5 для веб-дизайнеров/Джереми Кит: Манн, Иванов и Фербер; Москва; 2013.* – 80 с.
3. Лоусон Б., Шарп Р. Изучаем HTML5. Библиотека специалиста. — СПб.: Питер, 2011. — 272 с.
4. Сырых Ю.А. Современный веб-дизайн. Эпоха Веб 3.0. - М.: ООО "И.Д. Вильямс", 2013. - 368 с.

Інформаційні ресурси:

1. MyRusakov.ru
2. vlad@htmlbook.ru
3. www.html5book.ru
4. www.htmlbook.ru
5. learn.javascript.ru

ЗМІСТ