Лекція 6.

ОСНОВИ CSS. Селектори. Властивості

6.1. Концепція CSS

Практично кожен HTML-тег підтримує включення одного або декількох необов'язкових атрибутів. Деякі атрибути впливають на спосіб роботи тега або спосіб його обробки веб-браузером. Інші атрибути не впливають на функціональність, а скоріше на вигляд елемента на веб-сторінці. У попередніх версіях стандарту HTML часто було проблемою для веб-розробників вивчити і запам'ятати всі різні атрибути, які могли б вплинути на те, як браузер роздоб'є візуальні компоненти веб-сторінки. Але починаючи зі стандарту HTML 4.01 всі ці атрибути були об'єднані в загальнодоступний атрибут, який має назву *style*.

Стиль – це об’єднання одного або кількох атрибутів, які застосовуються до візуальних компонентів веб-сторінки. Такі компоненти можуть включати текст, графіку, кнопки введення і елементи веб-сторінки. Найпростішим способом використання стилів є включення їх з індивідуальним тегом HTML.

Розглянемо наступний приклад:

<body style=”background-color:#E6E6FA”>

Цей стиль застосовує колір фону до всього тіла веб-сторінки. Якщо ви бажаєте застосувати інший атрибут до цієї веб-сторінки (наприклад, креслення шрифту), ви можете створити наступний тег:

```
<body style=”background-color:#E6E6FA;
  font-family:New York”>
```

Такий підхід до використання стилів працює чудово в деяких ситуаціях, але це не завжди найкращий варіант. Розглянемо, наприклад, впорядкований або невпорядкований список на веб-сторінці, що містить десятки, або, можливо, сотні елементів. В цьому випадку аж занадто обтяжливим буде застосування стилю до кожного тега <li> у списку. Звичайно, було б дуже

нудно набирати, або навіть копіювати і вставляти, той самий атрибут стилю на десятки або сотні <li> тегів. З цієї причини HTML підтримує концепцію каскадного стилю.

Каскадний стиль – це об'єднання одного або кількох атрибутів, які можна визначити один раз у заголовку веб-сторінки, а потім застосувати до багатьох екземплярів певного теґу.

Розглянемо наступний приклад:

```
<style type=”text/css”>
li {font-family:Verdana;font-color:red}
</style>
```

Використовуючи цей каскадний стиль, атрибути сімейства шрифтів (накреслення та колір) буде застосовано до кожного з тегів <li> в поточному HTML-документі. Крім того, кілька тегів можуть бути включені до певного каскадного стилю. Теги, показані на рисунку 5–1, створюють каскадний стиль для кількох частин веб-сторінки. Каскадні стилі економлять веб-дизайнерам величезну кількість часу.

FIGURE 5–1 A cascading style definition

```
<style type="text/css">
  h1 {color:red}
  h2 {color:blue}
  h3 {color:purple}
  p {color:black}
</style>
```

Розглянемо докладніше концепцію каскадних таблиць стилів або CSS.

CSS (Cascading Style Sheets – Каскадні таблиці стилів) – мова графічного дизайну для визначення та створення презентації структурованого документа, написаного мовою розмітки. Вони широко використовується для встановлення візуального оформлення веб-документів та користувацьких інтерфейсів, написаних мовами HTML або XHTML. CSS мова може бути застосована до будь-якого XML документа, включаючи XHTML, SVG, XUL,

RSS, та ін. Разом з HTML та JavaScript, CSS є технологією, яка використовується багатьма веб-сайтами для створення візуально привабливих сторінок, користувацьких інтерфейсів для веб-застосунків, а також графічних користувацьких інтерфейсів (CSS) для багатьох мобільних застосунків (таких як Firefox OS).

CSS переважно призначений для позначення розділення змісту документа і способу його представлення, надання таких можливостей, як шари або макети, кольори та шрифти тощо. Розділення документу (логічне форматування) покращує доступність документів, забезпечує більшу гнучкість і контроль під час визначення презентаційних можливостей, надає можливості кільком HTML-документам ділитися певним стилем з використанням окремого аркуша стилів у файлі .css, а також зменшити складність і повторюваність коду у структурі документа.

Розділення формату і вмісту дозволяє представити один і той самий документ в різних стилях для різних методів візуалізації. Такі методи візуалізації (rendering methods) передбачають: відображення на екрані, відтворення у друкованому вигляді, відтворення голосом за допомогою голосового браузера або зчитувача екрана, а також застосування сенсорних пристроїв (система Брайля). Веб-сторінка також може відображатися по-різному залежно від розміру екрана або типу пристрою. Користувач може вказати інший аркуш стилю, наприклад аркуш стилю CSS, збережений на його комп'ютері, щоб перевизначити стиль дизайнера, який визначений для даної сторінки.

Специфікація CSS описує схему пріоритету для визначення того, які правила стилю застосовуються, якщо застосовано більше ніж одне правило, що відповідає одному певному елементу. Ці правила застосовуються за системою, що називається каскадом (*cascading*), так що пріоритети обчислюються та привласнюються правилам, тому результати візуалізації є передбачуваними.

6.2. Створення каскадних стилів

Каскадна таблиця стилів – це збірка одного або декількох каскадних кодів стилів, що була видобута з заголовка веб-сторінки і зберігається в окремому файлі в спеціальному (CSS) форматі. За домовленістю, файли каскадних таблиць стилів завжди отримують розширення *.css*, щоб веб-розробники могли відразу розпізнати їх як каскадні таблиці стилів (часто просто називають CSS файли або таблиці стилів). Існує дві основні причини використання таблиць стилів: спрощення заголовної частини HTML-документів та збереження всіх визначень стилю в одному місці, де їх можна легко знайти та модифікувати.

Компанії веб-розробки, як правило, після створення веб-сайту, можуть продавати його різним клієнтам. В цьому випадку CSS дозволяє налаштувати зовнішній вигляд і поведінку веб-сайту для кожного клієнта лише шляхом зміни значень в таблиці стилю. Це хороший спосіб для компаній максимізувати свій прибуток, мінімізуючи зусилля, необхідні для підтримки декількох клієнтів.

В наведеному покроковому **завданні 5.1**. описано створення файлу CSS. Необхідно змінити код, який знаходиться між тегами <head> і </head>. Після цього створюється файл CSS за допомогою текстового редактора. Під час збереження файлу CSS необхідно додати розширення файлу *.css*.

1. Відкрийте текстовий редактор.
2. Відкрийте існуючий HTML файл (eighteen. html).
3. За допомогою команди Save As збережіть його під іншим ім’ям (nineteen.html) в тій самій папці.
4. Модифікуйте частину <head>…</head> нового файлу так як це показано нижче.

```
<head>
<title>HTML and JavaScript</title>
<style type="text/css">
h1 {color:red}
h2 {color:blue}
h3 {color:purple}
p {color:black}
</style>
<link href="nineteen.css" rel="stylesheet"
type="text/css"></link>
</head>
```

В результаті повинні отримати наступний код

FIGURE 5–2 nineteen.html file header

```
<head>
<title>HTML and JavaScript</title>
<link href="nineteen.css" rel="stylesheet" type="text/css"></link>
</head>
```

5. Збережіть зміни.

6. Створіть новий текстовий документ.

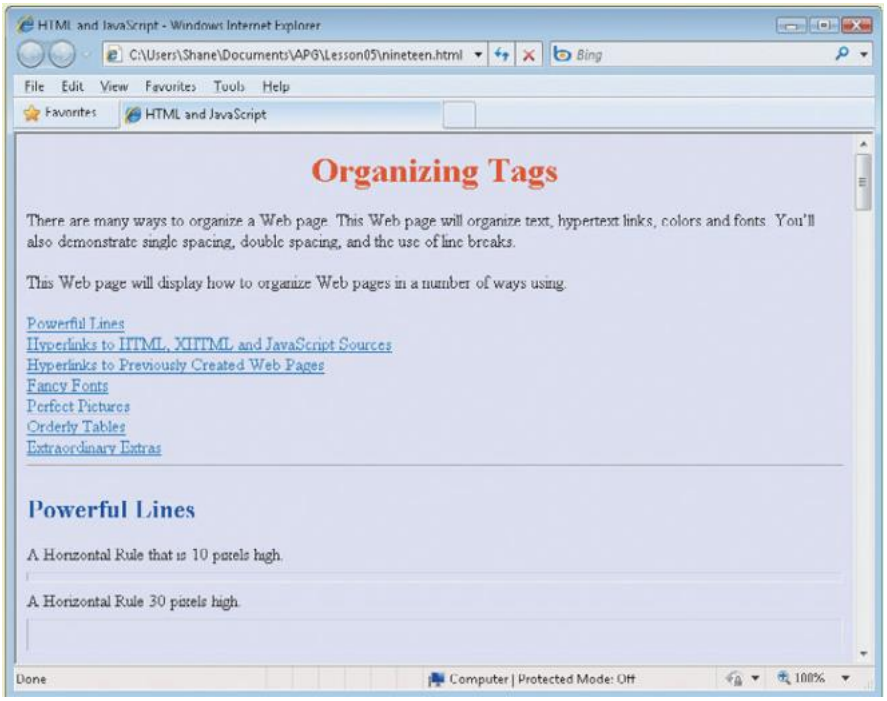
7. Додайте текст CSS, наведений на рисунку нижче (Figure 5-3).

FIGURE 5–3 Your first Cascading Style Sheet (CSS file)

```
h1 {color:red}
h2 {color:blue}
h3 {color:purple}
p {color:black}
```

8. Збережіть новий файл з іменем **nineteen.css** в тій самій папці, що й попередні.

9. Відкрийте файл HTML (**nineteen.html)** у браузері. Результат повинен мати вигляд зображений на рисунку нижче (**Figure 5–4)**.



6.3. Стилі гіперпосилань

Вигляд і поведінка гіперпосилань можуть бути змінені за допомогою каскадних таблиць стилів. За замовчуванням браузер Internet Explorer, як і багато інших популярних браузерів, відображає текстові гіперпосилання підкресленим синім шрифтом. Файл веб-сторінки navbar.html, який ви переглянули у урок 4 (і показано на рисунку 5–5) є гарним прикладом формату текстових посилань, який типово використовується шрифтом та стилем.

Якщо ви переведете вказівник миші на будь-який з 18 гіперпосилань (наведених на рисунку нижче), вказівник змінює форму на зображення руки, але саме текстове посилання не змінює свій вигляд. Посилання як і раніше зображується підкресленим синім шрифтом. Крім того, якщо ви вкажете на посилання (наприклад, вісімнадцять.html), а потім натисніть і утримуйте ліву кнопку миші, текстове посилання все одно відображатиметься як підкресленим синім шрифттом; це не змінюється. Однак, після того, як ви відпустите кнопку миші, тим самим клацнувши посилання, ви відкриєте пов'язану веб-сторінку в браузері. Коли ви повернетеся до сторінки з посиланнями, шрифт тексту посилання буде змінено на підкреслений фіолетового кольору, як показано на рис. 5–6. Ця зміна зовнішнього вигляду вказує на те, що посилання було відвідано.

FIGURE 5–5 Navigation bar with default style links

FIGURE 5–6 Navigation bar with default visited link

Описаний типовий сценарій поведінки гіперпосилання ілюструє чотири можливі стани посилання: нормальний, наведення, активний та відвідний.

Нормальним станом гіперпосилання є те, як він з'являється, коли вказівник миші не знаходиться над ним і його ніколи ще не було активовано.

Стан *наведення* визначає вигляд посилання, коли вказівник миші знаходиться над ним, але його ще не було натиснуто.

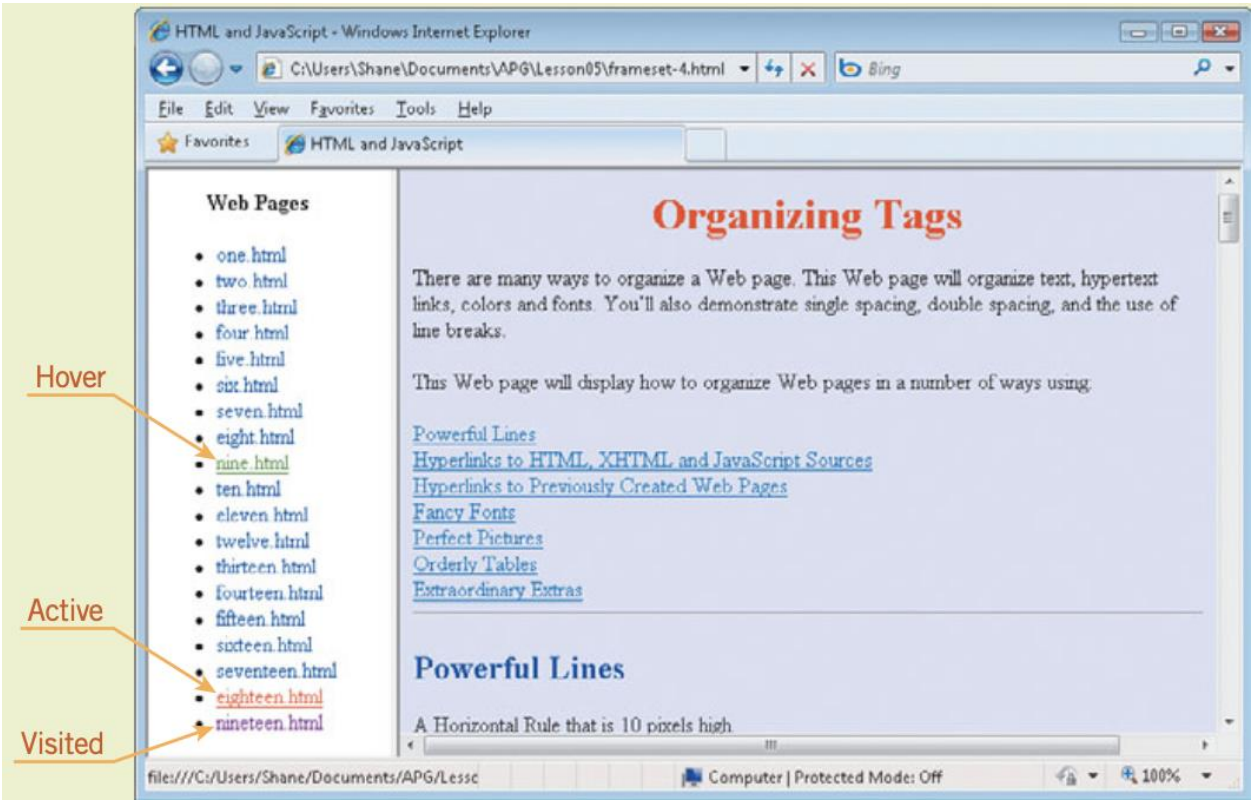
Активний стан посилання полягає в тому, що посилання фокусується на веб-сторінці під час натискання кнопки миші (проміжний стан, який користувач рідко помічаю).

Відвідуваний стан визначає зовнішній вигляд посилання після того, як його було активовано (натиснуто) принаймні хоча б один раз.

Описані стани гіперпосилань можуть бути визначені наступним кодом CSS:

```
a:link {color:blue;text-decoration:none}
a:visited {color:purple;text-decoration:none}
a:hover {color:green;text-decoration:underline}
a:active {color:red;text-decoration:underline}
```

За застосування цього коду отримуємо наступний результат:



6.4. Застосування каскадних стилів

Каскадні стилі дуже корисні для застосування декількох атрибутів до багатьох різних елементів відображення, без необхідності багаторазового написання цих атрибутів у тегах кожного з елементів. Проте, такий підхід не завжди є доцільним. Наприклад, веб-сторінка складається з кількох розділів (або підрозділи) тексту, необхідно застосувати різний набір атрибутів до кожного з підрозділів. Очевидно, що застосування єдиного стилю до всієї сторінки не дасть бажаних результатів. Технологія CSS дозволяє визначити класи стилів, які можуть бути застосовані до певних тегів всередині однієї веб-сторінки, не впливаючи на інші теги одного типу.

Клас стилів – це група атрибутів, яким привласнена певна назва, і які визначаються каскадною таблицею стилів та можуть бути застосовані до визначених тегів в HTML-документі.

Приклад коду в HTML-документі:

```
<html>

<head>
<title>HTML and JavaScript</title>
<link href="welcome-2.css" rel="stylesheet" type="text/css"></link>
</head>

<body>
<div class="text1">Welcome</div>
<br>
<div class="text2">to</div>
<br>
<div class="text3">HTML and JavaScript</div>
</body>

</html>
```

Приклад коду в CSS-документі:

```
.text1 {color:red;font-size:32;text-align:center}

.text2 {color:green;font-size:24;text-align:center}
.text3 {color:blue;font-size:32;text-align:center}
```

6.5. Sharing Style Classes

Як було продемонстровано в попередньому параграфі, одна веб-сторінка може використовувати кілька класів стилів. Однак, також можливо, що один клас стилю може бути використаний декількома веб-сторінками. Все, що потрібно для використання спільного класу стилів двома або більше веб-сторінками, це додати до цих веб-сторінок посилання на відповідні таблиці стилів, в яких визначається клас стилю. Це часто робиться, коли веб-сторінки, що мають спільний доступ до класу, повинні відображатися в одному дизайнерському стилі (належать до однієї компанії). Однак це не є обов’язковою вимогою. Дві або більше веб-сторінки, які зовсім не пов'язані одна з одною, туж можуть мати один і той самий визначений клас стилів, поки вони мають доступ до відповідного файлу CSS.

У наведеному нижче прикладі створюється спільна таблиця стилю, яка пов’язана з двома різними веб-сторінками в рамках спільного дизайнерського стилю. Цей файл CSS містить клас стилів, який надає веб-сторінкам загальний блідо-жовтий колір фону.

Приклад коду HTML документу:

FIGURE 5–19 welcome-3.html file

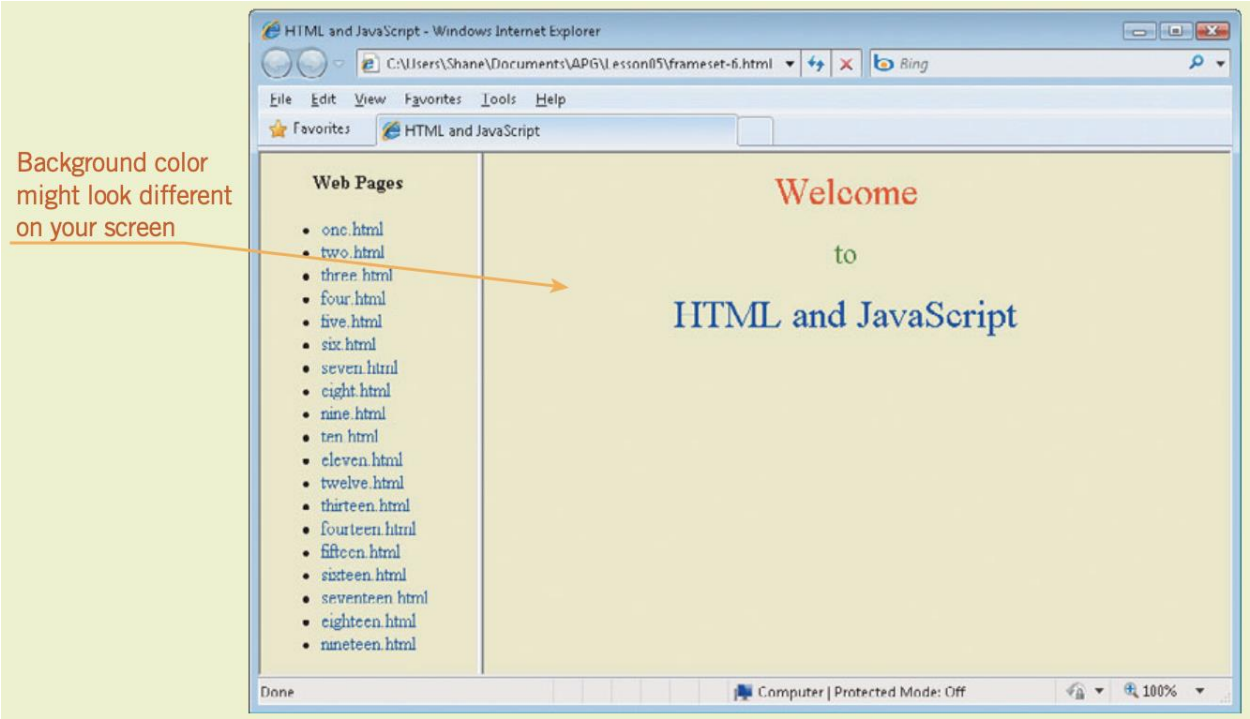
```
<html>

<head>
<title>HTML and JavaScript</title>
<link href="welcome-2.css" rel="stylesheet" type="text/css"></link>
<link href="common.css" rel="stylesheet" type="text/css"></link>
</head>

<body class="bgcolor1">
<div class="text1">Welcome</div>
<br>
<div class="text2">to</div>
<br>
<div class="text3">HTML and JavaScript</div>
</body>

</html>
```

Приклад коду CSS документу:

FIGURE 5–20 common.css file

```
.bgcolor1 {background-color:#EFEFCF}
```

Результат виконання кодів має наступний вигляд:



6.6. Використання CSS для оформлення таблиць

До цих пір весь досвід, який ви отримали завдяки стилям HTML, певною мірою пов’язаний з підтасуванням тексту. Всі стилі, каскадні стилі і класи стилів, які були створені у наведених прикладах, пов'язані з такими елементами, як колір шрифту, колір фону, вирівнювання тексту і так далі. Але стилі можуть бути використані для управління всіма видами атрибутів тегів HTML, а не тільки тими, що стосуються тексту.

Розглянемо, наприклад, створення таблиці засобами HTML. Таблиці підтримують різноманітні атрибути, які використовуються для визначення ширини стовпців, товщини границь і стилю, вирівнювання текста всередині комірок, а також інші властивості таблиці. Оскільки веб-дизайнери хочуть, щоб різні таблиці на їхніх веб-сторінках мали різні атрибути, це чудова можливість використовувати класи стилів.

Приклад коду HTML документу:

```
<html>

<head>
<title>HTML and JavaScript</title>
<link href="twenty.css" rel="stylesheet" type="text/css"></link>
</head>

<body style="background-color:#E6E6FA">
```

```
<h2><a name="tables">Orderly Tables</a></h2>
<table class="table1">
```

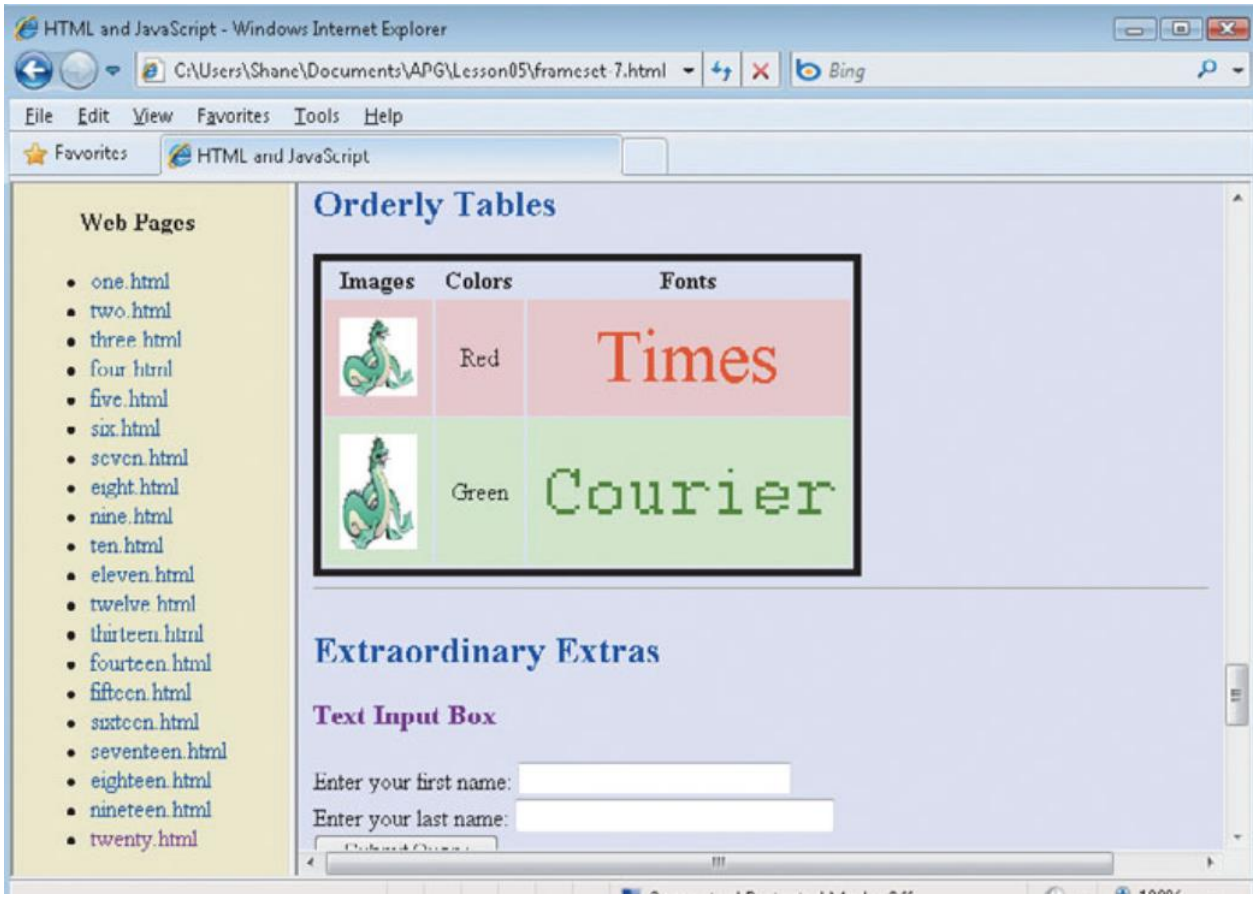
```
  <tr>
    <th>Images</th>
    <th>Colors</th>
    <th>Fonts</th>
  </tr>
  <tr>
    <td class="row1"><img src="levy.gif" height="50" width="50"></td>
    <td class="row1">Red</td>
    <td class="row1"><font face="times" size="7" color="red">Times</td>
  </tr>
  <tr>
    <td class="row2"><img src="levy.gif" height="75" width="50"></td>
    <td class="row2">Green</td>
    <td class="row2"><font face="courier" size="10"
color="green">Courier</td>
  </tr>
</table>
<hr />
```

Приклад коду CSS документу:

FIGURE 5–25 twenty.css file

```
h1 {color:red}
h2 {color:blue}
h3 {color:purple}
p {color:black}

.table1 {border-color:black;border-style:solid;border-width:5}

.row1 {background-color:#EFCFCF;padding:10;text-align:center}
.row2 {background-color:#CFEFCF;padding:10;text-align:center}
```

Результат виконання кодів має наступний вигляд:



6.7. Застосування CSS до HTML

Розглянемо три методи застосування CSS до документа: з зовнішньою таблицею стилів, з внутрішньою таблицею стилів, і з вбудованими стилями.

Зовнішні таблиці стилів

Зовнішня таблиця стилів містить CSS в окремому файлі з розширенням *.css*. Це найпоширеніший та найефективніший метод застосування CSS до документа HTML. Можна пов'язати один CSS-файл з декількома веб-сторінками, для того щоб всі вони використовували одні CSS коди.

Приклад посилання на зовнішню таблицю стилів CSS з елемента HTML документу <link>:

```
<!DOCTYPE html>
<html lang="en-GB">
  <head>
    <meta charset="utf-8" />
    <title>My CSS experiment</title>
    <link rel="stylesheet" href="styles.css" />
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>This is my first CSS example</p>
  </body>
</html>
```

Код CSS документа при цьому може мати наступний вигляд:

```
h1 {
  color: blue;
  background-color: yellow;
  border: 1px solid black;
}

p {
  color: red;
}
```

Атрибут *href* елемента <link> має бути використано для посилання на файл у певній файловій системі. У прикладі, наведеному вище, файл CSS знаходиться в тій самій теці, що і HTML-документ, але його можна розмістити в іншому місці файлової системи та вказати шлях. Наведемо три приклади:

```
<!-- Inside a subdirectory called styles inside the current
directory -->
<link rel="stylesheet" href="styles/style.css" />

<!-- Inside a subdirectory called general, which is in a
subdirectory called styles, inside the current directory -->
<link rel="stylesheet" href="styles/general/style.css" />

<!-- Go up one directory level, then inside a subdirectory
called styles -->
<link rel="stylesheet" href="../styles/style.css" />
```

Внутрішні таблиці стилів

Внутрішня таблиця стилів знаходиться безпосередньо в коді HTML-документа. Для створення внутрішньої таблиці стилів, код CSS знаходиться всередині елемента <*style*>, що міститься всередині тега <head>.

Код HTML документа за використання внутрішнього визначення стилю має наступний вигляд:

```
<!DOCTYPE html>
<html lang="en-GB">
  <head>
    <meta charset="utf-8" />
    <title>My CSS experiment</title>
    <style>
      h1 {
        color: blue;
        background-color: yellow;
        border: 1px solid black;
      }

      p {
        color: red;
      }
    </style>
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>This is my first CSS example</p>
  </body>
</html>
```

У деяких обставинах застосування внутрішніх стилів можуть бути ефективними. Наприклад, якщо, ви працюєте з системою управління вмістом, за якої блокуються зміни від зовнішніх CSS файлів.

Проте для сайтів з більш ніж однією сторінкою застосування внутрішніх стилів стає менш ефективним способом роботи. Для застосування уніфікованого стилю CSS на декількох сторінках за допомогою внутрішніх таблиць стилів необхідно додавати внутрішню таблицю стилів на кожній веб-сторінці, яка використовуватиме цей стиль. Також до технічного обслуговування сайту застосовується штраф за ефективність. За застосування

внутрішніх стилів існує ризик, що навіть одна проста зміна стилю може вимагати зміни в кількох веб-сторінках одночасно.

Вбудовані стилі

Вбудовані стилі – це декларації CSS, які впливають на один HTML-елемент, що міститься в атрибуті *style*. Реалізація вбудованого стилю в HTML-документі може мати наступний вигляд:

```
<!DOCTYPE html>
<html lang="en-GB">
  <head>
    <meta charset="utf-8" />
    <title>My CSS experiment</title>
  </head>
  <body>
    <h1 style="color: blue;background-color: yellow;border:
1px solid black;">
      Hello World!
    </h1>
    <p style="color:red;">This is my first CSS example</p>
  </body>
</html>
```

Слід уникайти використання такого способу використання CSS, якщо це можливо. Такий спосіб має низку недоліків. По-перше, це найменш ефективна реалізація CSS для технічного обслуговування. Одна зміна стилю може потребувати декількох змін в межах однієї веб-сторінки. По-друге, вбудовані стилі CSS також змішує (CSS) презентаційний код з HTML кодом і контентом, утворюючи код складний для читання і розуміння. Відокремлення коду та вмісту полегшує обслуговування для всіх, хто працює на веб-сайті.

Хоча існує кілька випадків, за яких використання вбудованих стилів є припустимим. Можливо, вам доведеться скористатися вбудовуваним стилем, якщо ваше робоче середовище є дуже обмежене в ресурсах. Наприклад, можливо, ваші CMS дозволяють редагувати лише HTML-тіло. Крім того, ви

можете побачити багато вбудованих стилів в HTML-листах, для досягнення сумісності з якомога більшою кількістю клієнтів електронної пошти.

6.8. Селектори

Селектори призначено для застосування стилів до вмісту. Якщо CSS не застосовується до вмісту, як очікувалося, ваш селектор може не відповідати тому, як ви вважаєте за потрібне застосовувати стиль.

Кожне правило CSS починається з селектора (або списку селекторів) для того, щоб повідомити браузеру, до якого елемента або елементів повинні застосовуватися правила. Всі наведені нижче приклади є робочими селекторами або списками селекторів.

```
h1
a:link
.manythings
#onething
*
.box p
.box p:first-child
h1, h2, .intro
```

CSS-*селектор* є першою частиною правила CSS. Це шаблон елементів та інших термінів, які визначають для браузера, які елементи HTML повинні бути обрані, щоб мати значення властивостей CSS, що розташовані всередині правила, яке застосовується до них. Елемент або елементи, які вибираються селектором, називають *предметом* селектора.

```
h1 {
  color: blue;
  background-color: yellow;
}

p {
  color: red;
}
```

У CSS селектори визначаються в специфікації CSS Selectors; як і будь-яка інша частина CSS вони повинні підтримуватися браузерами для їх роботи. Більшість селекторів, з якими ви зіткнетесь, визначені в специфікації селекторів рівня 3.

Списки селектора

Якщо є більше ніж один елемент, який використовує один CSS, то окремі селектори можуть бути об'єднані в список селекторів так, щоб правило визначення стилю було застосовано до всіх окремих селекторів. Наприклад, якщо є CSS для h1, а також для класф .special, можна записати це як два окремих правила.

```
h1 {
  color: blue;
}

.special {
  color: blue;
}
```

Але можна об'єднати їх у список селектора, додавши кому між ними.

```
h1,
.special {
  color: blue;
}
```

Встановлення пробілу є допустимим до або після коми. Крім того, ви можете знайти селектори, які є більш читабельними, якщо кожен з них знаходиться в окремому рядку.

```
h1,
.special {
  color: blue;
}
```

У наведеному нижче прикладі спробуйте об'єднати два селектори, які мають однакові декларації. Візуалізація має бути однаковою до та після їх об'єднання.

Type selectors

Veggies es bonus vobis, proinde vos postulo essum magis kohlrabi welsh onion daikon amaranth tatsoi tomatillo melon azuki bean garlic.

Gumbo beet greens corn soko **endive** gumbo gourd. Parsley shallot courgette tatsoi pea sprouts fava bean collard greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea peanut soko zucchini.

Turnip greens yarrow ricebean rutabaga *endive cauliflower* sea lettuce kohlrabi amaranth water spinach avocado daikon napa cabbage asparagus winter purslane kale. Celery potato scallion desert raisin horseradish spinach

```
span {
    background-color: yellow;
}

strong {
    color: rebeccapurple;
}

em {
    color: rebeccapurple;
}
```

```
<h1>Type selectors</h1>
<p>Veggies es bonus vobis, proinde vos postulo essum magis
<span>kohlrabi welsh onion</span> daikon amaranth tatsoi tomatillo
    melon azuki bean garlic.</p>

<p>Gumbo beet greens corn soko <strong>endive</strong> gumbo
gourd. Parsley shallot courgette tatsoi pea sprouts fava bean
collard
    greens dandelion okra wakame tomato. Dandelion cucumber
earthnut pea peanut soko zucchini.</p>

<p>Turnip greens yarrow ricebean rutabaga <em>endive
cauliflower</em> sea lettuce kohlrabi amaranth water spinach
avocado
    daikon napa cabbage asparagus winter purslane kale. Celery
potato scallion desert raisin horseradish spinach
```

Якщо селектори сгруповані у такий спосіб, що будь-який селектор синтаксично не припустимий, то все правило буде проігноровано.

У наступному прикладі, некоректне правило вибору класу буде проігноровано, тоді як h1 все одно буде стилізовано.

```
h1 {
  color: blue;
}


..special {
  color: blue;
}
```

Однак, за згрупування, ані h1, ані клас не будуть оформлені належним чином, оскільки все правило вважається недійсним.

```
h1, ..special {
  color: blue;
}
```

Види селекторів

Існує декілька різних видів селекторів, і знання того, який тип селектора вам може знадобитися, допоможе обрати інструмент для роботи.

Селектори типів, класів і ідентифікаторів

До цієї групи належать селектори, які націлені такі на HTML-елементи як, наприклад, <h1>.

```
h1 {
}
```

До цієї групи належать також селектори, які націлені на класи:

```
.box {
}
```

а також ідентифікатори ID:

```
#unique {
}
```

Атрибути селекторів

Ця група селекторів надає різні способи вибору елементів, заснованих на наявності в елемента певного атрибуту:

```
a[title] {
}
```

Такі селектори дозволяють також зробити вибір, заснований на наявності атрибута з певним значенням:

```
a[href="https://example.com"]
{
}
```

6.9. Властивості

Ви можете зіткнутися з ситуаціями, коли два селектори обирають один і той самий HTML елемент. Розглянемо таблицю стилів утворену за допомогою селектора p, який встановлює синй колір для тексту абзацу. Однак існує також клас, який задає текст вибраних елементів як червоний.

```
.special {
  color: red;
}

p {
  color: blue;
}
```

Припустимо, що в нашому HTML-документі ми маємо абзац з класом *special*. Діють обидва правила. Який селектор переважає? Текст параграфа якого кольору синього або червоного ми маємо очікувати?

```
<p class="special">What color am I?</p>
```

У мові CSS є правила для керування тим, який селектор сильніший у разі конфлікту. Ці правила називаються каскадними або специфічними. У прикладі коду нижче визначено два правила для селектора *p*, але текст параграфа буде синім. Це відбувається тому, що декларація, яка встановлює текст абзацу синім, з'являється пізніше в таблиці стилів. Пізніші стилі замінюють конфліктні стилі, які з'являються раніше в таблиці стилів. Це правило каскаду.

```
p {
  color: red;
}

p {
  color: blue;
}
```

Однак, у випадку нашого більш раннього прикладу з конфліктом між селектором класів і селектором елементів, клас переважає, надаючи абзацу червоний колір тексту. Хоча конфлікт стилів виникає пізніше ніж визначення кольору червоним, проте клас оцінюється як більш специфічний (має вищий пріоритет), ніж селектор елементів, тому скасовує інші суперечливі оголошення стилю.

Універсальний селектор

Зірочка (*) – універсальний селектор

```
*[lang^=en]{color:green;}
*.warning {color:red;}
*#maincontent {border: 1px solid blue;}
```

```
<p class="warning">
  <span lang="en-us">Зелёный span</span> в красном параграфе.
</p>
<p id="maincontent" lang="en-gb">
  <span class="warning">Красный span</span> в зелёном параграфе.
</p>
```

Зелёный span в красном параграфе.

Красный span в зелёном параграфе.

Може використовуватися на просторі імен

- `ns|*` - вхождения всех элементов в пространстве имён ns
- `*|*` - находит все элементы
- `|*` - ищет все элементы без объявленного пространства имён

Приклади селекторів за посиланням

https://html5css.ru/cssref/css_selectors.php

Селектор	Пример	Пример описания
.class	.intro	Выбор всех элементов с class="intro"

#id	#firstname	Выбор элемента с id="firstname"
*	*	Выбор всех элементов
element	p	Выбор всех элементов <p>
element,element	div, p	Выбор всех элементов и всех элементов <p>
element element	div p	Выбор всех <p> элементов внутри элементов
element>element	div > p	Выбор всех элементов <p>, в которых родительский элемент является элементом
element+element	div + p	Выбор всех <p> элементов, помещенных сразу после элементов
element1~element2	p ~ ul	Выбор каждого элемента , которому предшествует элемент <p>
[*attribute*]	[target]	Выбор всех элементов с целевым атрибутом
[*attribute=value*]	[target=_blank]	Выбор всех элементов с target="_blank"

[*attribute*~=*value*]	[title~=flower]	Выделяет все элементы с атрибутом title, содержащим слово "flower"
[*attribute*|=*value*]	[lang|=en]	Выбор всех элементов со значением атрибута lang, начиная с "en"
[*attribute*^=*value*]	a[href^="https"]	Выбирает каждый элемент <a>, значение атрибута href которого начинается с "https"
[*attribute*$=*value*]	a[href$=".pdf"]	Выбирает каждый элемент <a>, значение атрибута href которого заканчивается ".pdf"
[*attribute**=*value*]	a[href*="html5css.ru"]	Выбирает каждый элемент <a>, значение атрибута href которого содержит подстроку "html5css.ru"
:active	a:active	Выбор активной ссылки
::after	p::after	Вставка чего-либо после содержимого каждого элемента < p >
::before	p::before	Вставка чего-либо перед содержимым каждого элемента <p>
:checked	input:checked	Выбор каждого отмеченного <input> элемента
:disabled	input:disabled	Выбор всех отключенных элементов <input>

:empty	p:empty	Выбор каждого элемента <p>, не имеющего дочерних элементов (включая текстовые узлы)
:enabled	input:enabled	Выбор всех включенных элементов <input>
:first-child	p:first-child	Выбирает каждый элемент <p>, являющийся первым дочерним элементом родительского элемента
::first-letter	p::first-letter	Выбор первой буквы каждого элемента <p>
::first-line	p::first-line	Выбор первой строки каждого элемента <p>
:first-of-type	p:first-of-type	Выбор каждого элемента <p>, являющегося первым элементом <p> родительского элемента
:focus	input:focus	Выбор элемента ввода, имеющего фокус
:hover	a:hover	Выбор ссылок при наведении указателя мыши
:in-range	input:in-range	Выбор входных элементов со значением в указанном диапазоне
:invalid	input:invalid	Выбор всех входных элементов с недопустимым значением

:lang(*language*)	p:lang(it)	Выбирает каждый элемент <p> с атрибутом lang, равным "it" (Итальянский)
:last-child	p:last-child	Выбирает каждый элемент <p>, являющийся последним дочерним элементом родительского элемента
:last-of-type	p:last-of-type	Выбирает каждый элемент <p>, являющийся последним <p> элементом его родительского элемента
:link	a:link	Выбор всех непосещаемых ссылок
:not(*selector*)	:not(p)	Выбор каждого элемента, не являющегося элементом <p>
:nth-child(*n*)	p:nth-child(2)	Выбирает каждый элемент <p>, являющийся вторым дочерним элементом родительского элемента
:nth-last-child(*n*)	p:nth-last-child(2)	Выбирает каждый элемент <p>, являющийся вторым дочерним элементом родительского объекта, считая от последнего дочернего элемента
:nth-last-of-type(*n*)	p:nth-last-of-type(2)	Выбирает каждый элемент <p>, являющийся вторым элементом <p> родительского элемента, считая от последнего дочернего элемента

:nth-of-type(*n*)	p:nth-of-type(2)	Выбор каждого элемента <p>, являющегося вторым элементом <p> родительского элемента
:only-of-type	p:only-of-type	Выбор каждого элемента <p>, являющегося единственным элементом <p> родительского элемента
:only-child	p:only-child	Выбирает каждый элемент <p>, являющийся единственным дочерним элементом родительского элемента
:optional	input:optional	Выбор входных элементов без атрибута "обязательный"
:out-of-range	input:out-of-range	Выбор входных элементов со значением за пределами заданного диапазона
:read-only	input:read-only	Выбор входных элементов с указанным атрибутом "ReadOnly"
:read-write	input:read-write	Выбор входных элементов с атрибутом "ReadOnly" не указан
:required	input:required	Выбор входных элементов с указанным атрибутом "required"
:root	:root	Выбор корневого элемента документа

::selection	::selection	Выбор части элемента, выбранной пользователем
:target	#news:target	Выбор текущего активного элемента #news (щелчок по URL-адресу, содержащему это имя привязки)
:valid	input:valid	Выбор всех входных элементов с допустимым значением
:visited	a:visited	Выбор всех посещенных ссылок