

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НУ «Запорізька політехніка»

Кафедра ПЗ

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни
“Інженерія вебзастосунків”
для студентів спеціальності
121 “Інженерія програмного забезпечення”

2023

Методичні вказівки до виконання лабораторних робіт з дисципліни “Інженерія вебзастосунків” для студентів спеціальності 121 “Інженерія програмного забезпечення” / Укл.: О.О. Степаненко, Є.М. Федорченко – Запоріжжя: НУ «Запорізька політехніка», 2023. – 63 с.

Укладачі: О.О. Степаненко, к.т.н., доцент кафедри ПЗ,
Є.М. Федорченко, ст. викладач каф. ПЗ.

Рецензент: А.В. Пархоменко, к.т.н., доцент кафедри ПЗ.

Відповідальний
за випуск: С.О. Субботін, зав. каф. ПЗ, д.т.н., проф.

Затверджено
на засіданні кафедри
“Програмних засобів”

Протокол № 12
від “9” червня 2023 р

.

ЗМІСТ

Вступ.....	5
1 Лабораторна робота 1. Загальна структура HTML-документа. Форматування тексту	6
1.1 Теоретичні відомості	6
1.2 Порядок виконання роботи	12
1.3 Завдання для виконання	14
1.4 Контрольні запитання	14
2 Лабораторна робота 2. Розробка людино-машинного інтерфейсу веб-сайту. Створення гіпертекстових посилань. Впровадження в сторінку графічних елементів	15
2.1 Теоретичні відомості	15
2.2 Порядок виконання лабораторної роботи	18
2.3 Завдання для виконання	19
2.4 Контрольні запитання	19
3 Лабораторна робота 3. Елементи таблиць як приклад інформаційної моделі даних. Видобування, опрацювання та зберігання даних за допомогою таблиць	20
3.1 Теоретичні відомості	20
3.2 Порядок виконання лабораторної роботи	23
3.3 Завдання для виконання	24
3.4 Контрольні запитання	26
4 Лабораторна робота 4. Управління зовнішнім видом веб-сторінки за допомогою каскадних таблиць стилів, як елемент людино-машинного інтерфейсу.....	27
4.1 Теоретичні відомості	27
4.2 Порядок виконання лабораторної роботи	32
4.3 Завдання для виконання	35
4.4 Контрольні запитання	35
5 Лабораторна робота 5. Робота з HTML-формами. Створення контактної форми для сайту.....	36
5.1 Теоретичні відомості	36
5.2 Завдання для виконання	41
5.3 Контрольні запитання	41

**6 Лабораторна робота 6. Основні конструкції JavaScript.
Об'єктна модель документа. Створення функцій на**

JavaScript.....	43
6.1 Теоретичні відомості	43
6.2 Порядок виконання лабораторної роботи	49
6.3 Завдання для виконання	50
6.4 Контрольні запитання	50
Перелік рекомендованої літератури.....	511
Додаток А Властивості форматування блокових	елементів..... 522
Додаток Б Основи програмування на мові JavaScript.....	56
Додаток В Методи та властивості об'єктів документа.....	58

ВСТУП

Професії, що пов'язані з веб-розробкою, з року в рік стають все більш поширеними.

Задачею даної дисципліни є навчання основним навикам, що необхідні для розробки веб-сайтів. Хороший сайт вирізняють:

- висока якість його наповнення;
- оригінальність та естетична привабливість сторінок;
- доступність вмісту сайту для максимально широкого кола користувачів незалежно від типів пристроїв та версій браузерів, що застосовуються;
- ергономічність елементів інтерфейсу користувача сайту.

В результаті вивчення даної дисципліни студенти ознайомляться з основними технологіями, необхідними для реалізації веб-проектів будь-якого призначення.

Зміст звіту з лабораторної роботи повинен включати:

1. Тему роботи, мету та завдання.
2. Структуру html-документа.
3. Зовнішній вигляд html-документа.
4. Відповіді на контрольні запитання.

1 ЛАБОРАТОРНА РОБОТА 1

Загальна структура HTML – документа.

Форматування тексту

Мета роботи: ознайомитися зі загальною структурою HTML-документа, відформатувати тексти, що підготовлені для сайту згідно з поточним стандартом мови HTML.

1.1 Теоретичні відомості

Загальна структура HTML – документа. Як і будь-яка інша мова, HTML припускає деяку стандартизовану структуру побудови програми - у даному випадку, html-документа. Така структура описує не послідовність команд, а черговість проходження ряду обов'язкових блоків, що містять безпосередньо програмний код. На відміну від інших мов, директиви HTML називаються не "командами", "процедурами" або "операторами", а мають власне найменування - "теги" (від англ. tag - оцінка). Теги HTML беруться у кутові дужки, синтаксис їх запису в загальному виді виглядає як <тег>. Всі об'єкти, не взяті в кутові дужки, інтерпретатор сприймає як текстові елементи, відображаючи їх на екрані комп'ютера "як є".

Практично всі теги HTML, за винятком деяких окремих випадків, – парні. Така пара складається з "відкриваючого" та "закриваючого" тегу, що відрізняються лише наявністю в останньому символу "/". Усе, що розташовано між відкриваючим та закриваючим тегом, обробляється інтерпретатором відповідно до алгоритму, що присвоєно даному конкретному тегу. У загальному виді програмний рядок HTML з відкриваючим і закриваючим тегами виглядає так:

<тег> Значення, що обробляється </тег>

Дана властивість HTML дозволяє використати принцип вкладення одного тегу в інший, коли значенням, що обробляється, однієї команди може служити інша команда. Наприклад:

<тег1> <тег2> Значення, що обробляється
</тег2> </тег1>

Тег може мати атрибути. У загальному виді синтаксис запису тегу HTML разом з його атрибутами виглядає таким чином:

```
<тег атрибут1 = "значення" атрибут2 = "значення"> </тег>
```

Структура *html*-документа наведена на рис.1.1.



Рисунок 1.1 – Спрощена структура html-документа

Код, що відповідає структурі документа, що відображена на рисунку 1.1 в загальному випадку має наступний вид:

```

<html>
<head>
    ...
<title>Зовнішній заголовок</title>
    ...
</head>
<body>
    Основний код
</body>
</html>

```

Прямокутнику з написом "Документ HTML" відповідає спеціальна команда, що має "пояснити" браузеру, що він має справу саме з документом HTML, а не з текстовим або, наприклад, графічним файлом. Називається вона "тег верхнього рівня" і записується так:

```

<html xmlns="http://www.w3.org/1999/xhtml">
    Вміст
</html>

```

Атрибутом тегу <html> є параметр xmlns, що задає простір імен XML. Він потрібен для створення документів, сумісних з вимогами XHTML, зокрема, стандарту XHTML 1.0. Тег верхнього рівня, як стає ясно з запропонованого приклада, - парний, причому його вміст саме і є весь код HTML, що складає документ. Таким чином, правило застосування даної директиви також очевидне: відкриваючий тег записується найпершим рядком html-документа, а закриваючий - останнім.

Заголовок web-сторінки містить вичерпну інформацію про сам документ, а іноді також спеціальні директиви транслятора, що підказують вбудованому в браузер інтерпретаторові HTML правила, згідно з якими варто обробляти складаючий сторінку код. Вміст заголовка не відображається в браузері та не впливає на зовнішній вигляд документа: це, якщо можна так сказати, службова інформація, що необхідна, насамперед, самому браузеру. Синтаксис тегу заголовка в загальному виді виглядає так:

```

<head>Вміст</head>

```


Всередині `<head>` можуть розташовуватися різні елементи, серед яких наступні:

- `<title>` - назва документа;
- `<base>` - вихідна URL документа;
- `<meta>` - додаткова інформація про сторінку.

Мнемоніка зовнішнього заголовка записується таким чином:

```
<title>Зовнішній заголовок</title>
```

Саме він відображається у верхньому полі браузера як назва сторінки при її відкритті, і саме значення тегу `<title>` підставляється за замовчуванням у відповідне діалогове вікно, коли користувач заносить документ у папку "обране".

Тіло документа, описуване тегами `<body></body>`, містить у собі весь основний код розмітки сторінки, що і визначає відображення html-документа на екрані монітора. Основний текст, ілюстрації, елементи навігації й усе, що ви хочете продемонструвати відвідувачам вашого сайту, розміщується всередині даного тегу.

Елементи форматування тексту та списків.

Заголовки. Заголовки ранжуються з 1-го по 6-й рівень і дозволяють відображати структуру документа. Для того щоб відобразити заголовок на web-сторінці, необхідно використати тег `<h1></h1>`, де n - ціле число від 1 до 6, що позначає номер рівня заголовка, причому найвищим рівнем прийнято вважати 1. Пропускати рівні не можна. Тобто `<h3>` не може бути після `<h1>`, якщо між ними немає `<h2>`. На практиці іноді застосовують заголовки різних рівнів просто для того, щоб задавати розміри шрифтів. Це неправильно, тому що різні браузери по-різному будуть відображати ці розміри. Набагато краще для зміни розміру шрифту використати таблиці стилів.

Абзаци. Для поділу тексту на абзаци використовується тег `<p></p>`. Веб-браузери за замовчуванням ігнорують роздільники між буквами, що перевищують один пробіл, тому й ознака повернення каретки (відповідна натисканню *Enter*) теж буде проігнорована. Елемент `<p>` повідомляє браузеру про те, що весь текст, вміщений між відкриваючим і закриваючим тегами, – це єдиний абзац. Коли інтерпретатор браузера зустрічає в коді відкриваючий тег `<p>`, він вставляє порожній рядок, позначаючи тим самим початок нового абзацу.

Елементи фізичного форматування. Текст може бути виділений напівжирним, курсивом або підкресленням. Ці елементи є абсолютними, а отже, будь-який графічний веб-браузер зобов'язаний відображати їх однаково. Основні елементи фізичного форматування наведені в таблиці 1.1.

Таблиця 1.1 - Елементи фізичного форматування

Елемент	Призначення
, 	Жирний
<i>, </i>	Курсив
<tt>, </tt>	Моноширинний
<u>, </u>	Підкреслений
<big>, </big>	Збільшений розмір
<small>, </small>	Зменшений розмір
_,	Нижній індекс
[,]	Верхній індекс

Елементи логічного форматування. Логічним є стиль, конкретні параметри якого задаються браузером. Браузер робить текст жирніше, виділяє його курсивом, розфарбовує в зелений колір або вимовляє голосніше (голосові браузери) залежно від можливостей конкретної програми. Будь-який елемент логічного стилю має відкриваючий і закриваючий тег, що формує контейнер для тексту, що виділяється. В таблиці 1.2 наведені елементи логічного форматування.

Що стосується графічних браузерів, то звичайно відповідає курсиву, а - жирному шрифту. Але це не завжди так. Іноді може означати підкреслення, а - виділення яскравістю. У голосових браузерах текст, позначений , вимовляється голосніше, ніж звичайний, а - ще голосніше. Основна ідея полягає просто в тому, що робить текст виділеним, а - посилено виділеним.

Таблиця 1.2 - Елементи логічного форматування

Елемент	Призначення
<code></code> , <code></code>	Виділення
<code></code> , <code></code>	Посилене виділення
<code><cite></code> , <code></cite></code>	Цитата або посилання на зовнішнє джерело
<code><dfn></code> , <code></dfn></code>	Вихідний код програми
<code><samp></code> , <code></samp></code>	Приклад роботи програми, часто відображається так само, як і попередній
<code><kbd></code> , <code></kbd></code>	Текст, що вводиться з клавіатури
<code><var></code> , <code></var></code>	Змінна або значення

Інші стилі звичайно використовуються для деяких специфічних цілей, найчастіше пов'язаних з набором технічної або наукової літератури. Знову ж, браузер, призначений для подібних цілей, дуже грамотно обробляє відповідні теги. Що стосується звичайних браузерів, то вони швидше за все виділять текст курсивом, виведуть його моноширинним шрифтом або взагалі залишать без змін.

Може здатися, що логічні елементи є надлишковими. Насправді це не так, більше того, саме їм і віддають перевагу. Тому що не всі браузери можуть відображати фізичні стилі, наприклад виділення жирним шрифтом. Якщо, наприклад, браузер, вбудований у мобільний телефон, не вмє цього, то він просто проігнорує тег ``. Якщо ж використати логічні елементи, то телефон постарасься виділити це місце в тексті як-небудь по-своєму.

Елементи списків. У списку завжди повинно бути два елементи, один із яких задає тип списку, а інший відповідає за один конкретний пункт. Цими пунктами можуть бути слова, речення, абзаци й інші елементи HTML, наприклад зображення. У більшості списків використовується наступний формат:

```
<тип списку>
<li>Перший пункт</li>
<li>Другий пункт</li>
<li>Третій пункт</li>
</тип списку>
```

Кожен елемент `` - це пункт списку, що завжди пишеться з нового рядка. З чого починається цей рядок, залежить від того, чи упорядкований список, маркірований він або нумерований.

Упорядковані й неупорядковані списки. Найпростіше зрозуміти, що це таке, якщо говорити в термінах "нумерований список" й "маркірований список" (відповідно). Нумерований/упорядкований список задається елементом ``, а маркірований - елементом ``. Але пункти кожного з них позначаються за допомогою ``. При цьому якщо список упорядкований (``), то в початок кожного рядка вставляється його порядковий номер у списку; якщо він неупорядкований (``), то вставляється позначка маркування.

Список визначень. Він містить у собі два рівні елементів: терміни і їхні визначення. Синтаксично такий список складається з наступних елементів: головного контейнера списку `<dl>` (*definition list*), терміна `<dt>` (*definition term*) і визначення `<dd>` (*definition*). `<dt>`. За ідеєю, він повинен міститися на одному рядку, хоча якщо буде потреба браузер перенесе на наступний рядок `<dd>`, тоді це буде цілий текстовий абзац. Більшість графічних браузерів розміщують визначення під терміном з невеликим відступом.

1.2 Порядок виконання роботи

1. *html*-сторінки представляють собою звичайні текстові файли, що містять текст і команди розмітки, розміщення зображень, гіпертексту й мультимедіа на сторінці. Всі ці команди - текстові, вони інтерпретуються браузером. Текстова природа документів HTML означає, що для їхнього написання цілком достатньо звичайного текстового редактора. Для розробки web-сторінок у рамках даного курсу ми будемо користуватися будь-яким текстовим редактором.

1.1 Створіть новий каталог з ім'ям *htdocs*, у якому пізніше будуть розміщуватися файли, що вмішуватимуть код сторінок вашого сайту.

1.2 Створіть новий файл із ім'ям *index.html*. У нього необхідно помістити код головної сторінки сайту.

В початок будь-якого web-документа варто включати *DTD*, тобто визначення типу документа (*document type definition*). *DTD* являє

собою елемент, вставлений в початок сторінки, звідки можна довідатися, які мови й стандарти в ній використовуються.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional //EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd">
```

У наведеному прикладі зазначено *DTD* перехідної версії стандарту XHTML. У ньому браузеру повідомляється, що ви використовуєте специфікацію XHTML 1.0 Transitional. Слідом за атрибутом *PUBLIC* йде ім'я *DTD*, у якому зазначена мова, що використовується (у прикладі - *EN*, англійська). Майте на увазі, що, навіть якщо ви використаєте іншу мову на своїй сторінці, мовою XHTML однаково залишається *EN*. Наступним атрибутом *DOCTYPE* є URL того файлу *DTD*, що підтримується W3C. Елемент *DOCTYPE* є єдиним в XHTML, який пишеться прописними буквами.

2. Після елемента *DOCTYPE* розмістіть на сторінці наступний код:

```
<html>
<head>
  <title></title>
</head>
<body>
  Основний код
</body>
</html>
```

Всередині тегу `<title></title>` помістіть назву, обрану вами для html-сторінки. При виборі назви для сторінки необхідно враховувати наступні вимоги:

- назва документа повинна бути інформативна й коротка, довга назва може виглядати дивно в рядку заголовка вікна браузера й може не поміститися в рядок списку закладок або таблиці "Обране", намагайтеся використати в назві не більше 60 символів;

- уникайте загальних слів, намагайтеся максимально точно повідомити, чому присвячена ваша сторінка, пам'ятайте, що будь-яка назва може бути використана як запис у пошукових системах.

Всередині тегу `<body></body>` помістіть текст, підготовлений для сторінки, й відформатуйте його.

При створенні html-документа у відповідності зі стандартом XHTML 1.0 необхідно дотримуватися наступних вимог:

- якщо десь у тексті програми зустрічається відкриваючий тег, обов'язково повинен бути і закриваючий;
- теги можуть бути вкладеними, але не можуть перетинатися;
- всі елементи повинні записуватися малими літерами (виключення становить тег *DTD*);
- значення всіх атрибутів повинні братися в лапки;
- обов'язковим є наявність в html-документі тегів *DTD* й `<title></title>`.

3. Перевірте результат, відкривши отриману сторінку за допомогою браузера.

1.3 Завдання для виконання

1. Оберіть тему для Вашого сайту.
2. Створіть html-шаблон сторінки, що містять теги, які визначають структуру html-документа.
3. Відформатуйте текст, підготовлений для розміщення на сторінці, відповідно до вимог стандарту XHTML 1.0. Обов'язковим є включення в текст наступних елементів: заголовок, абзац, список, виділений текст.
4. Розробіть додатково 2-3 сторінки таким чином, щоб вони містили всю необхідну інформацію.
5. Оформіть звіт.

1.4 Контрольні запитання

1. Назвіть інструменти для роботи з html-кодом.
2. Наведіть структуру html-документа.
3. Які теги визначають структуру html-документа?
4. Який формат запису тегів?
5. Які правила формування назви сторінки?
6. Які основні елементи форматування тексту?
7. В чому відмінність у відображенні тегів фізичного й логічного форматування?
8. Яких вимог необхідно дотримуватися при створенні html-документа у відповідності зі стандартом XHTML 1.0?
9. Для чого в документі вказується DTD?

2 ЛАБОРАТОРНА РОБОТА 2

Розробка людино-машинного інтерфейсу веб-сайту. Створення гіпертекстових посилань. Впровадження в сторінку графічних елементів.

Мета роботи: навчитися створювати посилання на різні типи документів, використовуючи відносні й абсолютні URL, а також додавати до web-сторінок графічні елементи як основу людино-машинного інтерфейсу.

2.1 Теоретичні відомості

Гіпертекстові посилання. Відносні й абсолютні URL.

Більшість гіпертекстових посилань включаються в HTML-документи за допомогою елемента "якір" (<a>). Він повинен доповнюватися атрибутом href, у якому вказується URL (Uniform Resource Locator), на яку варто перейти по даному посиланню. Сама URL повинна бути взята у лапки, оскільки є значенням атрибута href. Посилання на HTML має наступний формат:

```
<a href="URL">Помістіть сюди посилання</a>
```

URL - уніфікована адреса, що використовується для однозначної ідентифікації будь-яких web-компонентів. URL складається із двох частин: імені протоколу й шляху призначення. Ім'я протоколу говорить про те, з яким типом Інтернет-ресурсу ви збираєтеся мати справу. Найпоширенішим в Інтернеті протоколом є *http*, за допомогою якого запитуються web-документи. Шлях призначення може бути іменем файлу, каталогу або комп'ютера. URL виду <http://www.fakecorp.com/products/index.html> вказує на конкретний документ, а <ftp://ftpnetscape.com> каже браузеру про те, що необхідно використати протокол *FTP* для доступу до комп'ютера з ім'ям [ftp.netscape.com](ftp://ftp.netscape.com).

Розрізняють абсолютні та відносні адреси:

```
http://www.olelondonisp.net/drwatson/index.html  
http://www.olelondonisp.net/drwatson/resume.html
```

Обидві URL є абсолютними – з їх допомогою можна потрапити на ці сторінки з будь-якого місця Інтернету. Якщо ж створити посилання на сторінці `index.html`, що повинна вказувати на `resume.html`, можна використати відносну URL, оскільки більша частина адреси нічим не відрізняється. Ця відносність, означає те, що можна одержати доступ і до інших каталогів того ж сервера, використовуючи загальну нотацію. Наприклад, якщо потрібно потрапити на сторінку, розташовану в підкаталозі відносно поточного, можна скористатися URL виду `/pages/house.html` або виду `/assistances/roger/resume.html`. У такий же спосіб можна переходити й до елементів, розташованих у батьківських каталогах. Наприклад, на сторінці, що зберігається в каталозі *products*, потрібно розмістити посилання на головну сторінку, розташовану в кореновому каталозі даного сайту. Це робиться таким чином:

```
.. /index.html
```

Дві крапки на початку адреси - це стандартний синтаксис для позначення переходу на один каталог "нагору" по ієрархії.

Правила створення посилань:

- опис посилання повинен бути інформативним, посилання, на яких підписано "натисніть тут", не дають достатньої інформації про те, куди саме користувач потрапить, якщо він дійсно виконає запропоновану дію;

- необхідно помістити хоч що-небудь всередину якоря, нехай це буде текст або зображення, у протилежному випадку користувачеві не буде на чому клікати, щоб перейти по посиланню;

- не можна створювати вкладені посилання, перш ніж створювати нове, потрібно закрити попереднє;

- можна створювати посилання не тільки на HTML-документи, але й на будь-які інші файли, якщо їх тип підтримується браузером або операційна система знає, що з ним робити.

Посилання на елементи поточної сторінки. Якщо є великий HTML-документ з багатьма розділами, то може виникнути необхідність створення посилання, клікнувши на якому користувач зміг би перейти на один з них. Для цього потрібно спочатку привласнити імена тим частинам, на які необхідно посилатися, а потім встановити якоря, що вказують на них.

У місці, де починається потрібний розділ, необхідно використати якір з атрибутом *id*:

```
<a id="dl"><h1>Розділ 1</h1></a>
<p>Зміст розділу 1</p>
```

Значення *id* можуть містити й букви й цифри, однак вони повинні починатися з букви, а далі вже можуть бути дефіси, підкреслення, двокрапки й крапки. Так, 12 не буде сприйматися коректно, а *question: one* або *myitem56* - буде. Тепер потрібно поставити посилання на цей розділ. Для цього використовується якір, що вказує не на URL, а на ім'я розділу, перед цим ім'ям необхідно використати знак #:

```
<p>Дивіться<a href="#dl">Розділ 1</a></p>
```

На іменовані розділи можна посилатися і з інших сторінок. Просто ім'я розділу стає частиною URL, що визначає розміщення сторінки:

```
<p>Зверніть увагу на
<a href =
"http://www.fakecorp.com/questions.html#dl">
Розділ 1</a></p>
```

При цьому браузер шукає сторінку questions.html, на ній шукає розділ з назвою dl і виводить його.

Створення посилання на e-mail здійснюється наступним чином:

```
<a href="mailto:my@mail.com">Надсилайте ваші
питання прямо на e-mail.</a>
```

Деякі браузери дозволяють автоматично заповнювати поле «Тема листів». Робиться це за допомогою атрибута *title* в складі елемента «якір»:

```
<a href="mailto:my@mail.com" title="Хочу купити
книгу!">Заказати книгу</a>
```

Елемент ``. Стандартним способом додавання зображень є використання елемента ``. За допомогою елемента `` резервується місце на сторінці під зображення. Базова конструкція повинна виглядати таким чином: `. Атрибут `src` розшифровується як `source` (джерело) і містить адресу графічного файлу. Тобто замість слів `image_URL` повинна бути реальна URL. Адреса файлу може бути як абсолютною (наприклад, `http://www.fakecorp.com/images/product1.jpg`), так і відносною (якщо, наприклад, файл розташований на тому ж сервері, що й HTML-сторінка). Варто пам'ятати, що не можна розміщувати на веб-сторінці посилання на файли, що розташовані на локальному жорсткому диску. Якщо створити посилання виду ``, то користувачі не зможуть побачити зазначене зображення.

HTML дозволяє додавати альтернативний текст, він задається за допомогою атрибута `alt`, що є обов'язковим атрибутом елемента ``. У ньому міститься рядок тексту, що замінює зображення у випадку неможливості його виводу на екран через якісь причини. Найчастіше цей текст виводиться в окремому прямокутнику, що окреслює те місце, де повинне бути зображення.

Наявність атрибута `alt` вважається гарним тоном, оскільки це показує, що ви піклуєтесь про свою аудиторію. Альтернативний текст повинен бути коротким, але дійсно інформативним. Якщо ви бажаєте як можна повніше описати відсутнє зображення, це можна зробити таким чином. Помістіть в елемент `` атрибут `longdesc`. Цей атрибут дозволяє розмістити посилання на URL, де може розташовуватися хоч ціла стаття, присвячена даному малюнку.

```

```

Обов'язковими атрибутами елемента `` є також: `width` й `height`, їх значення задаються в пікселях і визначають ширину й висоту зображення.

2.2 Порядок виконання лабораторної роботи

1. Додайте до сторінки `index.html`, що була розроблена у попередній роботі переходи на інші сторінки, що містять текстову та графічну інформацію. Наприклад:

```
<a href="http:// file.html"> file.html</a>
<a href="subdir/file.zip">Download</a>
```

2. Розробіть структуру Ваших сторінок та додайте навігацію між ними та всередині кожної сторінки. Наприклад:

```
<a href="#paragraph4">Перехід на параграф №4</a>
```

```
.....
```

```
<a name="paragraph4">Параграф</a>
```

3. Додайте посилання на e-mail.

```
<a href="http://zntu.edu.ua">zntu.edu.ua</a>
```

4. Створіть сторінки, що містять графічний матеріал.

```

```

5. Додайте навігацію до розробленої сторінки.

2.3 Завдання для виконання

1. Розробити 2-3 сторінки з відповідною структурою та фотокартками, що ілюструють розділи сторінки.

2. Створіть сторінку index.html, що дозволяє переходити на сторінки, створені в лабораторній роботі 1.

3. Оформити звіт.

4. Відповісти на контрольні запитання.

2.4 Контрольні запитання

1. В чому особливість адресації в Інтернеті?

2. В чому різниця між відносними та абсолютними адресами?

3. Як додати посилання на елементи поточної сторінки?

4. Які атрибути елемента `` використовуються для задання інформації про зображення?

5. Які основні правила створення посилань?

6. Які класи мереж ви знаєте?

7. З чого складається IP-адреса?

8. Для чого використовується домена система імен (DNS)?

9. Як прискорити завантаження html-сторінок, що містять багато малюнків?

10. Що являє собою людино-машинний інтерфейс веб-сайту?

3 ЛАБОРАТОРНА РОБОТА 3

Елементи таблиць як приклад інформаційної моделі даних. Видобування, опрацювання та зберігання даних за допомогою таблиць

Мета роботи: навчитися створювати таблиці та розміщувати різні елементи на сторінці за допомогою таблиць, як основного елементу моделі даних, освоїти зчитування\зберігання даних у таблицях.

3.1 Теоретичні відомості

Прикладом основної моделі даних при роботі з HTML є таблиця. Контейнерний елемент *table* використовується для того, щоб зводити разом групи інших елементів, що визначають рядки, а усередині кожного рядка - ячейку. Робота з таблицями в XHTML дуже нагадує роботу з додатками для створення електронних таблиць. Таблиця XHTML складається з рядків і стовпців. Місце перетину конкретного рядка з конкретним стовпцем називається коміркою. Кожна окрема комірка містить дані. Взагалі, комірки можуть вміщувати все що завгодно – чи то текст, чи то інші елементи XHTML, наприклад, гіперпосилання й зображення. Для створення таблиці буде потрібно написати наступний код:

```
<table>
<tr>

  <th>Заголовок стовпця 1</th>
  <th>Заголовок стовпця 2</th>
  <th>Заголовок стовпця 3</th>

</tr>
<tr>

  <td> Ячейка 1-1</td>
  <td> Ячейка 1-2</td>
  <td> Ячейка 1-3</td>

</tr>
<tr>
```

```

<td> Ячейка 2-1</td>
<td> Ячейка 2-2</td>
<td> Ячейка 2-3</td>

</tr>
</table>

```

Головним тут є контейнер `<table></table>`. Усередині `<table>` існують такі елементи: контейнер `<tr>` - задає рядок; усередині його можуть бути елементи `<td>`, що задають комірку. Більшість таблиць може мати також елементи `<th>` - вони дозволяють задавати заголовки рядків або стовпців. Таблиця, код якої наведений вище, відобразиться в браузері як показано на рисунку 3.1.

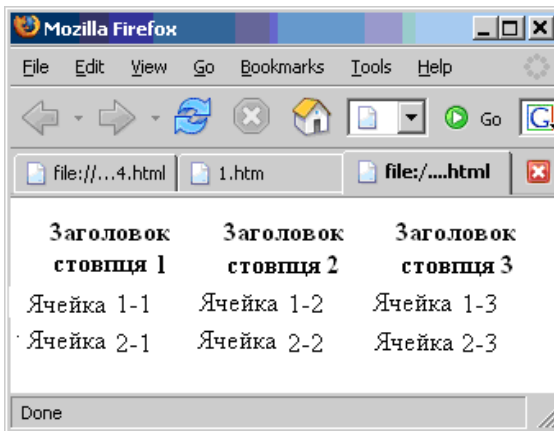


Рисунок 3.1 – Відображення таблиці у вікні браузера

До параметрів комірки належать параметри, значення яких задається атрибутами `colspan=""` та `rowspan=""`.

`colspan` установлює число комірок, які повинні бути об'єднані по горизонталі. Цей параметр має сенс для таблиць, що складаються з декількох стовпців. Наприклад, як для таблиці 3.1, де використовується горизонтальне об'єднання комірок. Дана таблиця вміщує два рядки й два стовпця, причому нижні горизонтальні комірки об'єднані за допомогою параметра `colspan`.

Таблиця 3.1 – Приклад горизонтального об'єднання ячеек

A	B
C	

Код, що відповідає таблиці 3.1, буде мати наступний вид:

```
<table>
<tr>
    <td>A</td>
    <td>B</td>
</tr>
<tr>
    <td colspan="2">C</td>
</tr>
</table>
```

rowspan встановлює число комірок, які повинні бути об'єднані по вертикалі. Цей параметр має сенс для таблиць, що складаються з декількох рядків. Наприклад, як для таблиці 3.2, де застосовується вертикальне об'єднання осередків.

Таблиця 3.2– Приклад вертикального об'єднання ячеек

A	B
	C

Код, що відповідає таблиці 3.2, буде мати наступний вид:

```
<table>
<tr>
    <td rowspan="2">A</td>
    <td>B</td>
</tr>
<tr>
    <td>C</td>
</tr>
</table>
```

Керування комірками таблиць дозволяє розміщувати будь-які елементи – зображення, текст, списки й посилання – там, де потрібно. Це робиться шляхом відповідного розміщення комірок, їхнім розтягуванням або об'єднанням, заданням необхідних внутрішніх

відстаней і відстаней між ними. Можна застосовувати вкладені таблиці, що дозволяють всередині ячейки однієї таблиці помістити іншу таблицю. Тоб-то, таблиця є універсальною формою зберігання даних у структурах (моделях) даних.

3.2 Порядок виконання лабораторної роботи

1. Розробіть модель даних (структуру) для Ваших сторінок відповідно до рисунку. Створіть необхідну навігацію.

Заголовок					
Меню 1	Меню 2			
Меню 1	Основне наповнення сторінки			Новини	
Меню 2					
.....					
@ copywrite					

Рисунок 3.2 – Структура web-сторінки

Наприклад:

```
<table width="100%" border="1">
<thead><th colspan="3">Header</th></thead>
<tr><td>Menu 1</td><td>Menu 2</td><td>Menu 3</td>
</tr>
<tr>
<td>Menu
<table>
<tr><td>Menu 1</td></tr>
<tr><td>Menu 2</td></tr>
<tr><td>Menu 3</td></tr>
</table>
</td>
<td>Content</td>
<td>News</td>
</tr>
<tr>
<td colspan="3">&copy; Copyright</td></tr>
</table>
```

2. Оформіть сторінки, що були розроблені у попередніх лабораторних роботах відповідно до розробленої структури.


3. Розробіть додатково сторінку з резюме відповідно до завдання. Ця сторінка повинна мати навігацію по пунктах Вашого резюме.

3.3 Завдання для виконання

1. Розробіть загальний інтерфейс Ваших сторінок, що були розроблені у лабораторних роботах №1 та №2 з використанням табличного дизайну.

2. Використовуючи знання, отримані про таблиці, а також навички набуті в ході виконання лабораторних робіт №1 й №2, підготуйте html-сторінку, що містить ваше резюме в наступному форматі:

Резюме

 <p>фото</p>	ПІБ Дата народження: Адреса: Контактний телефон: E-mail: Сімейний стан: Бажана посада:
--	--

Освіта:

з (мм/рррр)	по (мм/рррр)	назва навчального закладу	спеціальність	підсумкова атестація

Досвід роботи:

з (мм/рррр)	по (мм/рррр)	місце роботи	посада	обов'язки

Проекти:

Назва	Доля участі	Опис

Технічні навички	Рівень знань				
	не знаю	базовий	середній	високий	експерт
Операційні системи					

Бази даних					

Програми для роботи з графікою					
Adobe Photoshop					
Adobe Illustrator					
Macromedia Flash					

Офісні пакети					
Мови програмування і розмітки					
HTML					
CSS					
DHTML					
XML					
JavaScript					
PHP					
Perl					
C/C++					
ActionScript					
Іноземні мови					
Англійська					
Додаткові навички та вміння					

Додаткова інформація	
Сертифікати	
Особисті якості	
Графік роботи, якому віддана перевага	

3. Оформіть звіт.

4. Дайте відповіді на контрольні запитання.

3.4 Контрольні запитання

1. Поняття таблиці в XHTML. Формат коду для створення таблиці.

2. Атрибути елемента `<td></td>`. Як виконується об'єднання елементів таблиці?

3. Робота з комірками таблиці, як з основними компонентами моделі даних веб-сторінки. Опрацювання даних в таблиці.

4 ЛАБОРАТОРНА РОБОТА 4

Управління зовнішнім видом веб-сторінки за допомогою каскадних таблиць стилів, як елемент людино-машинного інтерфейсу.

Мета роботи: навчитися управляти зовнішнім видом веб-сторінки за допомогою каскадних таблиць стилів, як основним елементом людино-машинного інтерфейсу.

4.1 Теоретичні відомості

Каскадні таблиці стилів (css - *cascading style sheets*) - це набір параметрів форматування, що застосовується до елементів веб-сторінки для керування їх виглядом і положенням. Стили є зручним, практичним й ефективним інструментом при розмітці веб-сторінок й оформленні тексту, посилань, зображень й інших елементів, а саме вони є основними елементами людино-машинної взаємодії.

Підключення CSS. Для додавання стилів на веб-сторінку існує кілька способів, які відрізняються своїми можливостями й призначенням.

Таблиця зв'язаних стилів. При використанні таблиці зв'язаних стилів опис селекторів та їхніх властивостей міститься в окремому файлі, як правило, з розширенням `css`, а для зв'язування документа із цим файлом застосовується тег `<link/>`. Даний тег поміщується в контейнер `<head></head>`.

```
<link rel="stylesheet" type="text/css" href =  
"mysite.css"/>
```

Значення параметрів тегу `<link/>` - *rel* й *type* залишаються незмінними, як наведено в даному прикладі. Аргумент *href* задає шлях до CSS-файлу, він може бути заданий як відносно, так і абсолютно. Зверніть увагу, що таким чином можна підключати таблицю стилів, що знаходиться на іншому сайті.

Використання таблиці зв'язаних стилів є найбільш універсальним і зручним методом додавання стилю на сайт. Адже стилі зберігаються в одному файлі, а в HTML-документах вказується тільки посилання на нього.

Таблиця глобальних стилів. При використанні таблиці глобальних стилів властивості CSS описуються в самому документі й

звичайно розташовуються в заголовку веб-сторінки. За своєю гнучкістю й можливостями цей спосіб додавання стилю поступається попередньому, але також дозволяє розміщувати всі стилі в одному місці. У цьому випадку, прямо в тілі документа, за допомогою контейнера `<style></style>`.

```
<style type="text/css">
...
</style>
```

В даному прикладі визначено стиль, що потім можна повсюдно використати на даній веб-сторінці. Таблиця глобальних стилів може розміщуватися не тільки всередині контейнера `<head></head>`, але також у будь-якому місці коду HTML-документа.

Внутрішні стилі. Внутрішній стиль є по суті розширенням для одиночного тегу, що використовується на веб-сторінці. Для визначення стилю використовується атрибут тегу `style`, а його значення вказуються за допомогою мови таблиці стилів.

```
<h1 style="font-size: 120%; font-family: Verdana,
Arial, Helvetica, sans-serif; color: #336"> Заголовок
</h1>
```

В даному прикладі стиль тегу `<h1>` задається за допомогою параметра `style`, у якому через крапку з комою перераховуються стильові атрибути. Внутрішні стилі рекомендується застосовувати на сайті обмежено або взагалі відмовитися від їхнього використання. Справа в тому, що додавання таких стилів збільшує загальний обсяг файлів, що веде до підвищення часу їхнього завантаження в браузері, і ускладнює редагування документів для розроблювачів.

Всі описані методи використання CSS можуть застосовуватися як самостійно, так і у поєднанні один з одним.

Базовий синтаксис. Спосіб запису CSS відрізняється від форми використання тегів HTML й у загальному виді має наступний синтаксис.

```
селектор {властивість1: значення; властивість2:
значення; ...}
```

Селектором називається ім'я стилю, в якому зазначені параметри форматування. Селектори поділяються на декілька типів: селектори тегів, ідентифікатори й класи. Після вказівки селектора

йдуть фігурні дужки, у яких записується необхідна стильова властивість, а її значення вказується після двокрапки. Параметри розділяються між собою крапкою з комою, в кінці цей символ можна опустити. CSS не чутливий до регістру, переносу рядків, пробілів і символів табуляції, тому форма запису залежить від бажання розроблювача. Імена селекторів обов'язково повинні починатися з латинського символу (a-z, A-Z) і можуть містити в собі цифри.

Селектори тегів. Як селектор може виступати будь-який тег HTML, для якого визначаються правила форматування, такі як: колір, фон, розмір і т.д. Правила задаються в наступному виді:

```
тег {властивість1:значення; властивість2:значення;}
```

```
Наприклад: p {text-align:justify; color:green;}
```

В даному прикладі змінюється колір тексту й вирівнювання тексту параграфа. Стиль буде застосовуватися тільки до тексту всередині контейнера <p></p>.

Класи. Класи застосовують, коли необхідно визначити стиль для індивідуального елемента веб-сторінки або задати різні стилі для одного тегу. При використанні разом з тегами синтаксис для класів буде наступний.

```
тег.ім'я класу {властивість1: значення;  
властивість2: значення;}
```

Всередині стильової таблиці спочатку пишеться бажаний тег, а потім, через крапку користувальницьке ім'я класу. Щоб указати в коді HTML, що тег використовується з певним стилем, до тегу додається параметр class="ім'я класу".

Імена класів вибираються за бажанням, головне, щоб вони були зрозумілі й відповідали їхньому використанню. Можна, також, використати класи і без вказівки тегу. Синтаксис у цьому випадку буде наступний.

```
.ім'я класу {властивість1: значення; властивість2:  
значення;}
```

Класи зручно використовувати, коли потрібно застосувати стиль до різних тегів веб-сторінки: ячеек таблиці, посилань, параграфів.

Ідентифікатори. Ідентифікатор (що називається також "ID селектор") визначає унікальне ім'я елемента, який використовується для зміни його стилю й звертання до нього через скрипти, що дозволяє управляти стилем елемента динамічно.

Синтаксис використання ідентифікатора наступний.

```
#ім'я ідентифікатора {властивість1: значення;  
властивість2: значення;}
```

Звертання до ідентифікатора відбувається аналогічно класам, але як ключове слово в тегу використовується параметр *id*, значенням якого виступає ім'я ідентифікатора. Символ дієх при цьому вже не вказується. Як і при використанні класів, ідентифікатори можна застосовувати до конкретного тегу. Синтаксис при цьому буде наступний.

```
тег#ім'я ідентифікатора {властивість1: значення;  
властивість2: значення;}
```

Коментарі. Щоб позначити, що текст є коментарем, застосовують конструкцію `/* ... */` Для створення однорядкового коментаря застосовується подвійна коса риса (`//`). Усе, що перебуває після цих символів браузером ігнорується й розцінюється як коментар. Перенос рядка в цьому випадку не допускається.

Контекстні селектори. При створенні веб-сторінки часто доводиться вкладати одні теги всередину інших. Щоб стилі для цих тегів використовувалися коректно, допоможуть селектори, які працюють тільки в певному контексті. Наприклад, задати стиль для тегу `` тільки коли він розташовується всередині контейнера `<p></p>`. Таким чином можна одночасно встановити стиль для окремого тегу, а також для тегу, що перебуває всередині іншого. Контекстний селектор складається із простих селекторів, розділених пробілом. Так, для селектора тегу синтаксис буде наступний:

```
тег1 тег2 {...}
```

В даному випадку стиль буде застосовуватися до *тегу2*, коли він розміщується всередині *тегу1*, як показано нижче:

```
<тег1>  
  <тег2> ... </тег2>  
</тег1>
```

Не обов'язково контекстні селектори містять тільки один вкладений тег. Залежно від ситуації припустимо застосовувати два й більш послідовно вкладених один в інший тегів. Більш широкі можливості контекстні селектори дають при використанні ідентифікаторів і класів. Це дозволяє встановлювати стиль тільки для того елемента, що знаходиться всередині певного класу.

```
a {color:green;}
.menu a {color:navy;}
```

При такому підході легко управляти стилем однакових елементів, як от зображень і посилань, оформлення яких повинне розрізнятися в різних областях веб-сторінки.

Групування. При створенні стилю для сайту, коли одночасно використовується безліч селекторів, можлива поява повторюваних параметрів. Щоб не повторювати двічі ті самі елементи, їх можна згрупувати для зручності подання й скорочення коду. Селектори групуються у вигляді списку тегів, розділених між собою комами. У групу можуть входити не тільки селектори тегів, але також ідентифікатори й класи. Загальний синтаксис наступний.

```
селектор1, селектор2, ... селекторN {Опис правил  
стилю}
```

При такому записі правила стилю застосовуються до всіх селекторів, перерахованих в одній групі.

Блокові й рядкові елементи і їхні властивості.

Існує поняття рядкового (*in-line*) елемента розмітки й блокового (*block*) елемента розмітки. Найпростіше пояснити різницю між блоком і рядковим елементом можна на прикладі. Параграф – це блоковий елемент розмітки. Виділення курсивом – це рядковий елемент розмітки.

За набором атрибутів керування відображенням (атрибути опису стилю) рядкові й блокові елементи відрізняються. Деякі властивості елементів наведені в таблиці А.1 додатка А.

Блокові елементи дозволяють оперувати з елементами в термінах блоків. При цьому блок тексту стає елементом дизайну сторінки з тими ж властивостями, що й картинка, таблиця або прямокутна область додатка. Властивості блокового елемента наведені на рис.4.1.

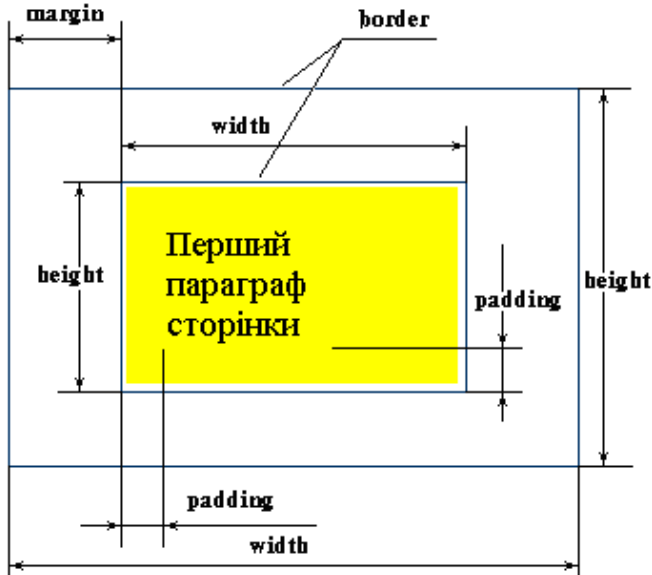


Рисунок 4.1 – Властивості блокового елемента

4.2 Порядок виконання лабораторної роботи

1. Для того, щоб реалізувати структуру документа, запропоновану у попередній лабораторній роботі з використанням блочних елементів, зазвичай застосовують наступну структуру.

```
<div id="header"><p>Заголовок</p></div>
<div id="main">
<div
id="leftmenu"><p>/<strong>Навігація</strong></p>
<ul>
  <li><a href="#">Розділ 1</li>
  .....
  <li><a href="#">Розділ N</li>
</ul>
</div>
<div id="content"><p>Це зміст</p></div>
<div id="rightmenu"><p>/<strong>Новини</strong></p>
  <p>.....</p>
```



```

</div>
</div>
<div id="footer"><p>@zntu2008</p></div>

```

Але ця структура буде відображатися послідовно. Для того, щоб отримати відповідне відображення використовується форматування за допомогою таблиць стилів. Наприклад:

Файл *style.css*

```

body {margin:0px; padding:0px;
      background-color:#f0f0f0;}

#header {
    margin:0px;
    padding:1%;
    text-align:center;
    font:bold xx-large normal;
}

#main {
    margin:0px;
    padding:0px;
    position:relative;
    width:100%;
    overflow:hidden;
    background-color:#ccc;
}

#leftmenu {
    float:left;
    width:18%;
    padding:0px;
    border:thin inset;
}

#content {
    width:57%;
    float:left;
    padding:1%;
}

```

```
#rightmenu {
    position:absolute;
    width:18%;
    padding:1%;
    right:0px;
}

#footer {
    margin:0px;
    padding:1%;
    font-size:x-small;
    clear:both;
}
```

3. Для створення навігації між сторінками сайту за допомогою CSS можна створити просте меню. Для цього в html додаємо список:

```
<div class="nav1">
  <ul>
    <li><a href="#">Розділ1</a></li>
    <li><a href="#">Розділ2</a></li>
    .....
  </ul>
</div>
```

А у файлі css відповідне форматування.

```
.nav1 ul,li {
    float:left;
    list-style:none;
    margin:0;
    padding:0px;
}

.nav1 li {
    padding:2px, 10px;
    font:bold small Tahoma;
    background:#ccc;
    color:white;
    border:solid 1px;
}

.nav1 a {
    color:white;
```

```

        text-decoration:none;
    }
    nav1 li a:hover{
        padding:2px 10px;
        font:thin xx-small Courier
        background:#0f0f0f;
        border:solid 1px;
        color:black;
    }

```

4.3 Завдання для виконання

1. Розробити власний дизайн сайту з використанням блочних елементів.
2. Оформити форматування сторінки в три колонки з використанням CSS.
3. Додати горизонтальне меню з використанням CSS.
4. Додати графічне оформлення сайту (фон та ілюстрації).

4.4 Контрольні запитання

1. Що означає CSS? Коли використовується?
2. Які способи підключення CSS ви знаєте?
3. Базовий синтаксис CSS.
4. Блокові елементи і їхні властивості.
5. В чому різниця при використанні класів та ідентифікаторів?
6. Які параметри форматування блокових елементів вам відомі?
7. Основні елементи людино-машинного інтерфейсу, управління якими може надати CSS.

5 ЛАБОРАТОРНА РОБОТА 5

Робота з HTML-формами. Створення контактної форми для сайту

Мета роботи: ознайомитися з елементами, необхідними для створення форм, створити контактну форму для сайту.

5.1 Теоретичні відомості

Додавання форми на сторінку, параметри тега <form>. Форми є одним з важливих елементів будь-якого сайту й призначені для обміну даними між користувачем і сервером. Область застосування форм не обмежена відправленням даних на сервер, за допомогою клієнтських скриптів можна одержати доступ до будь-якого елемента форми, змінювати його й застосовувати на власний розсуд.

Перед відправленням даних браузер підготовлює інформацію у вигляді пари "ім'я=значення", де ім'я визначається параметром name тегу <input/> або іншим, припустимим у формі, а значення введене користувачем або встановлене в поле форми за замовчуванням. Після натискання користувачем кнопки *submit*, відбувається запуск оброблювача форми, що одержує введену у формі інформацію, а далі робить із нею те, що мав на увазі розроблювач. Як оброблювач форми звичайно виступає CGI-програма, задана параметром *action* тегу <form>.

Для вказівки браузеру де починається й закінчується форма, використовується контейнер <form></form>. Між відкриваючим і закриваючим тегами <form> й </form> можна розміщувати будь-які необхідні теги HTML. Це дозволяє додати елементи форми в ячейки таблиці для їхнього форматування, а також використати зображення. Документ може містити кілька форм, але вони не повинні бути вкладені одна в іншу.

Кожна форма характеризується деякими параметрами, які вказуються в тегу <form>. Ці параметри задають ім'я форми, її оброблювача і метод відправлення даних на сервер, а також деякі інші характеристики.

```
<form action="/cgi-bin/handler.cgi" enctype =  
"multipart/form-data" method="POST">
```

Параметр *action* вказує оброблювач, до якого звертаються дані форми при їхньому відправленні на сервер. Як оброблювач може виступати CGI-програма або HTML-документ, що містить у собі серверні сценарії. Після виконання оброблювачем дій по роботі з даними форми він повертає новий HTML-документ. Якщо параметр *action* відсутній, поточна сторінка перезавантажується, повертаючи всі елементи форми до їхніх значень за замовчуванням.

Параметр *enctype* встановлює тип даних, які відправляють разом з формою. Звичайно встановлювати значення параметра *enctype* не потрібно, дані цілком правильно розуміються на стороні сервера. Однак якщо використовується поле для відправлення файлу (`<input type="file"/>`), варто визначити параметр *enctype* як *multipart/form-data*. Допускається також встановлювати відразу кілька значень, розділяючи їх комами.

Параметр *method* повідомляє серверу про мету запиту. Розрізняють два методи - *get* і *post*. Метод *get* є одним з найпоширеніших і призначений для одержання необхідної інформації й передачі даних в адресному рядку. Пари "ім'я=значення" приєднуються в цьому випадку до адреси після знаку питання й розділяються між собою амперсандом (символ `&`). Зручність використання методу *get* полягає в тому, що адресу з усіма параметрами можна використати неодноразово, зберігши його, наприклад, в "Обране" браузера, а також змінювати значення параметрів просто в адресному рядку.

Метод *post* посилає на сервер дані в запиті браузера. Це дозволяє відправляти більшу кількість даних, ніж доступно методу *get*, оскільки в нього встановлене обмеження в 4 Кб. Більші обсяги даних використовуються у форумах, поштових службах, заповненні бази даних і т.д.

Елементи форм. Форма являє собою лише контейнер для розміщення об'єктів, які дублюють елементи інтерфейсу операційної системи: кнопки, поле зі списком, перемикачі, прапорці і т.д.

Більшість елементів форми створюються за допомогою тегу `<input/>`:

- поле введення (`type="text"`) призначено для введення рядка символів за допомогою клавіатури;

- пароль (`type="password"`) звичайне текстове поле, але відрізняється від нього тим, що всі символи відображаються зірочками;

- прапорець (`type="checkbox"`) використовують, коли необхідно вибрати два або більше варіанти із запропонованого списку;

- перемикач (`type="radio"`) використовують, коли необхідно вибрати один єдиний варіант із декількох запропонованих;

- сховане поле (`type="hidden"`) не показується на сторінці й ховає свій вміст від користувача;

- керуюча кнопка (`type="button"`, `type="reset"` або `type="submit"`) – кнопка, призначена для підтвердження або скасування введення;

- поле із зображенням (`type="image"`) аналогічне по дії кнопці `submit`, але являє собою малюнок;




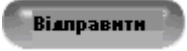

- відправлення файлу (`type="file"`) використовують для того, щоб можна було відправити на сервер файл, такий елемент форми відображається як текстове поле, поруч із яким розташовується кнопка *Browse*, при натисканні на цю кнопку відкривається вікно для вибору файлу, де можна вказати, який файл користувач бажає використати.

У загальному виді формат тегу `<input/>` наступний:

```
<input type="тип" name="ім'я" />
```

Обов'язковими є тільки атрибути *type* й *name*. Крім того, існує ряд додаткових атрибутів, але вони в основному є специфічними для яких-небудь елементів керування. Перелік типів та атрибутів тегу `<input/>` наведений в таблиці 5.1.

Таблиця 5.1 – Типи елемента `<input/>`, їх атрибути та зовнішній вид

Тип	Можливі атрибути	Зовнішній вид
<i>Text</i>	<i>name, value, size, maxlength</i>	
<i>password</i>	<i>name, value, size, maxlength</i>	
<i>checkbox</i>	<i>name, value, checked</i>	<input type="checkbox"/> Опис
<i>Radio</i>	<i>name, value, checked</i>	<input checked="" type="radio"/> Опис
<i>button, reset, submit</i>	<i>Name, value</i>	
<i>Image</i>	як для тегу <code></code>	
<i>File</i>	<i>name, size, maxlength</i>	

Можливі атрибути елемента `<input/>`:

- *type* – визначає тип елемента;
- *size* – ширина текстового поля, що визначається числом символів моноширинного шрифту. Іншими словами, ширина задається кількістю поручстоячих букв однакової ширини по горизонталі.
- *maxlength* – встановлює максимальне число символів, що може бути введено користувачем в текстове поле. Коли ця кількість досягається при наборі, подальше введення стає неможливим. Якщо цей параметр опустити, то можна вводити рядок довше самого поля.
- *name* - ім'я поля, призначене для того, щоб оброблювач форми міг його ідентифікувати.

- *value* – початкове значення поля.

Крім тегу `<input/>` для створення елементів форми використовуються й інші теги.

Багаторядковий текст - це елемент форми, призначений для створення області, в якій можна вводити кілька рядків тексту (рис.5.1).

```
<textarea rows="10" cols="45">текст</textarea>
```



Рисунок 5.1 – Елемент форми `textarea`

Атрибутами багаторядкового тексту можуть бути:

- *cols* – ширина поля в символах;
- *rows* – висота поля в рядках тексту;
- *disabled* – блокує доступ і зміну елемента;
- *readonly* – встановлює, що поле не може змінюватися користувачем;
- *wrap* – параметри переносу рядків;
- *name*;
- *value*.

Поле зі списком, яке ще називають спадаюче меню, один із гнучких і зручних елементів форми. Залежно від настройки, у списку можна вибирати одне або кілька значень. Перевага списку в його компактності, він може займати всього один рядок, а щоб переглянути весь список потрібно на нього натиснути. Однак це є і недоліком, адже користувачеві відразу не видно весь вибір (рис.5.2).

Поле зі списком створюється наступним чином:

```
<select параметри>
<option параметри>Пункт 1</option>
<option>Пункт 2</option>
<option>Пункт 3</option>
</select>
```

Тег `<select>` дозволяє створити елемент інтерфейсу у вигляді списку, що розкривається, а також список з одним або множинним вибором. Кінцевий вид залежить від використання параметра *size* тегу `<select>`, що встановлює висоту списку. Ширина списку визначається

найширшим текстом, що знаходиться у тегу <option>, а також може змінюватися за допомогою стилів. Кожен пункт створюється за допомогою тегу <option>, що повинен бути вкладений у контейнер <select></select>. Розглянемо параметри тегу <select>, за допомогою яких можна змінювати вид і подання списку:

- *multiple* – наявність параметра *multiple* повідомляє браузеру відображати вміст елемента <select></select> як список множинного вибору;

- *name* – визначає унікальне ім'я елемента;

- *size* – встановлює висоту списку.

Тег <option> також має параметри, що впливають на вид списку:

- *selected* – робить пункт списку виділеним. Якщо у тег <select> додано параметр *multiple*, можна виділяти більше одного пункту;

- *value* – визначає значення пункту списку, що буде відправлено на сервер.

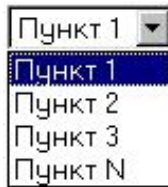


Рисунок 5.2 – Елемент форми *select*

5.2 Завдання для виконання

1. Ознайомитися з усіма елементами форм.
2. Створити html-сторінку з контактною формою, що містить всі необхідні поля. За бажанням можна замінити контактну форму на іншу, що відповідає вимогам (наприклад, форму реєстрації).
3. Відрегулювати зовнішній вигляд форми за допомогою таблиці й стилів.

5.3 Контрольні запитання

1. Для чого використовуються форми?
2. Які атрибути тегу <form>?
3. Які елементи можна створювати з допомогою тегу <input/>? Назвіть їхні атрибути.
4. Елемент *textarea* і його атрибути.

5. Елемент *select* і його атрибути.
6. Яким чином можна регулювати зовнішній вигляд форми?
7. Що ви розумієте під web-клієнтом та web-сервером?
8. Для чого використовуються CGI сценарії?
9. Назвіть відомі вам web-сервери.
10. Які методи передачі http-запитів вам відомі?
11. Намалуйте діаграму взаємодії між клієнтом та сервером.

6 ЛАБОРАТОРНА РОБОТА 6

Основні конструкції *JavaScript*. Об'єктна модель документа. Створення функцій на *JavaScript*

Мета роботи: ознайомитися з синтаксисом мови *JavaScript*, навчитися використовувати функції *JavaScript* у власних документах

6.1 Теоретичні відомості

Мови сценаріїв часто використовуються для створення програм, що працюють як на стороні клієнта, так і на стороні сервера. Сценарії, що вбудовують за допомогою дескрипторів `<script>` `</script>`, виконуються на стороні клієнта під управлінням інтерпретатора. Мови сценаріїв підтримуються багатьма Web-серверами й застосовуються для динамічної генерації HTML коду. Перед передачею браузеру HTML-код оброблюється сервером, і вбудовані сценарії виконуються під керуванням спеціального процесора. Мови сценаріїв також використовуються при розробці CGI програм.

На стороні клієнта сценарії зручно застосовувати для перевірки даних, введених користувачем, і для відображення повідомлень в процесі роботи. При використанні сценарію на стороні сервера часто виникають проблеми, пов'язані з продуктивністю, масштабованістю й підтримкою. Найбільш популярними є мови *JavaScript*, *VBScript*, *Perl*, *PHP*.

Основи мови JavaScript. JavaScript (JS) – це мова сценаріїв, що може виконуватися як на стороні клієнта, так і на стороні сервера. JS був розроблений компанією Netscape. Підтримка JS на стороні клієнта реалізована в Web-браузері Netscape Navigator, а на стороні сервера дана мова реалізована в продукті Netscape Enterprise Server. Продукти Microfot Internet Explorer та Internet Information Server також підтримують JavaScript, але в їх реалізації ця мова називається JScript. На сьогодні JavaScript прийнято як стандарт ECMA (European Community Manufacturers Accociation).

Код скрипта *JavaScript* розміщується безпосередньо на HTML-сторінці.

```
<script language="JavaScript">
    document.write("А це JavaScript!")
</script>
```

Все, що знаходиться між тегами `<script>` й `</script>`, інтерпретується як код мовою *JavaScript*.

Якщо браузер не підтримує *JavaScript*, то можна скористатися тегом коментарю з HTML - `<!-->`

```
<html>
<body>
<br>
Це звичайний HTML документ.
<br>
  <script language="JavaScript">
    <!--ховає код від старих браузерів
      document.write("А це JavaScript!")
    // -->
  </script>
<br>
Знову документ HTML.
</body>
</html>
```

Оператори мови *JavaScript* нагадують оператори мови C++. Основні конструкції мови *JavaScript* наведені в таблиці Б.1 додатка Б.

Об'єктна модель документа. Перед створенням сценаріїв, необхідно розглянути особливості взаємодії сценаріїв й об'єктів документа. Під об'єктами документів розуміються різні елементи HTML сторінки. Перша об'єктна модель документа була запропонована в браузері Netscape Navigator 2. Відтоді й аж до розробки четвертого покоління браузерів вихідна об'єктна модель придбала багато нових характеристик, загальних для браузерів різних типів, а також ряд засобів, які є просто унікальними з погляду перших браузерів як NetscapeNavigator, так і Internet Explorer. Для запобігання багатьох проблем, що виникають, консорціум World Wide Web Consortium (W2C) приступив до розробки стандарту об'єктної моделі документа (DOM-Document Object Model).

Загальна структура об'єктів документів наведена на рис. 6.1.

Програміст повинен знати ієрархічну структуру організації об'єктів в першу чергу тому, що вона застосовується в сценарії при заданні посилань на об'єкти, які знаходяться у вікні браузера.

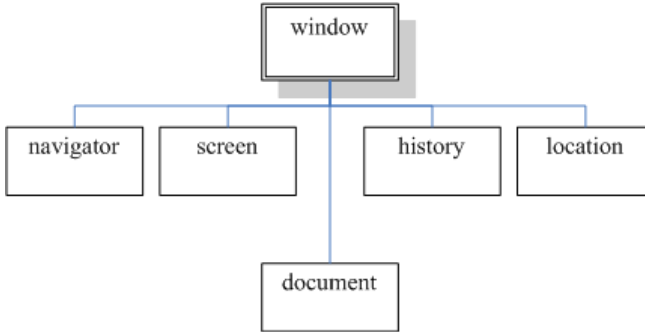


Рисунок 6.1 – Загальна ієрархічна структура об'єктів документа, що використовується браузерами

Оброблювачі подій. Важливою характеристикою об'єктів JavaScript є оброблювачі подій. Подіями називається все, що відбувається в документі. Як правило це результат дій користувачів. Наприклад, клік мишею на кнопці або введення в текстове поле символу.

У документі практично всі об'єкти *JavaScript* одержують повідомлення того або іншого роду. Щоб визначити, як повинен реагувати об'єкт на подію, необхідно ввести додатковий атрибут в опис об'єкта. Цей атрибут складається з імені події, знака рівності, після якого йде інструкція, що вказує на те, що ж потрібно робити при настанні конкретної події. Наприклад:

```
<A HREF="URL" onclick="alert('Тут Ваше повідомлення')"  
LANGUAGE="Javascript">Метод Alert</A>
```

Оброблювачі подій наведені в таблиці В.3 додатка В.

Об'єкт window. У найвищій частині ієрархічної структури об'єктів документа знаходиться об'єкт window.

Цей об'єкт є основним контейнером, у якому розміщується все те, чим можна управляти за допомогою Web-браузера. Протягом всього часу, поки вікно браузера відкрите, об'єкт window буде перебувати в поточній об'єктній моделі.

Для використання методів і властивостей об'єкта window існує наступний синтаксис:

```
window.ім'яВластивості
window.ім'яМетоду([параметри])
```

До об'єкта *window* часто звертаються за допомогою синоніму

self

```
self.ім'яВластивості
self.ім'яМетоду([параметри])
```

Ідентифікатор *self* використовується для позначення поточного вікна – вікна, у якому знаходиться документ із даним сценарієм.

Головне вікно браузера в сценарії не створюється. Однак за допомогою сценарію можна згенерувати будь-яке число додаткових вікон за допомогою методу

```
window.open().
```

Для закриття вікна використовується метод `window.close()`.

Приклад:

```
<script type="text/Javascript">
var newWindow
function mknewWindow() {
newWindow=window.open("", "", "height=300,width=
300");
}
function closenewWindow() {
if(newWindow) {
newWindow.close();
newWindow=null;
}
}
</script>
```

В таблиці В.1 додатка В наведені властивості й методи, що доступні у всіх браузерах, що підтримують сценарії.

Об'єкт document. В об'єкті *document* зберігається весь реальний вміст сторінки. Властивості й методи об'єкта *document* в основному впливають на зовнішній вигляд і вміст сторінки, завантаженої у вікні. У браузерах W3C DOM можна за допомогою сценарію одержувати доступ до тексту сторінки тільки в тому випадку, якщо документ повністю завантажений.

Одержати доступ до властивостей і методів об'єкта *document* можливо таким чином:

```
[window.]document.ім'я_Властивості  
[window.]document.ім'я_Методу(х-параметри)
```

Посилання на вікно *window* є необов'язковим.

Об'єкт *form* і вкладені в нього об'єкти. Форми та їхні елементи – єдиний спосіб надати користувачеві можливість введення текстової інформації.

Властивості елемента *form*:

- *action* – дана властивість подібна значенню, що присвоюється атрибуту *action* дескриптора `<form>`. Значення, що повертає – це звичайно URL сервера, на яку відправляють запити або дані, що оброблюються;

- *elements[]* – масив елементів керування форми. Дана властивість визначає в собі всі елементи інтерфейсу користувача, розміщені у формі;

- *length* – визначає кількість елементів у формі;

- *method* – має значення *GET* або *POST* атрибуту *method* дескриптора `</form>`;

- *name* – ідентифікатор форми;

- *target* – ім'я вікна або фрейму.

Методи елемента *form*

- *handleEvent(подія)* – керівник подій:

- *reset()* – викликається якщо для відновлення значень за замовчуванням потрібно використати сценарій;

- *submit()* – найбільш популярний спосіб відправити дані форми на сервер для обробки в CGI-програмі.

Оброблювачі подій:

- *onreset* – перш ніж кнопка *reset* поверне значення за замовчуванням, *JavaScript* генерує у формі подію *reset*. Включивши оброблювач події *onreset* у визначення форми, ви можете захопити цю подію перш, ніж відбудеться відновлення значень за замовчуванням;

- *onsubmit* – подія, що генерується у формі при натисканні кнопки *Submit*.

Об'єкт *кнопки*. Властивості:

- *form* – посилання на об'єкт *form*;

- *name* – ідентифікатор;

- *type* – рядок, що визначає тип елемента керування;

- *value* – рядок значення.

Методи об'єкта *кнопки*:

- *click()* – імітує «клік» користувача на кнопці.

Оброблювачі подій:

- *onclick* – виникає при натисканні кнопки;

- *onmousedown* та *onmouseup* – являє собою єдиний оброблювач *onclick*.

Об'єкти *checkbox* та *radio*. Властивості:

- *checked* – вказує чи встановлений значок. Приймає значення *true* або *false*;

- *form* – посилання на об'єкт *form*;

- *name* – ідентифікатор;

- *type* – рядок, що визначає тип елемента керування;

- *value* – рядок значення.

Методи об'єктів *checkbox* та *radio*:

- *click()* – імітує «клік» користувача на кнопці.

Оброблювачі подій: *onclick* – виникає при натисканні кнопки.

Об'єкт *image*. Властивості:

- *complete* – приймає значення *true* тільки після закінчення завантаження;

- *form* – посилання на об'єкт *form*;

- *name* – ідентифікатор;

- *src* – управляє адресою URL зображення, відображеного в елементі.

- *type* – використовується для ідентифікації об'єкта *image* у невідомій групі елементів.

Об'єкт *text* та *password*. Властивості:

- *defaultValue* – рядок, привласнений елементу за замовчуванням;

- *form* – посилання на об'єкт *form*;

- *maxLength* – визначає максимальну кількість символів, яку можна ввести в поле;

- *name* – ідентифікатор;

- *readonly* – логічний атрибут, що вказує, чи можна змінювати значення поля, або воно тільки для читання;

- *size* – розмір текстового поля. Якщо значення не зазначене, то за замовчуванням 20 символів;

- *type* – рядок, що визначає тип елемента керування;

- *value* – рядок значення.

Методи об'єктів *text* та *password*:

- *select()* - виділення поля під управлінням сценарію означає виділення всього тексту даного текстового об'єкта. У додатку звичайно знаходиться сценарій, призначений для перевірки введених користувачем даних на наявність помилок.

Оброблювачі подій

- *onchange* – використовується для негайної перевірки введених користувачем даних;

- *onselect* – виникає при виборі елемента.

6.2 Порядок виконання лабораторної роботи

1. Додайте оброблювачі подій для елементів форми. Наприклад:

```
<input type="button" value="Click here"
onClick='alert("Hello!")' />
<textarea name="msg" onFocus="eraseMsg()"> Some
text </textarea>
```

До коду скрипта додати:

```
erased=false;
function eraseMsg()
{ if (erased==false)
  { document.forms[0].msg.value="";
    erased=true; }
}
```

2. Використайте подію *onsubmit* для перевірки коректності заповнення електронної адреси. Виведіть інформацію, що міститься у формі у нове вікно. Наприклад:

```
<form method="post" id="lab6" onsubmit =
create_window()>
```

До коду скрипта додати:

```
function create_window() {
  if (check()!=true)
    return
  var win = window.open('', 'new', 'height=220,
width=450, continued from previous line toolbar=no,
directories=no, status=no, menubar=no, continued from
previous line scrollbars=no, resizable=no');
  win.document.write(document.forms['lab6'].mail.value)
```

```
function check() {
  var strmail=document.forms['lab6'].mail.value
```

```

if
((strmail.indexOf("@")<0)|| (strmail.indexOf(".")<0)|| (s
trmail.lastIndexOf(".")<strmail.indexOf("@")))
{
    alert("Введіть правильний e-mail!")
    return false
}
return true
}

```

6.3 Завдання для виконання

1. Використати форму розроблену у попередній лабораторній роботі.
2. Додати функції для перевірки обробки необхідних полів.
3. Додати функцію, що обробляє вміст форми та генерує нове вікно з отриманою інформацією.

6.4 Контрольні запитання

1. Який тег дозволяє включити *JavaScript* в HTML документ?
2. Перелічіть основні конструкції мови *JavaScript*.
3. Розкрийте ієрархію об'єктів документа.
4. Перелічіть основні властивості й методи об'єкта *window*.
5. Перелічіть основні властивості й методи об'єкта *document*.
6. Які події використовують при обробці даних форми?
7. Що таке подія?
8. Як виконується доступ до властивостей форми?
9. Що зберігається в масиві *elements*?
10. Як одержати доступ до об'єктів форми?

ПЕРЕЛІК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. І. Л. Бородкіна. Web-технології та Web-дизайн: застосування мови HTML для створення електронних ресурсів. І. Л. Бородкіна, Г. О. Бородкін / К.: Ліра-К, 2020 р. – 212 с.

2. Лаптон Еллен «Графічний дизайн: Нові основи» Еллен Лаптон, Дженніфер Коул Філіпс, / переклад: Михайлишена І., ArtHuss, 2019 р. – 262 с.

3. Олійник, О. П. Теорії та концепції дизайну [Текст]: навчальний посібник / О. П. Олійник. – Київ: НАУ, 2020. – 256 с. – ISBN 978-966-289-401-1

4. Брюханова, Г. В. Комп'ютерні дизайн-технології [Текст] : навч. посіб. / Г. В. Брюханова. – К. : ЦУЛ, 2019. – 180 с.

Додаткова.

5. Возняк, Л.С. Комп'ютерний практикум. Формування навичок роботи із сервісами мережі Інтернет. І.-Ф.: ВДВ ЦІТ, 2006.

6. Глинський, Я. М. Інтернет. Сервіси, HTML і web-дизайн: Навч. посіб.- 3-є вид. Львів : Деол, СПД Глинський, 2005.

7. Самсонов В. В. Методи та засоби Інтернет-технологій. Харків: Компанія СМІТ, 2008.

8. Юринець В.Є. Комп'ютерний практикум. Формування навичок роботи із сервісами мережі Інтернет. Львів: ВЦ ЛНУ, 2006

ІНФОРМАЦІЙНІ РЕСУРСИ

9. Вебтехнології. [Електронне видання], Режим доступу: <https://uk.wikipedia.org/wiki/Категорія:Вебтехнології>

Додаток А

Властивості форматування блокових елементів

Таблиця А.1 – Властивості елементів

Властивість	Опис	Можливі значення	Область застосування
1	2	3	4
padding padding-top padding-right padding-left padding-bottom	відступ від границі елемента до його вмісту	довжина, процент	для всіх елементів
margin margin-top margin-right margin-left margin-bottom	зовнішній відступ елемента	довжина, процент	для всіх елементів
border-width border-top- width border- right-width border-left- width border- bottom-width	товщина границі	довжина, thin, medium, thick	для всіх елементів
border-color	колір границі	колір	
border-style	стиль границі	none, dotted, dashed, solid, double, groove, ridge, inset, outset	
border border-top border-right border-left border-bottom	узагальнює перераховані вище властивості для вказаної границі	border-width border-style border-color	

Продовження таблиці А.1

1	2	3	4
Width	ширина елемента	довжина, процент	Для блокових елементів
Height	висота елемента		
Display	визначає, як буде відображатися елемент	none, block, inline, list-item	для всіх елементів
list-style-type	визначає вид маркеру списку	none, disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha	для елементів зі значенням display, що дорівнює list-item
list-style-image	задає вид маркеру у вигляді малюнку	none, URL	
list-style-position	визначає положення маркеру в залежності від елемента списку	inside - при переносі наступні рядки будуть відображатися без відступу, outside - за замовченням	
list-style	узагальнює перераховані вище властивості	list-style-type list-style-position list-style-image	
font-family	визначає шрифт, що використовується елементом. Якщо вказати URL, то шрифт автоматично встановиться на комп'ютер користувача	будь-який шрифт	для всіх елементів
font-style	стиль елемента	normal, italic	
font-weight	виділення (жирність) елемента	normal, bold, bolder, lighter, будь-яке значення від 100 до 900	

Продовження таблиці А.1

1	2	3	4
font-size	розмір шрифту	розмір, xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger	
Font	узагальнює перераховані вище властивості	font-family font-style font-variant font-weight font-size	
vertical-align	позиціонування елемента відносно до інших елементів, що стоять в одному рядку	baseline, sub, super, top-text, top, middle, bottom, bottom-text, процент	для рядкових елементів
text-align	вирівнювання тексту	left, right, center, justify	для блокових елементів
line-height	висота рядка	normal, довжина, процент	для всіх елементів
Color	Колір	колір	для всіх елементів
background-color	цвет фона елемента	цвет	для всіх елементів
background-image	фонове зображення	none, URL	
background-repeat	повторювання фонового зображення	repeat, repeat-x, repeat-y, no-repeat	
background-attachment	можливість прокрутки фонового зображення	scroll, fixed	

Продовження таблиці А.1

1	2	3	4
background-position	положення фонового зображення	процент від ширини + процент від висоти, top, middle, bottom, left, center, right, відстань від лівого краю + відстань від вершини	для блокових елементів
background	узагальнює перераховані вище властивості	background-color background-image background-position background-attachment background-repeat	для всіх елементів
position	встановлює спосіб позиціювання елемента відносно вікна браузера або інших об'єктів на веб-сторінці	absolute, fixed, relative, static	для всіх елементів
top right left bottom	відстань від краю батьківського елемента	довжина, процент	для всіх елементів
z-index	положення елемента на осі z	число	для будь-яких позиціонованих елементів

Додаток Б

Основи програмування на мові JavaScript

Мова JavaScript дуже схожа на мову C++.

Однорядкові коментарі починаються із символу `"/"`.

Багаторядкові коментарі починаються з `"/*"` і завершуються `"*/"`.

`//` Коментар обмежений даним рядком

`/*` Коментар починається тут

і завершується тут`*/`

Змінні оголошуються за допомогою ключового слова `var`, за яким йде ім'я змінної. Можливе оголошення відразу декількох змінних, у цьому випадку вони розділяються комами. Крім того, змінні можуть бути відразу ініціалізовані при оголошенні, тобто їм буде присвоєне значення.

```
var NameofVariable1, NameofVariable2 = 567;
```

Тут змінна *NameofVariable1* просто оголошена, а змінна *NameofVariable2* ще й ініціалізована при оголошенні і їй привласнене значення **567**.

Областю видимості змінної є поточна функція або, у випадку оголошення поза функцією, весь поточний документ (web-сторінка).

```
var globalString;
```

Типи змінних: чисельний, логічний, рядковий.

Чисельні змінні можуть містити цілі числа й числа зі плаваючою крапкою. Цілі числа можуть бути виражені у звичайному десятковому (на основі 10), шістнадцатерічному (на основі 16) або восьмирічному (на основі 8) поданні.

Логічні змінні можуть мати тільки два значення: *true* або *false*. Замість *true* й *false* можна використати відповідно 1 і 0.

Рядкові змінні містять будь-яку кількість символів – рядок. При присвоюванні рядкової змінної значення (рядка) рядок завжди обмежений одинарними або подвійними лапками.

```
var variable1 = 534, variable2 = true,
variable3 = "рядок";
```


Ми оголосили змінні: *variable1* – чисельна, *variable2* – логічна, *variable3* – рядкова.

Для створення масиву використовується наступний синтаксис:

```
var ім'я масиву = new Array (розмірність);
```

Звертання до елементів масиву здійснюється через відповідний індекс. Індикація масивів виконується з 0. Основні конструкції мови JavaScript наведені в таблиці Б.1.

Таблиця Б.1 – Основні конструкції мови JavaScript

Назва	Опис
Умовний оператор	if (вираз) Д1; else Д2;
Оператор вибору	switch (вираз){ case val1: Д1; break; case val2:Д2; break; default: DD; }
Ітераційний оператор циклу	for(опер_циклу=val; умова_виконання; опер_циклу++) { //тіло циклу}
Оператор циклу з передумовою	while (умова виконання){ // тіло циклу}
Оператор циклу з постумовою	do{//тіло циклу } while(умова виконання)

Шаблон оголошення функції на мові *JavaScript* має наступний вигляд:

```
function      ім'я_функції([параметр1      , ...,
[параметрN])
    { //тіло функції }
```

На імена, що привласнюються функціям, накладаються ті ж обмеження, що й на імена елементів і змінних HTML.

Змінні, оголошені поза функціями, називаються глобальними змінними. Змінні, оголошені в межах функцій, називаються локальними змінними.

В *JavaScript* межі глобальності для змінних доходять до розмірів поточного документа, завантаженого у вікно браузера. Тому ініціалізація змінної в якості глобальної має на увазі, що всі оператори сторінки одержують прямий доступ до значення цієї змінної.

Додаток В

Методи та властивості об'єктів документа

Таблиця В.1 – Властивості і методи вікна

№	Властивості і методи	Опис
1	2	3
1	window.status	Зберігає вміст статусного рядка вікна. <SCRIPT> window.status = "Ця сторінка написана на JavaScript!"; </SCRIPT>
Методи		
1	window.alert()	Даний метод генерує діалогове вікно, що відображає той текст, який передається методу як параметр. Єдина кнопка ОК призначена для закриття вікна. Метод Alert
2	window.confirm()	Діалогове вікно, яке може повернути значення <i>true</i> або <i>false</i> Метод Confirm
3	window.prompt()	Генерує діалогове вікно запиту. В ньому відображується визначене розробником сторінка повідомлення і виводиться текстове поле для вводу відповіді. Метод Prompt

Таблиця В.2 – Властивості і методи об'єкта *document*

№	Властивості і методи	Опис
Властивості		
	<code>document.forms[]</code>	Для визначення того, скільки в документі форм використовується команда <code>document.forms.length</code> Для одержання доступу використовуються індекси <code>document.forms[0]</code> або ім'я name дескриптора <code><form></code> <code>document.forms["ім'яФорми"]</code> <code>document.ім'яФорми</code>
	<code>document.images[]</code>	Масив зображень, які вставляються в документ за допомогою дескриптора <code></code>
Методи		
	<code>document.write()</code>	Даний метод використовується як для створення вмісту завантажуваної сторінки, так і для створення нового вмісту у поточного або іншого вікна. <code>document.write("А це JavaScript!")</code>
	<code>document.createElement()</code>	Дозволяє створити в пам'яті зовсім новий об'єкт. Щоб у точності вказати тип створюваного елемента треба підставити ім'я дескриптора цього елемента в рядковий параметр методу. <code>var newel=document.createElement("p")</code>
	<code>document.createTextNode()</code>	Даний метод дозволяє створювати новий текстовий вузол <code>var newText=document.createTextNode("Hi")</code>
	<code>document.getElementById()</code>	Дозволяє дізнатись код (дескриптор елемента) <code>var v1=document.getElementById("Res")</code>

Таблиця В.3 – Оброблювачі подій

Оброблювач	Опис
1	2
Onactivate onbeforedeactivate ondeactivate	Якщо елемент активізується, то подія <i>onactivate</i> запускається перед <i>onfocus</i> і навпаки, до деактивації елемента події запускаються в наступній послідовності: <i>onbeforedeactivate</i> , <i>ondeactivate</i> , <i>onblur</i>
Onbeforecopy	Даний оброблювач запускається до того, як відбувається копіювання, ініційоване користувачем в результаті виклику опції меню <i>Edit</i> або контекстного меню. <i>Даний оброблювач не працює з елементами введення тексту у формах</i>
Onbeforecut	Запускається до того, як відбувається вирізка, ініційована користувачем в результаті вибору опції меню <i>Edit</i> або контекстного меню. Якщо додати оброблювач в елемент HTML, у контекстному меню операція <i>Cut</i> буде заборонена.
Onclick	Реагує на одинарний клік курсора миші. Приклад
Oncontextmenu	Відбувається, коли користувач клацає на об'єкті правою кнопкою миші. Цей клік викликає лише дві події <i>onmousedown</i> та <i>oncontextmenu</i>
Oncopy oncut	Відбуваються після того, як користувач або сценарій ініціюють дію копіювання або вирізки поточного об'єкта.
Ondblclick	Реагує на подвійний клік курсора миші. Приклад

Продовження таблиці В.3

1	2
ondrag, ondragend, ondragstart	Реагує на перетягування курсором миші. <code> Приклад </code>
Ondrop	Відбувається тоді, коли користувач відпускає кнопку миші, завершуючи тим самим операцію "перетягування".
Onfocus	Відбувається, коли елемент активізується; як правило в цей момент деактивізується інший елемент.
Onhelp	При одинарному кліку курсором миші можна натиснути кнопку F1, при цьому відобразиться підказка з інструкціями про те, що дане посилання реагує на подвійний клік курсором миші. <code> Приклад </code>
onkeydown	Реагує на натискання і утримування клавіші. <code> Приклад </code>
onkeypress	Реагує на натискання клавіші. <code> Приклад </code>

Продовження таблиці В.3

1	2
onkeyup	Реагує на відпускання клавіші. <code> Приклад </code>
onmousedown	Реагує на натискання кнопки миші. <code> Приклад </code>
onmouseup	Може використовуватися, щоб виконати функції сценарію після того, як миша користувача була натиснута й відпущена – протилежність події OnMouseDown. <code> Приклад </code>
onmousemove	Виконується тоді, коли покажчик миші переміщується по поточному об'єкту. При цьому кнопка миші не обов'язково повинна бути натиснутою, хоча дана подія в більшості випадків використовується при переягуванні об'єкта.
onmouseout	Реагує на подію, коли курсор миші залишає посилання. <code> Приклад </code>
onmouseover	Реагує на подію, коли курсор миші потрапляє у область посилання. <code> Приклад </code>
Onpaste	Виникає відразу ж після того, як користувач або сценарій ініціює операцію вставки для поточного об'єкта.
onresize	Запускається при зміні розмірів об'єкта у відповідь на дії користувача або сценарію.

Продовження таблиці В.3

1	2
Onselectstart	<p>Виконується щораз, коли користувач починає вибирати деякий текст, що є змістом елемента.</p> <pre> <table BORDER="8" BgColor="red" WIDTH="65%" cellspacing=8> <tr><td> <p ID="paraSelectStart">Вибip будь-якого елемента з цього тексту змусить змінювати його кольори ... іноді дикувати <script LANGUAGE="VBScript"> <!-- Sub paraSelectStart_onselectstart() Dim hexColor Dim hexColor2 hexColor=hex(rnd*16777215) if hexColor>"ffffff"then hexColor="ffffff" hexColor2=hex(rnd*16777215) if hexColor2>"ffffff"then hexColor2="ffffff" on error resume next set elRef=document.all("paraSelectStart") elRef.style.backgroundColor=hexColor elRef.style.color=hexColor2 self.event.returnValue=false End Sub --> </script> </td></tr> </table> </pre>