МОДУЛЬ І. Мова HTML 4.01. Каскадні таблиці стилів CSS

Лабораторна робота 1.1.

Структура HTML-документа. Форматування тексту веб-сторінки. Гіпертекстові посилання. Використання глобальних і локальних посилань в HTML-документах. Створення навігаційних карт

Мета: ознайомитися з мовою HTML і створенням найпростішого документа; набуття навичок форматування веб-сторінки; ознайомитися з принципами створення гіперпосилань, способами додавання графіки на веб-сторінку та технологією створення навігаційних карт.

Вимоги до обладнання та програмного забезпечення

Лабораторна робота виконується на ПК з використанням програми Microsoft Expression Web 4 та браузерів Google Chrom, Opera, Mazila Firefox.

Основні теоретичні відомості

Елементи розмітки бувають двох типів - одиночні й парні (теги). Одиночний тег використовується самостійно, а парний може включати в себе інші теги або текст. Елемент розмітки складається з трьох компонентів: початкового елемента розмітки, вмісту і кінцевого елемента. У деяких елементах може не бути кінцевого елемента розмітки.

Всі елементи можна умовно розбити на декілька категорій:

- Структурні, які задають структуру документа (наприклад, *html, head, body і title*);
- блокові елементи, призначені для форматування цілих текстових блоків (наприклад, *div, h1, p, pre*);
- текстові елементи, які задають розмітку тексту *(em, dfn,code, samp)*;
- спеціальні елементи порожнього рядка *(br, hr, nobr)*, якірний елемент *a*, вбудовані елементи *(embed, uimg, map)*, елементи форм *(input select)*, елементи таблиць (*table*) та інші.

Часто елементи розмітки містять додаткові елементи – атрибути. Допустимі різні атрибути, які розділяються між собою пробілом. Втім, є теги без будь яких додаткових атрибутів. Умовно атрибути можна розділити на обов'язкові, неодмінно повинні бути, і необов'язкові (їх додавання залежить від мети застосування тега). Атрибут записується після імені елемента розмітки перед закриваючою дужкою і складається, як правило, з пари «ім'я атрибут та – значення».

Правила сумісності описує стандарт XHTML*:*

- всі імена елементів розмітки і атрибутів пишуться маленькими літерами;
- всі елементи розмітки обов'язково мають закриватися: парні – звичайним способом, непарні: <... /> (пробіл перед косою рискою обов'язковий з міркувань оберненої сумісності із старими браузерами);
- значення атрибутів завжди існують і завжди записуються в лапках;
- використовувати застарілі і специфічні для конкретного браузера елементи не можливе;
- можливе оформлення виноситься у таблиці стилів.

HTML-документ складається з трьох частин:

1. оголошення типу документа,
2. заголовка документа,
3. тіла документа.

Приклад 1.1:

```
<!DOCTYPE HTML >
<html>
<head>
    <title>Задаємо назву документа</title>
</head>
<body>
Основна частина документа
</body>
</html>
```

Елементи, що відносяться до заголовка документа:

<head> – містить технічну інформацію про сторінку: заголовок, опис, ключові слова для пошукових машин, кодування і т.д.;

<title> – текст, розміщений всередині цього тега, відображається в рядку заголовка веб-браузера. Довжина заголовка повинна бути не більше 60 символів, щоб повністю поміститися в заголовку.

<style> – використовується для вставлення в документ таблиці стилів CSS. Таких елементів на сторінці може бути декілька;

<link> – описує взаємозв'язок документа з іншими документами на сайті, вказуючи його місце в ієрархічній структурі сайта. Елемент не має кінцевого елемента розмітки. У заголовку може містити кілька елементів *<link>*;

<meta> – технічна інформація про документ. З його допомогою можна задати опис вмісту сторінки і ключові слова для пошукових машин, автора HTML-документа й інші властивості метаданих. Елемент не має кінцевого елемента розмітки;

<script> – дозволяє приєднувати до документа різні сценарії. Тег обов'язково закривається, при цьому текст сценарію може розташовуватися або всередині цього елемента, або в зовнішньому файлі. Якщо текст сценарію розташований у зовнішньому файлі, то він підключається за допомогою атрибутів елемента;

елемент *<body>* призначений для розміщення даних основної частини веб-сторінки. Вона включає блокові і текстові елементи. Блокові елементи відносяться до частин тексту абзацу. Текстові елементи описують властивості окремих фраз і ще більш дрібних частин тексту.

Правила вкладення елементів:

- елементи не повинні перетинатися. Іншими словами, якщо відкриваючий елемент розташовується всередині елемента, то і відповідний закриваючий елемент повинен розташовуватися в межах цього самого елемента;
- блокові елементи можуть містити вкладені блокові і текстові елементи;
- текстові елементи можуть містити вкладені текстові елементи;
- текстові елементи не можуть містити вкладені блокові елементи.

Мова HTML допускає два підходи до виділення фрагментів тексту. Розробник може прямо вказати, що шрифт на якійсь ділянці тексту повинен бути жирним або курсивом, тобто змінити фізичний стиль тексту. Або може виділити деякий фрагмент тексту як такий, що має відмінний від звичайного логічний стиль, залишивши інтерпретацію цього стилю браузеру.

При використанні логічних стилів автор документа не може знати заздалегідь, що побачить на екрані читач. Різні браузери тлумачать одні й ті самі мітки логічних стилів по-різному. Деякі браузери ігнорують деякі мітки взагалі і показують нормальний текст замість виділеного логічним стилем.

Логічні елементи наведено в табл. 1.1.

Таблиця 1.1

* ... *	Від англійського *emphasis* – наголос, виділяє текст, за замовчуванням при відображенні на екрані виводить його курсивом
* ... *	Від англійського *strong emphasis* – сильний акцент
<code> ... </code>	Призначений для відображення однієї або декількох рядків тексту, який являє собою програмний код
<samp> ... </samp>	Використовується для відображення тексту, який є результатом виведення комп'ютерної програми або скрипта
<kbd> ... </kbd>	Від англійського *keyboard* – клавіатура. Виділяє текст, що вводиться з клавіатури
<var> ... </var>	Від англійського *variable* – змінна. Використовується для виділення змінних комп'ютерних програм

Елемент *<font>*, хоча і скасований, залишається досить популярним. Цей елемент керує зовнішнім виглядом шрифту і повинен містити хоча б один з трьох атрибутів:

- *face* – вказує гарнітуру шрифту. Наприклад, *face="arial"*. Якщо такий шрифт не встановлено на комп'ютері, то через кому вказують кілька шрифтів у порядку переваги;
- *size* – встановлює розмір шрифту у власних умовних одиницях від 1 до 7 (найбільший). Значення за замовчанням – 3 (близько 12 пунктів);
- *color* – вказує колір шрифту в системі RGB.

Під фізичним елементом прийнято розуміти пряму вказівку браузеру на модифікацію поточного шрифту. Наприклад, все, що розташовується між елементами ** і **, буде написано жирним шрифтом. Текст між елементами ** і ** буде написаний курсивним шрифтом. Дещо окремо стоїть елемент розмітки *<tt>*.

Фізичні елементи наведено в табл. 1.2.

Таблиця 1.2

**	жирний шрифт
**	курсив
<tt>	шрифт фіксованої ширини (як на друкарській машинці)

<u>	підкреслений текст
<strike>	закреслений шрифт
<s>	теж закреслений шрифт
<big>	шрифт більшого розміру
<small>	шрифт меншого розміру

Шрифти можуть комбінуватися. Наприклад, використовуючи такий текст **приклад**, ми отримаємо комбінацію жирного і курсивного шрифтів.

Блочні елементи використовують для розбивання тексту документа на блоки. Прикладами текстових блоків є параграфи, абзаци і заголовки. Для відділення однієї частини тексту від іншої також використовують розділові горизонтальні лінії і символи повернення каретки.

Елементи *h1,h2,...h6* використовують для створення заголовків тексту. Їх слід використовувати тільки для виділення заголовків нового розділу або підрозділу. При використанні заголовків потрібно враховувати їх ієрархію, тобто за *<h1>* повинен піти *<h2>* і т.д. Також не допускається вкладення інших тегів у теги *<h1>* ... *<h6>*. Вони мають атрибут *align* – який визначає спосіб вирівнювання заголовка по горизонталі. Можливі значення: *left, right, center, justify*.

Елемент розмітки *p* розбиває текст на окремі абзаци.

Елемент розмітки * * використовують для логічного виділення блоку HTML-документа. Він призначений для виділення фрагмента документа з метою зміни виду вмісту. Як правило, вид блока управляється за допомогою стилів. Розміщуючись між початковим і кінцевим елементами розмітки, текст або HTML-елементи за замовчуванням оформляються як окремий параграф.

Елемент розмітки *<address>* призначений для зберігання інформації про автора або адреси і може включати в себе будь-які елементи HTML на зразок посилань, тексту, виділень і т.д. Оформлення виражається у виділенні рядка адреси курсивом.

Лістинг 1.2:

```
<address>
Ця сторінка розроблена веб-дизайнером Петренком В.І.
</address>
```

Елемент розмітки *<blockquote>* оформляє текст що розташований між початковим і кінцевим елементами як цитату. Використовується для довгих цитат (на відміну від елемента *<cite>*). Цитований текст відображається окремим абзацом зі збільшеним відступом.

Елемент розмітки *
* – здійснює перехід до нового рядка, аналогічний натисканню *Enter* у текстовому редакторі.

Елемент розмітки *<hr>* додає в текст горизонтальну лінію.

Приклад 1.3:

<hr noshade width="90%" color=”red”>

<pre> – виводить текст без форматування зі збереженням пробілів і переносів тексту. Може бути використаний для відображення комп’ютерного коду, повідомлення електронної пошти і т.д. Бажано уникати використання символу горизонтальної табуляції всередині *<pre>*, оскільки він може бути неадекватно інтерпретований деякими браузерами. Замість символу табуляції рекомендується використовувати число пробілів, кратне чотирьом.

Списки в HTML бувають двох видів:

- впорядковані (пронумеровані);
- невпорядковані (не пронумеровані).

Відрізняються вони лише способом оформлення. Перед пунктами невпорядкованих списків зазвичай ставляться маркери (*bullets*), наприклад, крапки, квадрати та інше, тоді як у пунктах впорядкованих списків використовуються номери.

Елемент розмітки *<ul>* створює невпорядкований список. Між початковим і кінцевим елементами повинні бути один або кілька елементів *<li>*, що позначають окремі пункти списку.

Приклад 1.4:

```
<ul>
<li> Системне програмування </li>
<li> Бази даних  </li>
<li> Інтернет-технології </li>
</ul>
```

Елемент розмітки <*ol*> створює впорядкований список. Між початковим і кінцевим елементами повинні бути один або кілька елементів <*li*>, що позначають окремі пункти списку. Має атрибути:

start – визначає перше число, з якого починається нумерація пунктів (лише цілі числа)

type – визначає стиль нумерації пунктів. Може мати значення:

"A" – заголовні букви A, B, C ...

"a" – рядкові букви a, b, c ...

"I" – великі римські числа I, II, III ...

"i" – маленькі римські числа i, ii, iii ...

"1" – арабські числа 1, 2, 3 ...

За замовчуванням <*ol type="1"*>.

Приклад 1.5:

```
<ol type="i" start="2">
<li> Другий пункт впорядкованого списку </li>
<li> Третій пункт впорядкованого списку </li>
<li> Четвертий пункт впорядкованого списку </li>
</ol>
```

Елемент розмітки <*li*> створює пункт у списку. Розташовується усередині елементів <*ol*> або <*ul*>.

Елемент розмітки <*dl*> відкриває і закриває список визначень (термінів та їх описів). Визначення задаються з допомогою елементів <*dt*> та <*dd*>.

Приклад 1.6:

```
<dl>
<dt>Кава</dt><dd>бадьорий гарячий напій</dd>
<dt>Молоко</dt><dd>корисний холодний напій</dd>
</dl>
```

Елемент розмітки <*dt*> створює термін у списку визначень всередині елемента <*dl*>

Елемент розмітки <*dd*> створює визначення терміна всередині елемента <*dl*>.

Гіперпосилання – посилання на інші документи в HTML, які створюються за допомогою елемента розмітки <*a*>. Текст посилання відображається в браузері з підкресленням, колір шрифту – синій, при наведенні на посилання курсор миші змінює вид. Походить від англійського слова *anchor – «якір».*

Елемент розмітки <*a*> застосовується, якщо посиланням планується зробити частину тексту або ціле зображення. Посилання - інструмент, що дозволяє зв'язувати між собою різні документи.

HTML означає «мова маркування гіпертекстів». Гіпертекст, на відміну від звичайного тексту, який можна читати лише від початку до кінця, дозволяє здійснювати миттєвий перехід від однієї частини тексту до іншої. Системи допомоги багатьох популярних програмних продуктів влаштовані саме за гіпертекстовим принципом. Натиснувши лівою кнопкою миші на деякий виділений фрагмент документа, можна перейти до деякого заздалегідь призначеного документа або фрагмента документа.

Елемент розмітки *a* має атрибути:

- *href* – визначає розташований між початковим і кінцевим елементами текст або зображення як гіпертекстове посилання (URL, або лінк) на документ (і/або область документа), зазначений у значенні даного атрибута. Можливі значення:
- http://... – надає доступ до веб-сторінки за протоколом HTTP;
- ftp://... – здійснює запит до FTP-сервера на отримання файла;
- mailto:... – запускає сеанс поштового зв'язку із зазначеним адресатом і хостом. Якщо після адреси поставити знак питання, то можна вказати додаткові атрибути, розділені знаком "*&*";
- *name* – позначає розташовану між початковим та кінцевим елементами область документа як можливий об'єкт для посилання. Як значення потрібно латиною написати будь-яке слово-покажчик, унікальне для даного документа.

  Наприклад: *Розділ1*. Тепер можна посилатися на позначену область простим зазначенням її імені після імені документа. Наприклад, лінк * Розділ1* відправить в розділ "part" файла document.html, а лінк *У кінець документа* – в розділ "*bottom*" поточного документа.
- *target* – визначає вікно (фрейм), на яке вказує гіпертекстове посилання. Цей атрибут використовується лише разом з атрибутом *href*. Як значення потрібно задати ім'я одного з наявних фреймів або одне з таких зарезервованих імен: *_self, _parent, _top, _blank.*

Приклад 1.7:

```
<!-- Використання атрибута name: -->
```

Історія спорту
...
Новини спорту
...
Повернутися до розділуІсторія

Приклад 1.8:

<!-- Використання атрибуту href: -->
FTP-site
Російський проект з OpenGL

Приклад 1.9:

<!-- Створимо посилання для листа -->

Відправити запрошення .
<!-- або просто листа : -->
авторам.

Елемент *a* не може бути вкладеним у собі подібні, тобто неприпустимі конструкції на зразок

Перше посилання
Друге посилання
Продовжуємо перше посилання

Описуються посилання так: *Посилання на документ*. Може містити атрибут *title*, що описує посилання. Якщо користувач наведе курсор миші на посилання, то з'явиться його опис – вміст цього атрибута.

Розрізняють відносні й абсолютні посилання. Приклади відносного посилання:

** – на документ *"index.html"*, який міститься в одному каталозі з поточним документом;

** – те саме;

** – перехід на файл *"index.html"* у підкаталозі *"folder"* поточного каталогу;

** – піднятися вгору на один крок по дереву каталогів, перейти у каталог *"folder"* та перейти на *"index.html"*.

В абсолютних посиланнях вказується повний шлях до файла:

**

Відносні посилання зручніш. Вони не прив'язані до конкретної структури каталогів, тобто можна створити свій сайт на комп'ютері, тестувати і виправляти помилки не підключаючись до Інтернету. І лише коли він буде готовий перенести його на сервер, він працюватиме так само.

Можна робити посилання всередині документа. Це зручно для переходу до різних глав однієї великої статті. Для цього потрібно створити якусь опорну точку – анкер (*anchor* – якір,англ.), на яку і відбуватиметься перехід.

Щоб зробити графічне посилання, слід набрати той самий HTML-код, що і для текстового посилання. Але в тому місці, де був текст, потрібно вставити графічне зображення.

*<img src="dog1.gif"> *

Для розміщення зображення на сторінці слід скористатись елементом розмітки **<***img***>**, обов'язковий атрибут якого *src* визначає адресу графічного файла.

Найбільш популярні графічні формати, які підтримуються усіма сучасними графічними браузерами, це:

- **Формат JPEG** *(Joint Photographic Experts Group)*. Зображення JPEG ідеальні для фотографій, вони можуть містити мільйони різних кольорів. Стискають зображення краще за формат GIF, але текст і великі площі із суцільним кольором можуть покритися плямами.
- **Формат PNG** *(Portable Network Graphics)*. Включає в себе кращі риси GIF- і JPEG-форматів. Містить 64 000 кольорів і дає можливість зробити один із кольорів прозорим, при цьому стискає зображення в менший розмір, ніж GIF-файл.
- **Формат GIF** *(Graphics Interchange Format)*. Ідеальний для стискання зображень, у яких є області із суцільним кольором, наприклад, логотипів. GIF-файли дають змогу встановити один із кольорів прозорим, завдяки чому фон веб-сторінки може проявлятися через частину зображення. Також GIF-файли можуть включати в себе просту анімацію. GIF-зображення містять лише 256 відтінків, тому зображення виглядають плямистими і нереалістичного кольору, як плакати.

Приклад 1.10:

<img src="pic.gif"> – малюнок "pic.gif" знаходиться в тому ж каталозі, що і поточний документ;

<img src="images/pic.gif"> – перейти в папку "images" поточного каталогу і взяти файл звідти;

<img src="../images/pic.gif"> – піднятися вгору на один крок, перейти у каталог "images" і взяти малюнок звідти;

<img src="http://www.sampl.com/edu/ 339/artics/ practic/ myphoto.gif"> – зазначення повного шляху до файлу. Зазвичай застосовується, якщо малюнок розміщується на іншому сервері, або використовується картинка з іншого сайта.

Емент розмітки *<img>* має атрибути:

- *alt="пояснювальний текст"* – альтернативний текст. Текст, який з'являється замість зображень при перегляді сторінки браузером із відключеним автоматичним завантаженням зображень (відключають – для економії часу на повільних лініях зв'язку);
- *border="n"* – рамка навколо малюнка, де *n* – її товщина в пікселях. При *n = "0"* рамка відсутня;
- *width="n"* – задається ширина зображення в пікселях або у відсотках від ширини вікна браузера (тоді після *n* ставлять знак %);
- *height="n"* – висота зображення у пікселях або у відсотках від висоти вікна браузера.

Тег *<map>* служить для подання графічного зображення у вигляді карти з активними областями. Існує два типи зображень-карт:

- клієнтські (*client-side*) – коли користувач клікає по малюнку, браузер сам інтерпретує координати кліка. Він вибирає посилання, визначені для даної області, і переходить по ньому (або виконує задану дію);
- серверні (*server-side*) – координати кліка передаються для інтерпретації на сервер, і вже він робить відповідні дії (наприклад, повертає браузеру URL для переходу).

Перший тип простіший і доступніший, тому далі будемо розглядатимемо лише його.

Для створення зображення-карти використовуються елементи розмітки *<map>* та *<area>*.

Елемент *<map>* повинен обов'язково мати атрибут *name*. Це дає змогу вказати браузеру, до якого саме малюнка на сторінці належить ця карта.

Елемент *<area>* має такі атрибути:

- *shape* - описує форму виділеної області, можливі значення:
  - *rect* – прямокутник;
  - *circle* – коло;
  - *poly* – багатокутник;
  - *default* – визначає всю область.
- *coords* – координати, що визначають розміри та розташування області на зображенні. Кількість і порядок значень залежить від значення атрибута *shape*:
  - *rect*: – лівий-*X*, верхній-*Y*, правий-*X*, нижній-*Y* (тобто спочатку координати лівого верхнього кута, потім правого нижнього);
  - *circle*: – центр-*X*, центр-*Y*, радіус (тобто горизонтальна і вертикальна координати центру кола і радіус);
  - *poly*: – $X_1, Y_1, X_2, Y_2, ..., X_n, Y_n$ (просто перераховуються координати всіх вершин багатокутника).
- *target* – значення цього атрибута (*"_top", "_blank", "_self"* або *"_parent*") визначає, в якому вікні буде відкритий документ.

Приклад 1.11:

```
<!-Створюємо карту з ім'ям ImageMap: -->
<map name="ImageMap">
<area href="something.html" shape="rect" coords="0,0,70,140" alt="Ліва частина">
<area href="anything.html" shape="rect" coords="71,0,140,140" alt="Права частина ">
</map>
<!-Підключаємо зображення -->
<img src="/img/mapa.gif" usemap="#ImageMap" height="140" width="140" border="0">...
```

Завдання до самостійної роботи

1. Створіть HTML-сторінку, в якій буде інформація про ваше улюблене заняття (хобі). При створенні сторінки повинні використовуватися всі досліджені в лабораторній роботі елементи та атрибути (заголовки, абзаци, списки…).
2. Створену сторінку про хобі доповніть малюнками, спробуйте встановити фонове зображення. Використовуйте різні способи розміщення (вирівнювання) малюнків. Також створіть усі види посилань.
3. Створіть навігаційну карту, яка показує розклад занять.

Контрольні запитання та завдання

1. Назвіть категорії елементів, які використовуються для створення веб-сторінок.
2. Назвіть елементи, що відносяться до заголовка документа.
3. Чим відрізняються логічні стилі від фізичних?
4. Які існують види гіперпосилань?
5. Назвіть основні графічні формати.
6. Опишіть призначення навігаційних карт.

Лабораторна робота 1.2.

Елементи створення таблиць HTML. Таблична верстка. Створення форм в HTML-документах.

Мета: Набуття навичок створення таблиць.; ознайоммитися з використанням таблиць для більш ефективного розміщення тексту й графіки на веб-сторінці; навчитися створювати HTML-форми; вивчити основні елементи форм.

Вимоги до обладнання та програмного забезпечення

Лабораторна робота виконується на ПК з використанням програми Microsoft Expression Web 4 та браузерів Google Chrom, Opera, Mazila Firefox.

Основні теоретичні відомості

Таблиці складаються з рядків, стовпчиків та комірок, які можуть містити текст і малюнки. Зазвичай таблиці використовують для впорядкування та представлення даних, однак можливості таблиць цим не обмежуються. За допомогою таблиць зручно верстати макети сторінок, розташувавши потрібним чином фрагменти тексту і зображень.

Для додавання таблиці на веб-сторінку використовується елемент розмітки <*table*>.

Для додавання рядків використовуються елементи <*tr*> і </*tr*>. Щоб розділити рядки на стовпці, застосовуються елементи <*td*> і </*td*>.

Для створення комірок із заголовками застосовуються елементи <*th*> *і* </*th*>.

За замовчуванням таблиця відображається без рамки, а розмітка здійснюється автоматично залежно від обсягу інформації.

Приклад 1.12:

```
<table border="1">
<caption align="right"><strong>Таблиця №1</strong> </caption>
<tr>
<th>Заголовок 1 </th>
<th>Заголовок 2 </th>
<th>Заголовок 3 </th>
</tr>
<tr>
<td>Дані в комірці 1 першого рядка</td>
<td>Дані в комірці 2 першого рядка</td>
<td>Дані в комірці 3 першого рядка</td>
</tr>
<tr>
<td>Дані в комірці 1 другого рядка</td>
<td>Дані в комірці 2 другого рядка</td>
<td>Дані в комірці 3 другого рядка</td>
</tr>
<tr>
<td>Дані в комірці 1 третього рядка</td>
<td>Дані в комірці 2 третього рядка</td>
<td>Дані в комірці 3 третього рядка</td>
</tr>
</table>
```

Основними атрибути таблиць є:

- *align* – визначає спосіб горизонтального вирівнювання таблиці. Можливі значення: *left, center, right*. Значення за замовчуванням – *left;*
- *valign* – повинен визначати спосіб вертикального вирівнювання таблиці. Можливі значення: *top, bottom, middle;*
- *border* – визначає ширину рамки таблиці (у пікселях). При *border="0"* або при відсутності цього атрибута рамка не відображатиметься;
- *cellpadding* – визначає відстань (у пікселях) між рамкою кожної клітинки таблиці і матеріалом що міститься в ній;
- *cellspacing* – визначає відстань (у пікселях) між межами сусідніх комірок;
- *width* – визначає ширину таблиці. Ширина задається або в пікселях, або у відсотковому відношенні до ширини вікна браузера. За замовчуванням цей атрибут визначається автоматично залежно від обсягу матеріалу що міститься в таблиці;
- *height* – визначає висоту таблиці. Висота задається або в пікселях, або у відсотковому відношенні до висоти вікна браузера. За замовчуванням цей атрибут визначається автоматично залежно від обсягу матеріалу що міститься в таблиці;

- *bgcolor* – визначає колір тло комірок таблиці. Задається або RGB-значенням у шістнадцятковій системі, або одним із 16 базових кольорів;
- *background* – дозволяє заповнити тло таблиці малюнком. Як значення потрібно вказати URL малюнка.

Елемент <*caption*> задає заголовок таблиці. Зміст заголовка повиннен складатися тільки з тексту. Використання блокових елементів у цьому випадку неприпустимо.

Межі комірок відображаються лише в тому разі, коли вони мають якийсь зміст. Щоб отримати порожню комірку з межами, досить помістити в неї спеціальний символ , або маленьку прозору gif-картинку.

Приклад 1.13:

```
<table border="1" width="100%" cellpadding="10">
<caption align="right"><b>Приклад таблиці!</b></caption>
<tr>
<th rowspan="3">Об’єднана комірка по вертикалі</td>
<th colspan="3">Об’єднана комірка по горизонталі</td>
</tr>
<tr align="center">
<td>Комірка №1</td>
<td>Комірка №2</td>
<td>Комірка №3</td>
</tr>
<tr align="center">
<td>Комірка №4</td>
<td>Комірка №5</td>
<td>Комірка №6</td>
</tr>
</table>
```

Використання таблиць із невидимою межею відомий спосіб верстки, який застосовують для створення сайтів. Така таблиця фактично являє собою модульну сітку, в якій зручно розміщувати окремі елементи веб-сторінки.

Ширина таблиці, якщо вона явно не вказана, встановлюється браузером автоматично, виходячи з вмісту комірок. Коли таблиця застосовується для створення опорної сітки на сторінці, такий підхід неприйнятний, оскільки залежить від розміру даних. Тому ширину таблиці вказують завжди - у відсотках, якщо використовується «гумовий» макет, або в пікселях для макета фіксованої ширини.

Ширина комірок визначається атрибутом *width* елемента <*td*>. Переваги та недоліки табличної верстки наведено в табл. 1.3.

Таблиця 1.3

Основні переваги табличної верстки	**Недоліки табличної верстки**
Просте створення колонок «Гумова» верстка «Склейка» зображень Однозначне трактування браузерами	тривеле завантаження Громіздкий код Погана індексація пошуковими системами Немає поділу вмісту та оформлення Невідповідність стандартам Неможливість конвертації в мобільну версію

Зразок гумової табличної верстки сайта наведено в прикладі *1.14.*

Приклад 1.14

```
<!DOCTYPE html>
 <head>
 <meta charset="windows-1251" />
 <title>«Гумова» таблична верстка сайта</title>
 </head>
 <body>
 <table cellpadding="0" cellspacing="0" width="90%" align="center" border="1">
 <tr>
 <td width="20" align="center">Логотип</td>
```

```
 <td colspan="2" width="80%" height="60" align="center"><h1>Мій сайт</h1></td>
 </tr>
 <tr>
 <td width=20% height="460" valign="top"><ul>
<li><a href="index.html">Посилання 1</a></li>
<li><a href="index.html">Посилання 2</a></li>
<li><a href="index.html">Посилання 3</a></li>
</ul></td>
 <td width="70%" valign="top">Зміст</td>
 <td valign="top">Посилання</td>
 </tr>
 <tr>
 <td colspan="3" width="100%" align="left">&copy; Всі права захищені</td>
 </tr>
 </table>
</body>
</html>
```

Приклад жорсткої табличної верстки сайта наведено в прикладі *1.15.*

Приклад 1.15

```
<!DOCTYPE html>
 <head>
 <meta http-equiv="Content-Type" content="text/html"; charset="windows-1251" />
 <title>Жорстка таблична верстка сайта </title>
 </head>
 <body>
 <table cellpadding="20" cellspacing="0" width="860" align="center" border="1">
 <tr>
 <td width="30">Логотип</td>
 <td colspan="2" width="100%" height="60" align="center"><h1>Мій сайт</h1></td>
 </tr>
 <tr>
 <td width="30" height="460" valign="top"><ul>
<li><a href="index.html" title="Посилання">Посилання</a></li>
<li><a href="index.html" title="Посилання">Посилання</a></li>
<li><a href="index.html" title="Посилання">Посилання</a></li>
</ul></td>
 <td valign="top">Зміст</td>
 <td width="20" valign="top">Посилання</td>
 </tr>
 <tr>
 <td colspan="3" width="100%" align="center">&copy; Всі права захищені</td>
 </tr>
 </table>
</body>
</html>
```

За допомогою наведених нижче елементів можна створювати анкети, опитувальники і різні поля для введення тексту користувачем із можливістю подальшої відправки заповненої форми на сервер.

Елемент розмітки *<form>* використовується для створення заповнюваної форми. Обов'язково має бути початковий і кінцевий елементи. Усередині елемента *<form>* можна використовувати HTML-елементи. Має атрибути:

- *name* – визначає ім'я форми, унікальне для даного документа. Використовується, якщо в документі є декілька форм;
- *action* – обов'язковий атрибут. Визначає адресу URL, за якою буде відправлено вміст форми – шлях до скрипта сервера, який обслуговує цю форму;
- *method* – визначає спосіб надсилання вмісту форми. Можливі значення *get* (за замовчуванням) та *post*;
- *enctype* – визначає спосіб кодування даних;

- *autocomplete* – Включає автозаповнення полів форми.

Приклад 1.16:

```
<!-- Створюємо форму -->
<form action="/cgi-bin/thanks.pl" method=get name="TestForm">
<!-- Всередині форми створюємо поле введення: -->
Прізвище:
<input type="text" name="lastname" size="20" value="Петренко"><br>
<!-- Кнопка " Відправити ": -->
<input type="submit" value="Відправити">
</form>
```

Під час налагодження скрипта, який приймає дані, частіше використовують метод *get*. Метод *get* не дозволяє передати скрипту великий обсяг даних. Якщо передбачається, що користувач заповнюватиме дуже велику форму або вводитиме об'ємні текстові дані, пересилатиме файл, використовують *method="post"*.

Елемент розмітки *<textarea>* створює поле для введення декількох рядків тексту. У текстовому полі допустимо робити переноси рядків, вони зберігаються при відправці даних на сервер. Елемент *<textarea>* повинен розташовуватися усередині елемента *<form>*. Він має такі атрибути:

- *name* – обов'язковий атрибут. Застосовується для програмування форм;
- *rows* – визначає кількість рядків тексту, видимих на екрані;
- *cols* – визначає ширину текстового поля – у друкованих символах;
- *wrap* – параметри перенесення рядків.

Приклад 1.17:

```
<form action="receive.html" method=post>
<textarea name="address" wrap=virtual" cols="40" rows="3">Ваша адреса...</textarea><br>
<input type="submit" value="ok">
</form>
```

Якщо у формі поле *<textarea>*, передача дані передаються за допомогою методу POST.

Елемент розмітки *<select>* створює в заповнюваній формі меню типу «Вибір одного пункту з багатьох» або «Вибір кількох пунктів із багатьох». Повинен розташовуватися усередині елемента *form* і мати як початковий, так і кінцевий елементами розмітки. Містить кілька елементів *<option>*. Має атрибути:

- *multiple* – дає можливість вибрати декілька пунктів меню при утриманні клавіші *Ctrl*. За замовчуванням можна вибрати лише один пункт меню;
- *name* – визначає ім'я меню, унікальне для форми, яке використовуватиметься при передачі даних на сервер. Кожен виділений пункт меню при передачі на сервер матиме вигляд: *name/value*. Значення (*value*) формується елементом *<option>*;
- *size* – визначає кількість видимих пунктів меню. Якщо значення цього атрибута більше одиниці, то результатом буде список пунктів.

Приклад 1.18:

```
<form action="receive.cgi">
<select name="os" multiple>
<option value="iOS">iOS
<option value="win">MS Windows 10
<option value="unix" selected>UNIX
<option value="linux">Linux
</select>
<input type="submit" value="послати">
</form>
```

Елемент розмітки *<option>* використовується лише з елементом *select*. Елемент *<option>* створює окремі пункти меню. Не має кінцевого тега. Має атрибути:

- *selected* – визначає пункт меню, який буде обрано спочатку при завантаженні документа. Якщо меню має тип "один з багатьох", то командою *selected* може бути помічений лише один з пунктів меню;
- *value* – задає цим пунктом значення, яке буде використано поряд з іншими відомостями про вміст заповненої форми. При наданні інформації на сервер це значення буде об'єднано зі значенням атрибута *name* в елементі *<select>*.

Приклад 1.19:

```
<form action="script.cgi">
```

```
<select name="gender">
<option value="male" selected>Не одружений
<option value="female">Одружений
<option value="not_yet">Визначаюся
</select>
<input type="submit" value="ok">
</form>
```

Елемент розмітки *<input>* створює поле форми (кнопку, поле введення, чекбокс тощо), зміст якого може бути змінено або активізовано користувачем. Елемент не має кінцевого елемента. Елемент *input* повинен розташовуватися усередині елемента *form*.

Має атрибути:

- *name* – визначає ім'я, яке використовуватиметься для доступу до елемента форми, наприклад, у таблицях стилів CSS. Цей атрибут необхідний для більшості типів елемента *<input>*;
- *type* – визначає тип поля для введення даних. За замовчуванням – це *"text"*. Можливі значення:
  - *text* – створює поле введення одного рядка тексту. Зазвичай використовується спільно з атрибутами *size* і *maxlength*;
  - *file* – дає можливість користувачеві додати файл до поточної форми. Можливе використання спільно з атрибутом *accept*;
  - *password* – створює поле введення пароля, текст що вводиться користувачем, відображається у вигляді піктограм "*", щоб приховати введений пароль;
  - *checkbox* – прапорці. Дають змогу вибрати більш одного варіанта із запропонованих. Поле цього типу обов'язково повинно мати атрибути *name* і *value*, а також необов'язковий атрибут *checked*, який вказує на те, що поле активізовано;
  - *radio* – перемикачі. Використовуються, коли потрібно вибрати один варіант із декількох запропонованих. Як і для полів *checkbox*, атрибут *checked* необов'язковий; він може бути використаний для визначення виділеної кнопки у групі кнопок *(radio button);*
  - *submit* – створює кнопку, при натисканні якої заповнена форма надсилається на сервер. Атрибут *value* у даному випадку змінює напис на кнопці, зміст якої задано за замовчуванням залежно від браузера. Якщо атрибут *name* вказаний, то при натисканні цієї кнопки до інформації, що посилається на сервер, додається пара *name/value*, зазначена для атрибута *submit*, інакше пара не додається;
  - *image* – створює графічну кнопку-картинку, що запускає передачу даних на сервер. Місцезнаходження графічного зображення можна задати за допомогою атрибута *src*. При передачі даних серверу повідомляються координати *X* та *Y* тієї точки на зображенні, де було виконано клацання клавішею миші. Координати вимірюються з верхнього лівого кута зображення. При цьому інформація про поле типу *image* записується у вигляді двох пар значень *name/value*. Значення *name* утворюється за допомогою додавання до назви відповідного поля суфіксів "*X*" (абсциси), та "*Y*" (ординати);
  - *reset* – створює кнопку, що очищає значення полів форми до їх початкових значень. При натисканні кнопки дані на сервер не відправляються. Напис на кнопці можна змінити за допомогою атрибута *value*. За замовчуванням напис на кнопці залежить від версії браузера;
  - *hidden* – поля цього типу не відображаються на екрані монітора, що дає змогу розмістити «таємну» інформацію в межах форми. Вміст цього поля надсилається на сервер у вигляді *name/value* разом з іншою інформацією форми. Цей тип полів зручно використовувати для передачі даних від скрипта скрипту непомітно для користувача;
  - *button* – дозволяє створити кнопку у HTML документі, що, при використання мови JavaScript, додає формі функціональності. Атрибут *name* дозволяє задати ім'я цій кнопці, яке може бути використане для будь-якої функції в скрипті. Атрибут *value* дозволяє задати текст, який буде відображений на кнопці в документі;
- *value* – задає текстовий заголовок для полів будь-якого типу, зокрема і кнопок. Для таких полів, як *checkbox* або *radio*, буде повернуто значення, задане в атрибуті *value*;
- *checked* – вказує, що поля типів *checkbox* і/або *radio* активізовані;
- *size* – визначає розмір поля в символах. Наприклад, щоб визначити поле з видимою шириною 24 символи, треба вказати *size*="24";
- *maxlength* – визначає максимальну кількість символів, які можна ввести в текстове поле. Вона може бути більшою, ніж кількість символів, зазначених в атрибуті *size*. За замовчуванням кількість знаків не обмежена;
- *src* – задає URL-адресу зображення, що використовується при створенні графічної кнопки. Використовується спільно з атрибутом *type="image";*

- *align* – визначає спосіб вертикального вирівнювання для зображень. Використовується спільно з атрибутом *type="image"*. Повністю аналогічний атрибуту *align* елемента *img*. За замовчуванням має значення *bottom*;
- *accept* – конкретизує тип файла. Використовується лише спільно з параметром *type="file"*. Значення задається у вигляді *MIME*-типу.

Приклад 1.20:

```
<form name="form1" action="http://www.css.ua/cgi-bin/test.pl">
<input type="hidden" name="info" value="Реєстрація студентів">
<input type="radio" name="pol" value="male" checked> Не одружений <br>
<input type="radio" name="vol" value="female"> Одружений<br>
Ім'я:<br>
<input type="text" name="textfield" value="ім'я" size="30" maxlength="60"><br>
Пароль:<br>
<input type="password" width="10" name="passwd"><br><br>
<input type="submit" value="Відправити">
</form>
```

Приклад 1.21:

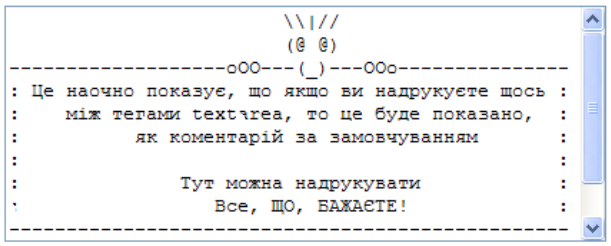
```
Хочу отримувати такі видання:<br>
<form name="form2" action="http://www.igf.ru/cgi-bin/magazines.pl">
<input type="checkbox" name="m1">Сьогодні<br>
<input type="checkbox" name="m2">Факти та коментарі <br>
<input type="checkbox" name="m3" checked>Урядовий кур'єр<br>
<input type="image" src="/img/button.gif" width="60" height="30">
</form>...
```

Завдання до самостійної роботи

1. Створіть таблицю за зразком:

№ з/п	Прізвище та ініціали студента	Модульна рейтингова оцінка									
		Модуль I					Модуль II				
		Поточна	Контрольна		Підсумкова		Поточна	Контрольна		Підсумкова	
			Бали	Нац. шкала	Бали	Нац. шкала		Бали	Нац. шкала	Бали	Нац. шкала

2. Використовуючи табличну верстку, створіть сторінку про свою навчальну групу.
3. Створіть форму, в якій повинно розміщуватися:
   - список перемикачів: синій, бузковий, блакитний, зелений, причому слова повинні відображатися відповідним кольором;
   - список прапорців повинен розміщуватися в центрі сторінки і містити назви восьми геометричних фігур;
   - дані про себе користувач повинен вводити в текстові поля (Прізвище, Ім'я), а кафедра у (КСУ або КСМ) і номер групи (320-326) вибираються за допомогою списків;
   - при натисканні кнопки «Відправити» повинна запускатися програма для відправки такого повідомлення:
     cvet= синій
     figura= еліпс
     figura =прямокутник
     fam= Шевченко
     imya= Максим
     fakultet= КСУ
     gruppa= 325
   - після натискання кнопки «Очистити поля форми» всі поля форми мають набути початкового вигляду;
   - в кінці сторінки створіть в точно таке оголошення (оголошення підказка для виконання цього завдання):



Контрольні запитання та завдання

1. Назвіть елементи побудови таблиць.
2. Поясніть особливості «гумової» табличної верстки сайта.
3. Поясніть особливості «жорсткої» табличної верстки сайта.
4. Назвіть призначення форм.
5. Як відправляються дані форми?
6. Які атрибути використовуються з елементом розмітки *<input>*?

Лабораторна робота 1.3.

Застосування каскадних таблиць стилів CSS. Блокова модель CSS

Мета: Ознайомитися з методами застосування каскадних таблиць стилів та особливостями блокової моделі CSS.

Вимоги до обладнання та програмного забезпечення

Лабораторна робота виконується на ПК з використанням програм Microsoft Expression Web 4, TopStyle 4 та браузерів Google Chrom, Opera, Mazila Firefox.

Основні теоретичні відомості

CSS *(Cascading Style Sheets)* – мова таблиць стилів, яка дозволяє прикріплювати стиль (наприклад, шрифти і колір) до структурованих документів (наприклад, документів HTML і додатків XML). Зазвичай CSS-стилі використовуються для створення і зміни стилю елементів веб-сторінок і призначених для користувача інтерфейсів, написаних мовами HTML *і* XHTML, але також можуть бути застосовані до будь-якого виду XML-документа, зокрема й XML, SVG *і* XUL. Відокремлюючи стиль подання документів від вмісту документів, CSS спрощує створення веб-сторінок і обслуговування сайтів.

Існують такі способи підключення стилів:

1. Зв'язування таблиці стилів з документом *(linking)*. Підключення CSS-файла відбувається за допомогою елемента *link*. Зовнішня таблиця стилів становить собою текстовий файл, який містить опис стилів:
   <link rel="stylesheet" type="text/css" href="style.css" />
2. Вкладення стилів *(embeding)*. Інформація про стилі міститься в заголовку веб-сторінки в елементі *style*.
   <head>
   <title>Назва документа</title>
   <style>
   h1{color:red; font-style:italic; font-size:32px}
   </style>
   </head>
3. Вбудовування стилів *(inline)*. Стиль поміщається всередину тега за допомогою атрибута *style*.
   <h1 style="font-size:40px; color:red; text-align:center">Заголовок розділу</h1>
4. Імпорт стилів *(import)*
   <head>
   <title>Назва </title>
   <style type="text/css">
   @import: url(mystyle.css);
   </style>
   </head>

Основним поняттям у CSS є селектор – це ім'я стилю, для якого додаються параметри форматування. Як селектор виступають елементи розмітки, класи й ідентифікатори. Загальний спосіб запису має такий вигляд:

селектор {властивість: значення;}

Спочатку вводять ім'я селектора, наприклад, *<h1>*. Це означає, що всі стильові параметри застосовуватимуть до елемента *<h1>*, потім йдуть фігурні дужки, в яких записують стильову властивість, а його значення вказують після двокрапки. Стильові властивості розділяють між собою крапкою з комою, в кінці цей символ можна не писати.

Приклад 1.22

```
<html>
<head>
<style type="text/css">
<!--
body {background-color: #9999cc;}
h1 {background-color: green;
font-size: 200%;
color: white;
text-align: center;}
-->
</style>
</head>
```

```
<body>
<h1> Синтаксис CSS </h1>
</body>
</html>
```

Коли потрібно застосувати однакові стилі для декількох елементі тоді слід перед фігурними дужками перелічити їх назви. Отримаємо груповий селектор: *p, h1 {color: darkred;}*.

Існують ще два способи визначення стилів – а саме через класи й ідентифікатори.

Припустимо, нам необхідно задати властивості елемента *<p>*, але кожен абзац повинен відрізнятися від попереднього. Досягти даної мети відомими нам способами неможливо, ось тут і приходять на допомогу класи.

Приклад 1.23

```
<html>
<head>
<style type="text/css">
<!--
p.one {background-color: #d6d2dd; font-style: regular; font-size: 15;}
p.two {background-color: #d1ded7; font-style: bold; font-size: 20;}
p.three {background-color: #ddd8d2; font-style: italic; font-size: 25;}
-->
</style>
</head>
<body>
<p class="one">CSS має дуже простий синтаксис.
<p class="two">Знаючи CSS, можна створити якісні сайти
<p class="three"> Каскадні таблиці стилів являють собою опис різних елементів HTML і створені вони для розширення властивостей останніх. Вперше стилі були запропоновані WWW Consorcium у рамках розробки специфікації HTML 3.0.
</body>
</html>
```

Коли треба створити клас, не прив’язаний до певного елементу, тоді визначення елемента потрібно опустити.

Приклад 1.24

```
<html>
<head>
<title>Селектор клас</title>
<style type="text/css">
.one {color: green;}
.two {color: blue;}
</style>
</head>
<body>
<div class="one">Текст відображається зеленим кольором.
<p class="two">Абзац відображається синім кольором.
<hr class="one">
</body>
</html>
```

Якщо потрібно виділити один елемент, унікальний, відмінний від всіх інших у документі, тоді можна використати ідентифікатор.

Приклад 1.25

```
<html>
<head>
<title>Селектор ідентифікатор</title>
<style>
#firstheader {
color: red; // задає червоний колір тексту
```

```
font-weight: bold; // шрифт стане жирним
text-align: center; // центрування
}
</style>
</head>
<body>
<h1 id="firstheader"> Перший заголовок на сторінці </h1>
</body></html>
```

Для форматування символів за допомогою CSS використовують властивості *font-family, font-size, font-style, font-variant, font-weight.*

font-family – визначає сімейство шрифтів. Можна задавати список шрифтів, у такому разі останній елемент списку – назва сімейства.

Основні сімейства шрифтів:

- *serif* – шрифти з зарубками (приклад: *Times New Roman*);
- *sans-serif* – шрифти без зарубок (приклади: *Arial, Verdana*);
- *cursive* – курсивні шрифти (приклад: *Comic Sans MS*);
- *fantasy* – декоративні шрифти (приклад: *Monotype Corsiva*);
- *monospace* – шрифт фіксованої ширини, ширина кожного символу в такому сімействі однакова (приклад: *Courier New*).

Якщо назва шрифту складається з двох або більше слів, її треба писати в лапках: *p {font-family: "Times New Roman", Times, serif; }.*

font-size – задає розмір шрифту. Абсолютні значення розміру шрифту задаються як *xx-small, x-small, small, medium, lagre, x-large, xx-large*. Відносні задаються як *larger, smaller*, або у відносних значеннях довжини, наприклад *em* (висота шрифту елемента), *ex* (висота символу *x*), пункти *(pt),* пікселі *(px)*, відсотки *(%)* та ін. (За 100 % береться розмір шрифту батьківського елемента).

font-style – визначає стиль шрифту з обраного сімейства: нормальний *(normal)*, курсивний *(italic)* або похилий *(oblique)*, який генерується з нормального невеликим нахиленням символів.

font-variant – визначає, як потрібно зображати малі літери – залишити їх без модифікацій чи робити їх всі великими зменшеного розміру. Такий спосіб зміни символів називається капітеллю: *normal* і *small-caps.*

font-weight – визначає жирність шрифту із заданого сімейства. Може задаватися числами 100, 200, ..., 900 або ключовими словами: *extra-light, light, demi-light*, *medium (normal)*, *demi-bold, bold, extra-bold*. Гарантовано підтримуються лише *normal (400), bold (700)*. Може визначатися відносно: *bolder, lighter.*

Можна задати всі властивості шрифту одночасно. Синтаксис:

font: font-style font-variant font-weight font-size/line-height font-family: p {font: italic small-caps 600 18pt/24pt "Arial, sans-serif"}.

Якщо якесь значення не задане, використовується відповідне значення за замовчуванням. Усі властивості шрифтів успадковуються вкладеними елементами і можуть застосовуватися до всіх елементів HTML.

Блокова модель CSS описує прямокутний блок, створюваний для елемента в дереві документа і виводиться згідно візуальної моделі форматування (специфікація W3C). Це означає, що елемент у HTML-документі виводиться всередині свого окремого прямокутного блока.

У специфікації HTML4.01 за замовчуванням до блочних елементів відносяться *address, blockquote, div, dl, fieldset, form, hr, h1-h6, noscript, ol, p, pre, table, ul.*

Типовим блоковим елементом HTML є елемент розмітки *div*. У елемента розмітки *div* доступні всі універсальні атрибути і події HTML, унікальних атрибутів у елемента розмітки *div* немає, вміст елемента *div* оформляється за допомогою таблиць стилів CSS.

Блок складається з безпосередньо контенту (внутрішнього вмісту), внутрішніх відступів, кордонів і, нарешті, зовнішніх відступів.

Для управління кожною із складових частин блока існують відповідні CSS-властивості: внутрішні відступи – *padding*, межі - *border*, зовнішні відступи – *margin*. За бажання ці властивості можна задати для кожної сторони блока окремо (рис. 1.1):

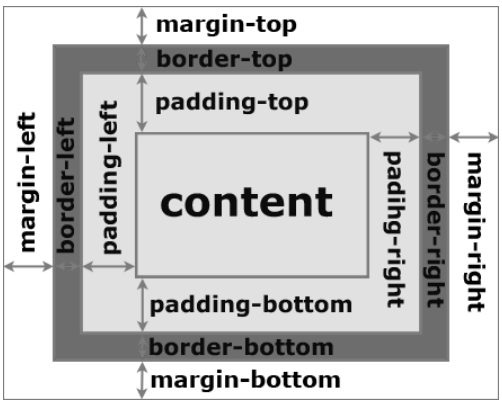


Рис. 1.1. Властивості блока

Приклад 1.26:

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset=utf-8">
<title>margin</title>
<style type="text/css">
BODY {
margin: 0; /* Прибираємо відступи */
}
div.big {
margin: 20%; /* Відступи навколо елемента */
background: #fd0; /* Колір фону */
padding: 10px; /* Поля навколо тексту */
}
div.small {
border: 3px solid #666; /* Параметри рамки */
padding: 10px; /* Поля навколо тексту */
margin: 10px; /* Відступи навколо */
}
</style>
</head>
<body>
<div class="big">
<div class="small">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diem nonummy nibh
euismod tincidunt ut lacreet dolore magna aliguam erat volutpat. Ut wisis enim
ad minim veniam, quis nostrud exerci tution ullamcorper suscipit lobortis nisl
ut aliquip ex ea commodo consequat.
</div>
</div>
</body>
</html>
```

Для елемента можна задати межі (рамку) і два види відступів. Чим же вони відрізняються?

Основна відмінність відразу впадає в очі: *padding* – це відступ між контентом і межею, а *margin* – це відступ між межею і «зовнішнім світом».

Звідси випливає друга відмінність – якщо для елемента задати фон *(background),* то цим фоном заллється і контент, і внутрішній відступ *(padding). Margin,* перебуваючи зовні, завжди залишається прозорим.

Padding, якщо провести аналогію – це поля на аркуші паперу. Вони мають той самий колір, що і лист, але текст на них не заходить. *Margin* – це відстань від краю аркуша до іншого краю, що лежить поруч листа або, скажімо, до краю столу.

Третя відмінність полягає в тому, що *padding* та *margin* по-різному беруть участь у підрахунку загальної ширини блока. Це залежить від прийнятої блочної моделі.

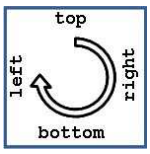


Рис. 1.2. Порядок встановлення параметрів відступів

Для браузера кожен елемент розмітки – це контейнер, у якого є вміст, внутрішній відступ, зовнішні поля, а також рамка. Блок займає простір на сторінці, що дорівнює сумі ширини вмісту, відступу, поля і рамки.

Margin – властивість для встановлення ширини полів для всіх сторін елемента або окремо. У цієї властивості може бути від одного до чотирьох значень. Якщо є лише одне значення, воно буде присвоєно відразу до всіх полів. Якщо два значення, перше з них присвоюється верхньому і нижньому полю, а друге – лівому і правому. Якщо ж три, то перше значення присвоюється верхньому полю, друге – лівому і правому, а третє – нижньому (рис. 1.2).

Можливі атрибути:

- *margin-top* – ширина верхнього поля;
- *margin-right* – ширина правого поля;
- *margin-bottom* – ширина нижнього поля;
- *margin-left* – ширина лівого поля;
- *inherit* – застосовується значення батьківського елемента.

Padding – властивість для встановлення ширини відступів відразу для всіх сторін елемента або окремо. У цієї властивості може бути від одного до чотирьох значень. Якщо є лише одне значення, воно буде присвоєно відразу всім полям. Якщо два значення, перше з них присвоюється верхньому і нижньому полю, а друге – лівому і правому. Якщо ж три, то перше значення присвоюється верхньому полю, друге лівому і правому, а третє – нижньому.

Можливі атрибути:

- *padding-top* – ширина верхнього відступу;
- *padding-right* – ширина правого відступу;
- *padding-bottom* – ширина нижнього відступу;
- *padding-left* – ширина лівого відступу;
- *inherit* – застосовується значення батьківського елемента.

border – властивість для визначення межі відразу з усіх сторін елемента. Значення встановлюється однаковими для всіх його сторін.

Можливі атрибути:

- *border-width* – товщина межі;
- *border-style* – стиль межі;
- *border-color* – колір межі;
- *inherit* – застосовується значення батьківського елемента.

Рис. 1.3. Типи границі блока

border-width – властивість для встановлення ширини межі відразу для всіх сторін елемента або окремо. У цієї властивості може бути від одного до чотирьох значень. Можливі атрибути:

- *border-top-width* – товщина верхньої сторони;
- *border-right-width* – товщина правої сторони;
- *border-bottom-width* – товщина нижньої сторони;
- *border-left-width* – товщина лівої сторони;
- *inherit* – застосовується значення батьківського елемента.

border-style – властивість для установки стилю межі відразу для всіх сторін елемента або окремо. У цієї властивості може бути від одного до чотирьох значень.

Можливі атрибути:

- *border-top-style* – стиль верхньої сторони.
- *border-right-style* – стиль правої сторони.
- *border-bottom-style* – стиль нижньої сторони.
- *border-left-style* – стиль лівої сторони.
- *inherit* – застосовується значення батьківського елемента.

border-color – властивість для установки кольору границь відразу для всіх сторін елемента. У цієї властивості може бути від одного до чотирьох значень.

Можливі атрибути:

- *border-top-color* – колір верхньої сторони.
- *border-right-color* – колір правого боку;
- *border-bottom-color* – колір нижньої сторони;
- *border-left-color* – колір лівої сторони;
- *inherit* – застосовується значення батьківського елемента.

Завдання до самостійної роботи

1. Виконайте приклади.
2. Відредагуйте сторінку, присвячену навчальній групі за допомогою стилів.
3. Розробіть персональну веб-сторінку, яка повинна включати вашу біографію використовуючи блокову модель таблиці стилів. Автобіографія повинна мати чіткий план і структуру (фото обов'язково).

Контрольні запитання та завдання

1. Назвіть призначення каскадних таблиць стилів.
2. Які ви знаєте способи використання стилів на веб-сторінках?
3. Як за допомогою CSS виконати відформатувати текст?
4. Назвіть особливості блокової моделі CSS.
5. Які існують відступи в блоковій моделі CSS?
6. У чому полягає призначення елемента розмітки *div*?

Лабораторна робота 1.4

Псевдокласи і псевдоелементи. Блокова верстка

Мета: ознайомитися з методами застосування псевдокласів і псевдоелементів та методами блокової верстки.

Вимоги до обладнання та програмного забезпечення

Лабораторна робота виконується на ПК з використанням програм Microsoft Expression Web 4, TopStyle 4 та браузерів Google Chrom, Opera, Mazila Firefox.

Основні теоретичні відомості

Псевдокласи визначають динамічний стан елементів, який змінюється за допомогою дій користувача. Прикладом такого стану може бути текстове посилання, яке змінює свій колір при наведенні на нього курсора миші. При використанні псевдокласів браузер не перенавантажує поточний документ, тому за допомогою псевдокласів можна отримати різні динамічні ефекти на сторінці.

Синтаксис застосування псевдокласів такий: *Селектор:Псевдокласс { Опис правил стилю }*

Псевдокласи застосовуються до імен ідентифікаторів або класів *(a.menu :hover {color: green})*, а також до контекстних селекторів *(.menu a :hover {background: #fc0})*. Якщо псевдоклас вказується без селектора попереду *(:hover)*, то він застосовуватиметься до усіх елементів документа.

Існують псевдокласи посилань. У цій категорії два псевдокласи *:link* і *:visited*. За їх допомогою задають стилі для елементів, що є посиланнями (під такими розуміються елементи, що мають атрибут *href*), і для тих посилань, на які користувач уже натискав.

:link – застосовується до посилань, які ще не відвідував користувач, і задає для них стильове оформлення;

:visited – для вже відвіданих посилань.

Приклад 1.27.

```
a:link {
   color: #036;
}
a:visited {
color: #80bd34;
}
```

Існують динамічні псевдокласи. Ці псевдокласи застосовуються до елементів залежно від дій користувача. *CSS2* визначає три такі псевдокласи *-:hover,:active і :focus*.

:hover – елемент, над яким розміщується покажчик миші. Щойно покажчик йде за межі елемента, стилі, задані цим псевдокласом, скасовується.

:active – елемент, активований користувачем (наприклад, посилання або кнопка у момент клацання).
:focus – елемент, якому належить фокус введення.

Приклад 1.28

```
a:hover
{
text-decoration: none;
background-color: #333099;
}
```

Використовуючи динамічні псевдокласи можна оживити веб-сторінки без використовування мови JavaScript.

Як селектор псевдокласу може виступати не лише який-небудь елемент – елемент розмітки, але й клас або ідентифікатор, наприклад: *.menu :hover { color:#ff0000;}*

Приклад 1.29

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="windows-1251">
  <title>Приклади застосування псевдокласів</title>
    <style>
   a:link {
    color: #036; /* Колір невідвіданих посилань */
    }
   a:visited {
    color: #606; /* Колір відвіданих посилань */
    }
   a:hover {
color: #f00; /* Колір посилань при наведенні на них курсору миші */
}
a:active {
 color: #ff0; /* Колір активних посилань  */
}
input:focus {
color: red; /* Червоний колір */
}
table { border-spacing: 0; }
td { padding: 4px; }
tr:hover {
background: #fc0; /* Змінює колір тла рядка таблиці */
}
</style>
</head>
<body>
<p>
<a href="1.html">Посилання №1</a> |
<a href="2.html">Посилання №2</a> |
<a href="3.html">Посилання №3</a></p>
<form action="">
<p><input type="text" value="Текстове поле №1"></p>
<p><input type="text" value="Текстове поле №2"></p>
</form>
<table border="1" width="100%" cellpadding="5">
<tr>
<th>Комірка №1 </th>
<th>Комірка №2 </th>
</tr>
<tr>
```

```
<td>Комірка №3 </td>
<td>Комірка №4 </td>
</tr>
</table>
 </body>
</html>
```

Існують структурні псевдокласи. До цієї групи належать псевдокласи, які визначають положення елемента в дереві документа, і застосовують до нього стиль залежно від його статусу.

:first-child – застосовується до першого дочірнього елемента селектора, який розташований в дереві елементів документа. Наприклад: *ul > li:first-child {margin-top: 10px;}*.

Аналогічно використовується псевдоклас *:last-child*.

Приклад 1.30

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Застосування псевдокласу :last-child </title>
<style type="text/css">
b:first-child {
color: blue; /* Червоний колір тексту */
}
p {
text-indent: 1em; /* Відступ першого рядка */
}
p:first-child {
text-indent: 0; /* Видаляємо відступ для першого абзацу */
}
</style>
</head>
<body>
<p><b>Lorem ipsum</b> dolor sit amet, <b>consectetuer</b>
adipiscing <b>elit</b>, sed diem nonummy nibh euismod tincidunt
ut lacreet dolore magna aliguam erat volutpat.</p>
<p><b>Ut wisis enim</b> ad minim veniam, <b>quis nostrud</b>
exerci tution ullamcorper suscipit lobortis nisl ut aliquip
ex ea <b>commodo consequat</b>.</p>
</body>
</html>
```

Псевдоелементи дають змогу задати стиль елементів, що не визначені в дереві елементів документа, а також генерувати вміст, якого немає у вихідному коді тексту.

Синтаксис використання псевдоелементів такий: *Селектор:Псевдоелемент { Опис правил стилю }.*

:first-letter – Задає стилі для першої літери всередині елемента (наприклад, перші літери абзаців сторінки мають інший стиль):

Приклад 1.31

```
p:first-letter {
font-size: 120%;
}
```

:first-line – Визначає стиль першого рядка блочного тексту.

:after :before – дають змогу вставити генерований контент на ходу після або до певного елемента. Щоб задати вставлений текст, використовується спеціальна властивість *content*. Властивість *content* задає вміст, який спочатку відсутній на сторінці. Виводиться до або після елемента, до якого цю властивість застосували. Використовується спільно з псевдоелементами *after* та *before*.

Приклад 1.32

```
<!DOCTYPE HTML>
<html>
```

```
<head>
<meta charset=windows-1251">
<title>Застосування псевдоелементів</title>
<style>
p:first-line {
color: red; /* Червоний колір тексту */
font-style: italic; /* курсив */
}
p:first-letter {
/* Гарнітура шрифту першої літери */
font-family: ‘Times New Roman’, Times, serif;
font-size: 200%; /* Розмір шрифту першого символу */
color: blue; /* Червоний колір тексту */
}
p:before {
/* Додаємо перед елементом списку символ в юнікод */
content: "\30aa ";
}
</style>
</head>
<body>
<p>Lorem ipsum dolor sit amet,consectetuer>
adipiscing elit, sed diem nonummy nibh euismod tincidunt
ut lacreet dolore magna aliguam erat volutpat.
Ut wisis enim ad minim veniam,quis nostrud
exerci tution ullamcorper suscipit lobortis nisl ut aliquip
ex ea commodo consequat.</p>
</body>
</html>
```

Використовуючи псевдокласи та псевдоелементи можна створювати динамічне меню навігації.

Меню дає змогу швидко переміщатися основними розділами сайта.

Меню може бути:

- Горизонтальним;
- Вертикальним;
- Багаторівневим.

Меню може бути дуже різноманітним, і зверстати його можна дуже різними способами: таблицями, блоками, просто посиланнями та ін.

Створимо горизонтальне меню за допомогою списків.

Приклад: 1.33

```
<!DOCTYPE HTML>
<html>
<head>
<title>Горизонтальне меню</title>
<style>
.menu_nav {
list-style: none; /* ховаємо маркери */
}
.menu_nav li {
float: left; /* розташовуємо елементи списку в один ряд */
margin-right: 15px; /* робимо відступ, щоб пункти меню не зливалися */
}
a {text-decoration: none;} /*Прибираємо підкреслення посилання*/
.menu_nav li a {
display: block; /* змінюємо відображення на блок, щоб мати можливість задавати внутрішні відступи */
padding: 3px 5px;
background: #ff7f50;
```

```
color: #000;
position: relative;
}
.menu_nav li a:hover {
background: #d2b48c;
color: #fff;
}
</style>
</head>
<body>
<ul class="menu_nav">
<li><a href="#">Головна</a></li>
<li><a href="#">Інститут</a></li>
<li><a href="#">Кафедри</a></li>
<li><a href="#">Розклад</a></li>
</ul>
</body></html>
```

Елемент * * – блоковий елемент який призначений для виділення фрагмента документа задля зміни виду даних. Зазвичай внаслідок вид блока управляється за допомогою стилів. Щоб не описувати щоразу стиль всередині елемента розмітки, можна виділити стиль у зовнішню таблицю стилів, а для елемента додати атрибут *class* або *id* з ім'ям селектора.

Як і при використанні інших блочних елементів, вміст елемента * * завжди починається з нового рядка. Після нього також додається перенесення рядка.

Завдяки цьому елементу HTML-код розпадається на ряд чітких наочних блоків, внаслідок чого верстка шарами називається також блокової верстkою. Код при цьому виходить більш компактним, ніж при табличній верстці, до того ж пошукові системи його краще індексують.

Розглянемо приклади сторінок (рис. 1.4).

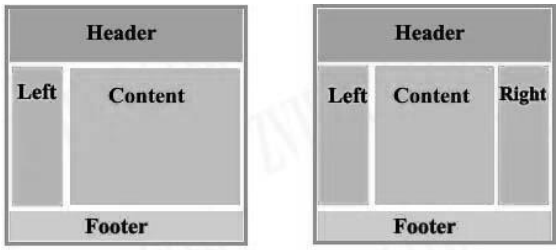


Рис. 1.4. Приклад для блочної розмітки

Тут наведено представлені варіанти для «гумової» верстки з двома і трьома колонками. Виконаємо розмітку двох колонок.

Спочатку виконаємо розмітку HTML:

Приклад 1.34

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset=UTF-8">
<title> Блокова верстка </title>
<link href="style.css" rel="stylesheet" type="text/css">
</head>
<body>
<div id="main_container">
<div id="header">
<h1>Header</h1>
</div>
<div id="left">
<h3>left Column</h3>
</div>
<div id="content">
<h1>Main Content </h1>
```

```
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Praesent aliquam, justo convallis luctus rutrum, erat nulla fermentum Integer turpis arcu, pellentesque eget, cursus et, fermentum ut, sapien. Fusce metus mi, eleifend sollicitudin, molestie id, varius et, nibh. Donec nec libero.</p>
<h1>Content</h1>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</p>
<p>Quisque ornare risus quis ligula Phasellus tristique purus a augue condimentum adipiscing. Aenean sagittis. Etiam leo pede, rhoncus venenatis, tristique in, vulputate at, odio.</p>
</div>
<div id="footer">
<p><strong>Footer</strong></p>
</div>
</div>
</body>
</html>
```

Створення сторінки починається з головного контейнера в який додаються інші. Наявність цього контейнера дає змогу регулювати інші параметри сторінки, що значно полегшує верстку блоками.

Далі створюємо файл *style.css,* в який додаємо стилі.

Приклад 1.35

```
body, html {
margin:0px; /*Обнуляємо поля і відступи, тому що різні браузери по-різному їх сприймають. */
padding:0px;
text-align:center; /*Вирівнюємо макет (вміст елемента body буде посередині) по центру в старих версіях браузерів */
}
#container{
margin:0 auto; /*вирівнюємо макет по центру в сучасних браузерах */
width:650px;
}
/*Вводимо стилі для голови сайта */
#header{
background-color: #00cc66;
}
/* Вводимо для лівої колонки сайта */
#left{
background-color: #ccff33;
}
/* Вводимо стилі для блоку контента */
#content{
background-color: #0099ff;
}
#content h1 {
margin:0px; /* Обнуляємо відступи для заголовка першого рівня, що міститься в блоці контенту.*/
}
/* Вводимо стилі для підвалу сайта */
#footer{
background-color: #ffcc33;
}
```

Розміщуємо блок *left* ліворуч від блока Content. Змінюємо ширину блока *Left* на *150px* і встановлюємо для нього обтікання:

```
#left{
width:150px; /*ширина колонки */
float:left; /*обов'язкове вирівнювання по лівому краю,
з включенням обтікання*/
}
```

Зміщаємо блок Content праворуч на ширину блока *Left*:

```
#content {
```

```
background-color:#83a0f3;
margin-left:150px;
}
```

Те саме можна зробити інакше, встановивши блоку *Left width:20%* і *float:left;,* а *блоку Content width:80%* і *float:right.*

При цьому не забудьте поставити для блока *footer* значення *cltar:both*, інакше при збільшенні кількості тексту в попередніх блоках може вийти сивий ефект.

Завдання до самостійної роботи

1. Оформіть список дисциплін, що вивчаються на четвертому курсі за допомогою вивчених псевдокласів і псевдолелементов. Обов'язково використовуйте: псевдокласи посилань, динамічні псевдокласи, псевдоелементи.
2. Використовуючи *Приклади 1.34* та *1.35* розробіть трьостопчиковий блоковий макет. Як зразок візьміть двостопчиковий макет і розтягніть вміст на всю ширину сторінки. Використовуючи раніше створені сторінки – «Хобі» і «Автобіографія», створіть сайт оформлений в єдиному стилі.
3. Розробіть динамічне багаторівневе навігаційне меню для сайта з використовуючи вивчений матеріал.
4. З допомогою меню необхідно зв'яжіть усі наявні сторінки.
5. Сторінки повинні містити зображення.

Контрольні запитання та завдання

1. Назвіть призначення псевдокласів.
2. Які ви знаєте псевдоелементи?
3. Як за допомогою CSS виконати форматування тексту?
4. Яку структуру мають сторінки, створені за допомогою блокової верстки?

Лабораторна робота 2.1.

Додавання сценаріїв JavaScript на веб-сторінку. Основні поняття та визначення: об'єкт, метод, властивості, події. Обчислення з використанням сценаріїв JavaScript.

Мета: навчитися використовувати різні способи доступу до властивостей і методів об'єктів JavaScript для внесення змін е HTML-документ.

Вимоги до обладнання та програмного забезпечення

Лабораторна робота виконується на ПК з використанням програм Microsoft Expression Web 4, TopStyle 4 та браузерів Google Chrom, Opera, Mazila Firefox.

Основні теоретичні відомості

JavaScript – це мова програмування призначена насамперед для створення інтерактивних HTML-сторінок.

Скрипти можуть бути пов'язані з HTML-документом трьома способами:

- у розділі *head*;
- у розділі *body*;
- підключені із зовнішнього файла.

У розділі *head* зазвичай поміщають функції. Виклики функції краще помістити в *body*.

У тілі сторінки JavaScript розміщують у самому кінці секції *body* (перед </*body*>), якщо не хочуть, щоб скрипт почав виконуватися до повного завантаження документа і це призвелоб до помилок. На сторінці може бути розміщено необмежену кількість скриптів, у тому числі і в <*body*>, і в <*head*> одночасно.

Скрипт також може зберігатися в зовнішньому файлі з розширенням *.js*. Використовувати зовнішні файли скриптів зручно у випадках, коли потрібно визначати код, який працюватиме на декількох сторінках веб-сайта.

Зовнішні скрипти, як і звичайні підключаються до сторінок за допомогою елемента <*script*>, проте в цьому випадку вміст елемента має залишатися порожнім і до нього має бути додано атрибут *src*, який містить адресу зовнішнього *.js* файла:

<*script type="text/javascript" src="exa.js"*></*script*>

JavaScript – це об'єктно-орієнтована мова програмування (ООП), заснована не на обробці команд коду, а на присвоєнні окремим елементам програми конкретних подій і виконання їх, якщо така подія була.

OBJECT (об'єкт) - це те, з чим проводиться дія, подія. Це може бути документ, що відкривається у вікні браузера, або саме вікно браузера, або якась частина документа, елементи розмітки. Об'єкт повинен мати унікальне ім'я *(ID),* щоб до нього можна було звернутися. Кожен об'єкт має свої методи (рис. 2.1).

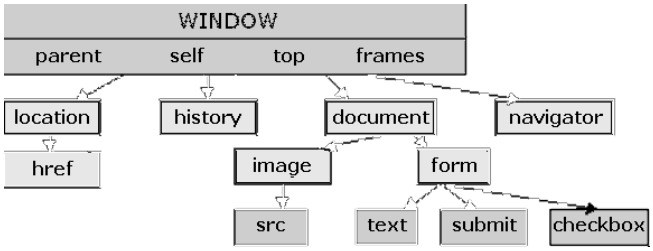


Рис. 2.1. Ієрархія об'єктів у JavaScript

METHOD (метод об'єкту) – це дії, які можна виконувати над об'єктом такого типу, або які сам об'єкт може виконувати.

Між ім'ям об'єкта і методом обов'язково ставлять розділовий оператор «крапка», після методу в дужках параметри методу:

Об'єкт . Метод ("параметри методу")

Параметри методу відносяться до типу даних - рядки символів. Рядки символів потрібно обов'язково взяти в лапки або в одинарні або подвійні лапки.

PROPERTY (властивість) – кожен об'єкт має свої властивості. Один і той самий об'єкт може мати багато властивостей. Часто ці властивості потрібно змінити при виникненні деякої події.

Для зміни властивості об'єкта слід дотримуватися синтаксису:

Об'єкт.властивість об'єкта = "нове значення властивості "

Наприклад, для зміни кольору тла документа *HTML* (ім'я цієї властивості *bgColor*) слід написати таке:

<*script type="text/javascript"*>
document.bgColor= 'red'
</*script*>

EVENT (подія) – це все, що сталося: відкриття вікна, завантаження в нього документа, клік клавішею мишки або просто переміщення курсору по екрану, натискання клавіші на клавіатурі. Це все події, і вони можуть ініціювати запуск великих і малих програм табл. 2.1.

Таблиця 2.1

Подія	**Опис**
OnAbort	Відбувається при перериванні завантаження графічного зображення
OnBlur	Відбувається, коли який-небудь елемент втрачає фокус
OnChange	Відбувається при зміні значення текстового поля
OnClick	Відбувається при натисканні кнопки миші в області елемента
OnError	Відбувається при помилці завантаження документа або графічного зображення
OnFocus	Відбувається при отриманні елементом фокуса.
OnLoad	Відбувається по завершенні завантаження сторінок або графічного зображення
onMouseOver	Відбувається при переміщенні курсора миші в області елемента
onMouseOut	Відбувається при переміщенні курсора миші з області елемента
OnReset	Відбувається при натисканні кнопки типу *Reset*
OnSelect	Відбувається при виборі тексту в текстовому полі
OnSubmit	Відбувається при натисканні кнопки типу *Submit*
OnUnload	Відбувається при переході на іншу сторінку або завершенні роботи з браузером

Щоб скрипт зреагував на подію, в ньому повинна бути описана спеціальна функція. Функції, які реагують на події, називаються обробниками подій *(event handlers)*. Є кілька способів призначити обробник події.

Обробник може бути призначений прямо в розмітці, в атрибуті, який називається *on*-подія. Наприклад, щоб прикріпити *click*-подія до *input*-кнопці, можна присвоїти обробник *onclick* у такий спосіб:

Приклад 2.1:

<input id="b1" value="Натисни мене" onclick="alert(‘Дякую!’)" type="button"/>

Можна використати властивість об’єкта. Для цього достатньо отримати елемент в *JavaScript*, наприклад *id,* і призначати обробник, використовуючи властивість DOM-елемента подія:

Приклад 2.2:

```
<input id="b1" type="button" value="Натисни мене"/>
<script>
var elem = document.getElementById(‘b1’);
elem.onclick = function() { alert(‘Спасибі’);}
</script>
```

Обробником можна призначити вже наявну функцію:

Приклад 2.3:

```
function sayThanks() {
alert(‘Дякую!’);
}
<input id="b1" value="Натисни мене" onclick=sayThanks() type="button"/>
//або
<script>
document.getElementById(‘b1’).onclick = sayThanks;
</script>
```

Характеристики подій браузер записує в об’єкт події, до якого може звернутися обробник. Усі сучасні браузери передають об’єкт події першим аргументом в обробник.

Приклад 2.4:

```
<input type="button" onclick="alert(event.type)" value="Тип події">
<input type="text" onMouseOver="alert(event.type)" value="Тип події">
```

Події клавіатури відбуваються при натисканні клавіші *keydown/keyup* (натискання/відпускання) і дозволяють отримати її код у властивості *keyCode*.

Код клавіші однаковий у будь-якій розкладці і в будь-якому регістрі. Наприклад, клавіша може означати символ *"w", "W"* або *"ц", "Ц"* в українській розкладці, але її код буде завжди однаковий: 87. Скан-коди перетворюються драйвером клавіатури в ASCII-коди в англійській розкладці, для різних типів клавіатур вони можуть відрізнятися.

Приклад 2.5:

```
<input onkeydown="this.nextSibling.innerHTML = event.keyCode"> <b></b>
```

Вирази будуються з літералів, змінних, знаків операцій, дужок. У результаті обчислення виразу виходить єдине значення, яке може бути або числом (цілим або дійсним), або рядком, або логічним значенням. Використовувані у виразі змінні потрібно ініціалізувати. Якщо при обчисленні виразу зустрічається невизначена або не ініціалізована змінна, фіксується помилка. У JavaScript існують літерали *null* для визначення невизначеного значення. Якщо змінній присвоєно значення *null*, вона вважається такою, що ініціалізовна.

Вирази формуються з операндів і позначень операцій. Наприклад, у формулі *a*b* операндами є *a i b,* позначенням операції - знак *.

Залежно від типу обчисленого значення вирази можна розділити на арифметичні, логічні й строкові. Арифметичні вирази виходять при виконанні операцій, наведених у табл. 2.2.

Таблиця 2.2

Позначення	Операція
+	Додавання
-	Віднімання
*	Множення
/	Ділення
%	Залишок від ділення цілих чисел
++	Збільшення операнда на одиницю
--	Зменшення операнда на одиницю

Оператори у виразі обчислюються зліва направо відповідно до пріоритетів арифметичних операцій. За

потреби за допомогою дужок можна змінити порядок виконання операцій.

Операції відношення застосовні до операндів будь-якого типа. Результат операції – логічне значення *true*, якщо порівняння правильне, і - *false* інакше. Наявні операції порівняння наведені в (табл.. 2.3)

Таблиця 2.3

< (менше) > (більше)	<= (менше або дорівних)	== (дорівнює)	!= (не дорівнює)	>= (більше або дорівнює)

Операція *!* (логічне НЕ) застосовується до операндів логічного типу, якщо значення операнда *a* дорівнює *true*, то значення виразу *!a – false*, якщо значення операнда *a* дорівнює *false*, то значення виразу *!a – true*.

Над рядковими значеннями визначена операція, яка називається конкатенація (з'єднання) рядків, рядкових значень операндів, наприклад, у результаті виконання операнда присвоєння:

st=" поточне " + "стан"

Змінна *st* набуде значення "поточний стан".

Можливий ще один варіант запису виконання конкатенації рядків.

Наприклад, задано дві змінні *st1="поточний"* і *st2= "момент"*.

Тоді в результаті виконання операції *st1+=st2* змінна *st1* набуде значення "теперішній момент".

Пріоритет операцій визначає порядок, у якому виконуються операції у виразі. У табл. 2.4 перераховані розглянуті операції у порядку зменшення пріоритетів.

Таблиця 2.4

Назва	**Позначення**	**Назва**	**Позначення**
Інкремент	++	Віднімання	-
Декремент	--	Порівняння	<, >, <=, >=
Заперечення	!	Рівність	==
Унарний мінус	-	Нерівність	!=
Множення	*	Логічне	I&&
Ділення, залишок від ділення	/,%	Логічне АБО	//
Додавання	+	Присвоєння	=, +=, -=, *=, /=, %=, !=

При інтерпретації HTML-сторінки браузером створюються об'єкти JavaScript. Взаємозв'язок об'єктів є ієрархічною структурою. На самому верхньому рівні ієрархії розташований об'єкт *window*, що являє собою вікно браузера. Об'єкт *window* є предком всієї решти об'єктів. Кожна сторінка, окрім об'єкта *window*, має об'єкт *document*. Властивості об'єкта *document* визначаються вмістом самого документа: колір тла, колір шрифту і т.д. На останньому місці на сторінці розташована форма. Форма *(form)* є нащадком об'єкта *document*, а всі елементи форми є нащадками об'єкту *<form>*. Посилання на об'єкт може бути здійснене по імені, заданому параметром *name* тега мови HTML. Для отримання значення, введеного користувачем у першому полі форми, повинна бути виконана конструкція *document.form1.num.value.*

При посиланні на форми і їх елементи можна не указувати *document*, тому набути значення, введеного в перше поле форми, можна і таким чином: *form1.num.value.*

Щоб обчислене значення потрапило в друге текстове поле, досить змінити значення *document.form1.res.value*.

Приклад 2.6

```
<html>
<head><title>Обчислення премії</title></head>
<body>
<h3>Обчислення розмірів премії</h3>
<form name="form1">
<p>Введіть розмір доходу і натисніть кнопку "Обчислити"</p>
Дохід: <input type="text" name="num" size="10">
<br><br>
<input type="button" value="Вирахувати"
onClick="document.form1.res.value=0.25*document.form1.num.value">
<br><br>
Премія:<input type="text" name="res" size="10">
<br><br>
```

```
<input type="reset" value="Очистити">
</form></body></html>
```

При створенні програми слід виділити в ній логічно незалежні частини (так звані підпрограми). Кожну частину за потреби можна розбити на окремі підпрограми. Розбиття програми на підпрограми полегшує процес налагодження, оскільки дає змогу налагодити кожну підпрограму окремо.

Один раз створену і налагоджену програму можна використовувати довільну кількість разів.

У багатьох мовах програмування поняття підпрограми реалізується за допомогою конструкцій процедур, функцій, модулів і т.п.

Основним елементом мови JavaScript є функція. Опис функції має вигляд:

```
function   function_name()
{
Оператори JavaScript
}
```

Зверніть увагу, що при створенні своєї власної функції вам доведеться змінити параметр *function_name*. В імені не повинно бути пропусків. Також зверніть увагу на те, що у функції окремі оператори (блоки операторів) поміщені у фігурні дужки – ***{ і }***. Це дає змогу браузеру розрізняти операторів в сценарії. Нижче наведено приклад функції:

```
 function display_time()
{ var now =  new Date()
var hours  =  now.getHours()
var minutes  =  now.getMinutes()
var  seconds  =  now.getSeconds()
var  current_time  = hours  +  ":"  + minutes  +  ":"  +  seconds
document.write("Поточний час: " + current_time) }
```

Викликати функцію можна декількома способами, найбільш простій – додавання оператора, який не містить нічого, окрім імені функції, поміщеної в круглих дужки: *display_time()*.

У прикладі 2.7 показано код сторінки, на якій розміщений сценарій. Цей сценарій «запитує» у користувача, чи хоче він дізнатися поточний час. У разі ствердної відповіді при натисненні кнопки «Час» викликається функція *display_time ()*.

Приклад 2.7.

```
<html>
<head>
<title>Виклик функції. Time()</title>
<script language="JavaScript" type="text/javascript">
function display_time() {
var now = new Date()
var hours = now.getHours()
var minutes = now.getMinutes()
var seconds = now.getSeconds()
var current_time = hours + ":" + minutes + ":" + seconds
document.write("Поточний час:" + current_time)
}
</script> </head> <body>
<form name=orm1>
Бажаєте дізнатися час?<br> Натисніть кнопку:<br>
<input type=button value="Час" onClick="display_time()">
</form>
</body>
</html>
```

Опис функції не може бути вкладений в опис іншої функції. Параметри функції усередині її тіла відіграють ту саму роль, що і звичайні змінні, але початкові значення цим параметрам присвоюються при зверненні до функції.

Якщо опис функції має вигляд:

```
function   function_name(a1, a2 ., an)
{
Оператори JavaScript
```

}

то виклик функції (у тексті основного *HTML*-коду, наприклад, при події *onClick*) повинен мати такий вигляд:

function_name(e1, e2 ., en)

де *e1, e2 ., en* – вирази, що задають фактичні значення параметрів.

Параметри *a1, a2 ., an,* вказані в описі функції, називаються формальними параметрами, щоб підкреслити той факт, що вони набувають значення лише після завдання у виклику функції фактичних параметрів *e1, e2 ., en*, з якими функція потім і працює.

Зазвичай усі визначення і функції задаються в розділі <*head*> документа. Це забезпечує інтерпретацію і збереження в пам'яті всіх функцій при завантаженні документа в браузер.

Функції можуть передавати сценарій за значенням, приклад 2.8.

Приклад 2.8

```
<html>
<body><head><title>Обчислення площі квадрата</title>
<script>
function care (a)
{return a*a}
</script>
</head>
<body><h4>Обчислення площі квадрата</h4>
<form name="form1">
<p>Введіть довжину сторони квадрата і натисніть кнопку <b>Визначити</b></p>
Сторона: <input type="text" name="num" size="4"><br><br>
<input type="button" value="Визначити"
onClick="document.form1.res.value= care(document.form1.num.value) ">
<br><br>
Площа: <input type="text" name="res" size="8"><br><br>
<input type="reset">
</form>
</body>
</html>
```

При виклику функції формальному параметру *a* присвоюється значення фактичного параметра *document.form1.num.value* і виконується тіло функції, яке в цьому завданні складається з оператора *return,* що видає значення функції. Виклик функції виступає в ролі виразу, який поміщається в полі форми.

Інший спосіб передачі параметрів – передача параметрів за іменем. В цьому разі значення фактичного параметра може бути змінено.

Видаване функцією значення визначається оператором *return*.

Завдання до самостійної роботи

1. Виконайте приклади.
2. За допомогою перших трьох прикладів змініть тло тіла документа <*body*> за допомогою подвійного клацання по ньому.
3. В прикладі 2.6 додайте кнопку «Разом», при клацанні по якій в текстовому полі «Дохід + Премія» виведеться відповідна сума.
4. Для прикладі 2.7 відредагуйте функцію *display_time()* так, щоб після натиснення кнопки «Час» на новій сторінці виводилися час і поточна дата (з нового рядка).
5. Використовуючи приклад 2.8 обчисліть площу прямокутника.

Контрольні запитання та завдання

1. У чому полягає програмування обробників подій?
2. Перерахуйте способи використання обробників подій.
3. Які ви знаєте способи розміщення JavaScript?
4. Які ви знаєте способи застосування функцій в JavaScript?
5. Як розподіляються пріоритети операцій у JavaScript?
6. Які є способи передачі параметрів функцій?

Лабораторна робота 2.2.

Об'єкти вікна браузера. Методи alert, prompt, confirm. Прийоми роботи JavaScript з зображеннями на веб-сторінці. Робота з масивами

Мета: навчитися використовувати об'єкти браузера; опанувати прийоми програмування зображень на веб-сторінках за допомогою JavaScript та застосування масивів.

Вимоги до обладнання та програмного забезпечення

Лабораторна робота виконується на ПК з використанням програм Microsoft Expression Web 4, TopStyle 4 та браузерів Google Chrom, Opera, Mazila Firefox.

Основні теоретичні відомості

У мові JavaScript визначені об'єкти, які називаються об'єктами браузера. Кожен об'єкт відповідає деякому елементу веб-сторінки: вікну, документу, зображенню, посиланню і т. п. Управляти частинами документа можна за допомогою методів браузера. Об'єкти браузера мають ієрархічну структуру. На самому верхньому рівні розташовується об'єкт *window*. Він є батьком інших об'єктів. Об'єкт *window* являє собою вікно браузера. Властивість *window.status* можна використовувати для зміни виду рядка стану.

При виконанні сценарію користувач може створювати нові вікна, завантажувати в них різні документи і т.п. Нове вікно браузера створюється з допомогою методу *open()*. Розглянемо оператор присвоювання, в правій частині якого використовується метод *window.open():*

wint = window.open ("anketa.htm", "mywin"," ")

Змінна *wint* використовується для зберігання даних у новому вікні (об'єкт *window*). За допомогою цієї змінної можна отримати властивості та методи нового створеного вікна. Як перший параметр при створенні нового вікна задається URL-адреса документа, що завантажується в створюване вікно. Якщо значення першого параметра не зазначено, створене вікно буде порожнім.

Другий параметр визначає назву вікна; це значення відповідає властивості *name* об'єкта *window*. У найпростішому випадку третій параметр можна не зазначати, тоді при створенні вікна його параметри використовуватимуть значення за замовчуванням.

Приклад 2.9

```
<html>
<head>
<title>Вікна, що визначаються користувачем</title>
<script language="JavaScript">
<!--
// створення нового вікна
function opw()
{ wint = window.open ("anketa.html") }
//-->
</script>
</head>
<body>
<h4>Виведення на екран діалогових вікон, визначених користувачем</h4>
<form name="form1">
<input type="button" value="Відкрити анкету" onClick="opw()">
<input type="button" value="Закрити анкету" onClick="wint.close()">
</form>
</body>
</html>
```

Для відображення діалогових повідомлень об'єкт *window* має три методи.

Метод *window.alert* відображає діалогове вікно, в яке поміщається повідомлення користувача (приклад 2.9). Цей метод часто використовується при обробці полів форми. Якщо яке-небудь з полів форми введено неправильне значення, користувачеві надсилається таке повідомлення, як показано на (рис. 2.2).

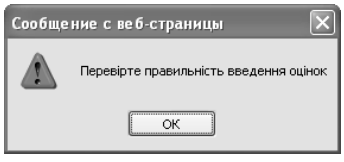


Рис. 2.2. Діалогове вікно методу alert

Приклад 2.10

```
<html>
<head>
<title>Обчислення розмірів стипендії</title>
<script language="JavaScript">
function msgot()
{window.status="Неправільно введені дані";
alert("Перевірте правильність введення оцінок")
window.status=""}
function st(obj)
{var a=Number(obj.num1.value);
var b=Number(obj.num2.value);
var c=Number(obj.num3.value);
var d=Number(obj.num4.value);
if ((a<2)||(a>5)||(b<2)||(b>5)||(c<2)||(c>5)||(d<2)||(d>5))
{msgot()}
else
{ var k; var t;
var m=0;
k=Math.min(Math.min(a,b),Math.min(c,d))
t=Math.max(Math.max(a,b),Math.max(c,d))
if (k==5) {m=100}
 if ((t==5) && (k==4)) {m=75}
if ((t==4) && (k==4)) {m=50}
if (m==0) {obj.stip.value="На жаль, вам стипендія не призначена"}
else {obj.stip.value="Вам призначена стипендія у розмірі "+m+" грн."}
}
}
</script>
</head>
<body>
<h4>Обчислення розмірів стипендії за результатами сесії</h4>
<h4>Введіть отримані оцінки в будь-якому порядку:</h4>
<form name="form1">
Оцінка 1 <input type="text" name="num1" size="3"><br>
Оцінка 2 <input type="text" name="num2" size="3"><br>
Оцінка 3 <input type="text" name="num3" size="3"><br>
Оцінка 4 <input type="text" name="num4" size="3"><br>
<hr>Для визначення розмірів стипендії натисніть кнопку:<hr>
<input type="button" value="Визначити" onclick="st(form1)"><br><hr>
<input type="text" size="50" name="stip"><hr>
<input type="reset" value="Очистити">
</form>
</body>
</html>
```

Метод *prompt* використовується для виклику діалогового вікна запиту даних. При клацанні по кнопці *ok* введені користувачем у текстове поле дані відображаються в документі. Якщо вибрано значення *cancel*, повертається значення *Null*. Метод *prompt* має інший параметр, за допомогою якого задається значення за замовчуванням.

Приклад 2.11

```
<html>
<head>
<title>Метод promt для об’єкта window</title>
<script>
function reg(obj)
{var s=window.prompt("Введіть ваше ім’я","")
```

```
obj.regname.value=s
}
</script>
</head>
<body>
<p align="center"><strong>Метод <em>prompt</em> для об’єкта window</strong></p>
<form name="form1">
<input type="button" value="Реєстрація" onclick="reg(form1)"><br><br>
<input type="text" name="regname">
</form>
</body>
</html>
```

Метод *confirm* відображає діалогове вікно підтвердження виконання операції. Воно містить дві кнопки *“ok”* і *“cancel”,* що дають змогу вибрати один із варіантів. Якщо користувач клацнув по кнопці *ok*, то повертається значення *true*, після клацання по кнопці *cancel* повертається значення *false*. У функції *quest()* прикладу 2.12 аналізується методом *confirm* значення і виконуються відповідні дії.

Приклад 2.12

```
<html>
<head>
<title>Метод confirm для об’єкта window</title>
<script language="JavaScript">
function quest()
 {if (confirm("Хочете закінчити роботу?"))
document.write("Дії при завершенні роботи <br>")
else
document.write("Скасування переходу при завершенні роботи <br>")
}
</script>
</head>
<body>
<p> <strong>Діалог із користувачем</strong><br>
<strong>Метод confirm для об’єкта window </strong></p>
<script>
quest()
</script>
</body>
</html>
```

Зображення на веб-сторінках застосовуються з різною метою. За допомогою схем та малюнків зручно ілюструвати теоретичний матеріал. Іноді зображення краще текстового опису.

JavaScript розроблений із використанням простої об’єктно-орієнтованої парадигми. Об’єкт – це конструкція з властивостями, які є змінними JavaScript. Об’єкт також може мати асоційовані з ним функції, які відомі як методи об’єкта.

Об’єкт JavaScript має асоційовані з ним властивості. Отримати доступ до властивості дуже просто:

Ім’я об’єкта. Ім’я_властивості

Ім’я об’єкта й ім’я властивості чутливі до регістру. Потрібно визначити властивість об’єкта, надаючи йому значення. Наприклад, припустимо, є об’єкт *myCar* (починаючи звідси, приймемо для зручності, що об’єкт вже існує).

У JavaScript можна посилатися на властивості об’єктів, на ім’я властивості або на порядковий індекс. Однак якщо спочатку визначали властивість за іменем, то звертатися до нього треба лише за іменем, а якщо спочатку визначили властивість за індексом, то звертатися до нього треба лише за його індексом.

Це застосовано при створенні об’єкта і його властивостей за допомогою конструктора функції, як у прикладі з типом об’єкта *myCar*, і якщо визначаєте окремі властивості явним чином (наприклад, *myCar.color = "red"*). Так, якщо визначили властивості об’єкта за індексом, як, наприклад, *myCar[0] = "25.jpg"*, потім можете звертатися до властивості лише як *myCar[0].*

Винятком цього правила є об’єкти, відображені в *HTML*, такі як масиви форм. До цих об’єктів завжди звертаються за порядковим номером (що залежить від місцезнаходження об’єкта в документі) або за їх

іменами (якщо вони визначені). Наприклад, якщо другий елемент розмітки *form* у документі має в атрибуті *name* значення *"myForm",* Можна посилатися на форму *document.forms[1],* або *document.forms["myForm",]* або *document.myForm*.

Якщо форма містить кілька елементів – текстові поля і кнопки, то властивість *elements* форми зберігає інформацію про всі її елементи в тому порядку, в якому вони зустрічаються в *HTML*-документі Отримати доступ до об'єктів форми можна, скориставшись властивістю *document.forms[0].elements[0].value***.**

При вирішенні завдань нове зображення завантажується на сторінку після того, як змінилося значення параметра *src*. Завантаження зображення з мережі може зайняти деякий час, іноді досить значний.

Якщо зображень на сторінці багато або вони мають великий розмір, доцільніше скористатися технікою попереднього завантаження зображення. Попередньою завантаженні зображення браузер «підготує», вони потім знадобляться для розміщення на сторінці, тобто завчасно завантажаться в пам'ять комп'ютера, не чекаючи безпосереднього звернення до цих зображень.

Подбати про попереднє завантаження зображень доведеться самому користувачеві, написавши відповідні рядки сценарію.

Приклад 2.13

```
<html>
<head>
<title>Зміна розмірів зображення</title>
<script language="JavaScript">
function big()
{
document.ris.width=400
document. ris.height=500
document. ris.border=7
}
</script>
</head>
<body bgcolor="#088880">
<center>
<img src="cat.jpg" name="ris" border=2>
<h3>Вбудовані зображення</h3>
<form name="form1">
Для зміни розмірів малюнка натисніть кнопку
<b>Збільшити</b><br>
<input type="button" value="Збільшити" onClick="big()">
<input type="reset" value="Відновити onClick="little()">
</form>
</center>
</body>
</html>
```

Оскільки масив являє собою різновид змінної, перш ніж його використовувати, масив слід оголосити. Для оголошення масиву використовується ключове слово *Var*, але синтаксис його використання дещо інший:

Var Array_Name = new Array()

Array_Name – Ім'я, яке використовується у ролі змінної

Е JavaScript масив оголошується об'єктом, тому для створення нового об'єкта *Array* використовується ключове слово *new*. Такий оператор, як *Array(),* називається конструктором, оскільки дозволяє сконструювати об'єкт у пам'яті комп'ютера.

Наприклад, для створення нового масиву *imgslide* слід використовувати оператор:

Var imgslide = new Array()

Якщо відомо кількість елементів, поміщених до масива, його можна задати:

Var Array_Name = new Array(Namber_of_Values)

Array_Name – використовується у ролі змінної

Namber_of_Values – кількість елементів, які будуть поміщені в масив.

Наприклад, для створення нового масиву *imgslide*, що містить чотири елементи, потрібно використовувати такий оператор:

var imgslide = new Array(4)

Щоб попередньо завантажити зображення, створимо масив *imgslide: imgslide = new Array()*. Кожен елемент масиву являє собою об'єкт *Image*, який створюється за допомогою конструктора *new Image ()*. Перший елемент масиву визначається так: *imgslide[0] = new Image()*. Властивість *src* щойно створеного об'єкта визначається в результаті виконання оператора присвоювання:

imgslide[0].src="ris1.jpg"

Аналогічно формуються інші елементи масиву. При роботі сценарію відбувається звернення до елементів масиву, і додаткове завантаження зображень уже не буде потрібно.

Приклад 2.14

```
<html>
<head>
<title>Зміна зображень із попереднім завантаженням</title>
</head>
<script language="JavaScript">
//попереднє завантаження зображень
numimg=0
imgslide=new Array()
imgslide[0]=new Image()
imgslide[1]=new Image()
imgslide[2]=new Image()
imgslide[3]=new Image()
imgslide[0].src="ris1.jpg"
imgslide[1].src="ris2.jpg"
imgslide[2].src="ris3.jpg"
imgslide[3].src="ris4.jpg"

//зміна зображень
function demoslides()
{document.images[0].src=imgslide[numimg].src
numimg++
if (numimg==4)
numimg=0;
setTimeout ("demoslides()", 1000)
}
</script>
<body bgcolor="#00ffff" onLoad ="demoslides()">
<center><img src="romashka.jpg" width="200" height="200" border="1"></center>
</body>
</html>
```

Завдання до самостійної роботи

1. Відредагуйте приклад 2.8 так, щоб результат виконання сценарію виводився в діалогове вікно, а не в текстове поле, а в рядку стану виводилося повідомлення: «Середній бал: ... (результат)».
2. Відредагуйте приклад 2.10 так, щоб форма мала три радіокнопки (перемикачі) – 401, 402, 406. Після вибору номера групи і після натискання кнопки «Реєстрація», де вводиться ім'я користувача, повинно, з'явитися вікно з повідомленням: «Ви ввели... (введене ім'я)», а після натискання кнопки «ОК» в текстовому полі з'явиться повідомлення «Група…(введена група)».
3. Відредагуйте приклад 2.12 так, щоб:
   а) після натискання кнопки «Відновити» запускалася функція *little()* для відновлення розмірів зображення;
   б) в приклад вставити ще одне зображення;
   в) звернення до зображень виконувалося згідно властивостей images об'єкта *document: document.images[0]*. Доступ до різних властивостей зображення отримати з допомогою властивостей самого зображення, наприклад, *document.images[0].width;*
   г) при клацанні мишею по другому зображенню воно збільшувалося, а після подвійного клацання його розміри відновлювалися;
   д) у зображень відображалася кольорова рамка після їх збільшення. Використовувати стилі *CSS*.

4. Створіть веб-сторінку з чотирма маленькими зображеннями, текстовим полем для введення номера зображення і дві кнопки «Відкрити» і «Закрити». Після введення номера та натискання кнопки «Відкрити» відкривається нове вікно з заданими розмірами зі збільшеним даним зображенням. При введенні неправильного номери зображення виводиться повідомлення про помилку. Щоб відкрити вікно використовуйте команду: *window.open('ris'+z+'.jpg','','width=400, height=300')*, де *z* – номер зображення.

Контрольні запитання та завдання

1. Які ви знаєте об'єкти браузера в JavaScript?
2. Які є методи створювання текстових вікон?
3. Назвіть особливості застосування методу *Confirm*.
4. Назвіть особливості роботи з зображеннями в JavaScript?
5. Які методи застосування масивів в JavaScript ви знаєте?

Як можна отримати доступ до об'єктів масив