

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО



МЕТОДИЧНІ ВКАЗІВКИ
ЩОДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ
З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ
«WEB-ДИЗАЙН»
ДЛЯ СТУДЕНТІВ ДЕННОЇ ФОРМИ НАВЧАННЯ
ЗІ СПЕЦІАЛЬНОСТІ 122 – «КОМП'ЮТЕРНІ НАУКИ»
ОСВІТНЬО-ПРОФЕСІЙНОЇ ПРОГРАМИ
«КОМП'ЮТЕРНІ НАУКИ»
ОСВІТНЬОГО СТУПЕНЯ «БАКАЛАВР»

Методичні вказівки щодо виконання лабораторних робіт з навчальної дисципліни «WEB-дизайн» для студентів денної форми навчання зі спеціальності 122 – «Комп'ютерні науки» освітньо-професійної програми «Комп'ютерні науки» освітнього ступеня «Бакалавр»

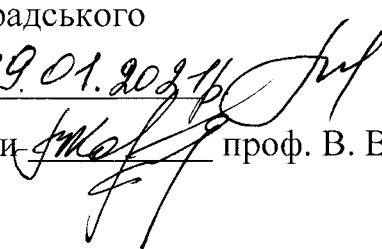
Укладачі: к. т. н., доц. І. Г. Оксанич,
асист. К. С. Король

Рецензент к. т. н., старш. викл. Ю. О. Краснопольська

Кафедра автоматизації та інформаційних систем

Затверджено методичною радою Кременчуцького національного університету імені Михайла Остроградського

Протокол № 3 від 19.01.2024

Голова методичної ради  проф. В. В. Костін

ЗМІСТ

Вступ.....	4
1 Вимоги до оформлення звіту з лабораторних робіт.....	6
2 Перелік лабораторних робіт.....	7
Лабораторна робота № 1 Створення WEB-сторінок мовою HTML.....	7
Лабораторна робота № 2 Візуалізація WEB-сторінок HTML за допомогою CSS.....	24
Лабораторна робота № 3 Використання блочної верстки для створення макету WEB-сторінки.....	40
Лабораторна робота № 4 Створення адаптивного дизайну.....	50
Лабораторна робота № 5 Застосування CSS-фреймворків для створення WEB-сторінок.....	54
3 Критерії оцінювання знань студентів.....	63
Список літератури.....	64
Додаток А Зразок оформлення титульної сторінки звіту.....	65

ВСТУП

Методичні вказівки щодо виконання лабораторних робіт з навчальної дисципліни «WEB-дизайн» містять 5 лабораторних робіт. Виконання лабораторних робіт спрямоване на набуття практичних навичок дизайну та створення WEB-сторінки, використовуючи мову HTML та CSS.

У наданих методичних вказівках розглянуто створення WEB-сторінок мовою HTML, їх візуалізацію за допомогою мови CSS та CSS-фреймворків.

Викладення матеріалу в методичних вказівках побудовано від простого до складного. Використовується велика кількість прикладів, що сприяє кращому засвоєнню матеріалу. Окрім того, у методичних вказівках надані всі відомості, необхідні для виконання лабораторних робіт, що надають можливість зрозуміти принцип створення WEB-сторінки.

Освоєння WEB-дизайну є додатковим аспектом у вивченні необхідних технологій, який забезпечує фахівця ще одним потужним інструментом для створення сайтів.

Засвоївши навички створення WEB-сторінок, фахівець має можливість створювати та працювати зі складними комплексними сайтами, розробляти та змінювати їх дизайн.

Майбутній фахівець, вивчаючи WEB-дизайн і володіючи навичками програмування, буде мати можливість самостійно провести розробку дизайну та створити WEB-сайт.

Вивчення WEB-дизайну стимулює засвоєння розуміння створення WEB-сайтів, також фахівець засвоює різні підходи розробки та проектування WEB-сторінок.

Метою вивчення навчальної дисципліни «WEB-дизайн» є вивчення основ роботи з протоколом, оволодіння навичок створення WEB-дизайну та спеціалізованих користувацьких інтерфейсів.

Завданням навчальної дисципліни «WEB-технології» є отримання студентами теоретичних основ і практичних навичок застосування WEB-дизайну для використання в задачах розробки WEB-застосунків.

Завдяки виконанню лабораторних робіт майбутні спеціалісти практично засвоять технологію створення WEB-сторінок з використанням мови HTML та CSS.

У результаті вивчення навчальної дисципліни студент повинен

знати:

- засоби виразності реклами, використовувані в Інтернеті;
- різні види програмно-технічних засобів, що дозволяють створювати електронні рекламні матеріали в Інтернеті;
- способи і методи формування і просування сайтів в Інтернеті;

уміти:

- формувати структуру (сценарій) сайтів з рекламою;
- використовувати відповідні програмно-технічні засоби для створення сайтів в Інтернеті;
- представляти свій варіант сайта в Інтернеті.

1 ВИМОГИ ДО ОФОРМЛЕННЯ ЗВІТІВ

3 ЛАБОРАТОРНИХ РОБІТ

Звіт про виконання лабораторних робіт (далі – звіт) має бути написаний студентом власноруч, розбірливим почерком, чисто й охайно, однаковим чорнилом (синім чи фіолетовим) чи пастою на аркушах білого паперу форматом А4 (210×297 мм) або набраний за допомогою комп'ютерної техніки шрифтом Times New Roman, розміром 14 пунктів на одному боці аркуша. На одній сторінці допускається не більш ніж три виправлення, зроблені охайно та розбірливо (допускається застосування коректора).

Зміст звіту

Звіт має містити:

- назву роботи;
- тему та мету роботи;
- завдання до самостійної роботи;
- тексти написаних програм;
- висновки з роботи.

Підготовка до виконання лабораторних робіт

Підготовка до кожної лабораторної роботи проводиться студентом самостійно. Студент повинен ознайомитися з теоретичними відомостями й порядком виконання роботи. За допомогою конспекту лекцій і додаткової літератури необхідно опрацювати певні теми й відповісти на контрольні питання, наведені в лабораторній роботі.

Під час виконання роботи необхідно виконати всі завдання, які наведені у відповідному пункті, та самостійно виконати завдання згідно зі своїм варіантом. Після завершення роботи необхідно оформити звіт, зміст якого визначається відповідним пунктом. Для захисту роботи необхідно виконати додаткове завдання, яке визначає викладач.

Лабораторна робота захищається під час індивідуальної бесіди студента з викладачем.

2 ПЕРЕЛІК ЛАБОРАТОРНИХ РОБІТ

Лабораторна робота № 1

Тема. Створення WEB-сторінок мовою HTML

Мета: ознайомлення та вивчення принципів розробки WEB-сторінок мовою HTML.

Короткі теоретичні відомості

1. Мова HTML

HTML (HyperText Markup Language) є мовою розмітки гіпертексту, яка використовується переважно для створення документів у мережі інтернет. HTML почав свій шлях на початку 90-х років як примітивна мова для створення WEB-сторінок, і зараз вже важко уявити собі інтернет без HTML. Переважна більшість сайтів так чи інакше використовують HTML.

2014 року офіційно була завершена робота над новим стандартом – HTML5, який фактично зробив революцію, несучи в HTML багато нового. Що саме привніс HTML5?

- HTML5 визначає новий алгоритм парсинга для створення структури DOM;
- додавання нових елементів і тегів, як, наприклад, елементи video, audio і низка інших;
- перевизначення правил і семантики вже існуючих елементів HTML.

Фактично з додаванням нових функцій HTML5 став не просто новою версією мови розмітки для створення WEB-сторінок, але і фактично платформою для створення додатків, а галузь його використання вийшла далеко за межі WEB-середовища інтернет: HTML5 застосовується також для створення мобільних додатків під Android, iOS, Windows Mobile і навіть для створення десктопних додатків для звичайних комп'ютерів (зокрема, в ОС Windows 8 / 8.1 / 10).

2. Інструменти для роботи з мовою HTML

Для роботи з HTML необхідно використовувати текстовий редактор, щоб набирати текст WEB-сторінок на HTML. На даний момент одним з найпростіших і найбільш популярних текстових редакторів є Notepad++, який можна знайти за адресою <http://notepad-plus-plus.org/>. До його переваг можна віднести безкоштовність, підсвічування тегів HTML.

Також варто згадати багатоплатформовий текстовий редактор Visual Studio Code. Цей редактор має дещо більші можливості, ніж Notepad++, і, крім того, може працювати не тільки в ОС Windows, але і в MacOS і в операційних системах на основі Linux.

Також необхідний WEB-браузер для запуску та перевірки написаних WEB-сторінок. Як WEB-браузера можна взяти останню версію будь-якого з поширених браузерів – Google Chrome, Mozilla Firefox, Microsoft Edge, Opera.

3. Створення документа HTML5

Для створення документа необхідно створити простий текстовий файл, а як розширення файлу вказати *.html. Потім відкриємо цей файл у будь-якому текстовому редакторі, наприклад, у Notepad ++. Додамо до файлу такий текст:

```
<!DOCTYPE html>  
  
<html>  
  
</html>
```

Для створення документа HTML5 нам потрібні першочергово два елементи: DOCTYPE і html. Елемент DOCTYPE або Document Type Declaration повідомляє WEB-браузеру тип документа. <! DOCTYPE html> указує, що даний документ є документом html і що використовується HTML5, а не HTML4 або якась інша версія мови розмітки.

Елемент html між своїми відкриваючим і закриваючим тегами містить весь вміст документа. У середині елемента html ми можемо розмістити два інших елемента: head і body. Елемент head містить метадані WEB-сторінки – заголовок WEB-сторінки, тип кодування і т. д., а також посилання на зовнішні

ресурси – стилі, скрипти, якщо вони використовуються. Елемент `body` власне визначає вміст `html`-сторінки.

Тепер змінимо вміст файлу `index.html` так:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Документ HTML5</title>
  </head>
  <body>
    <div>Содержание документа HTML5</div>
  </body>
</html>
```

Содержание документа HTML5

Рисунок 1.1 – Результат роботи коду

В елементі `head` визначено два елементи:

- елемент `title` є заголовком сторінки;
- елемент `meta` визначає метадані сторінки.

Для коректного відображення символів переважно вказувати кодування. У цьому випадку за допомогою атрибута `charset = "utf-8"` вказуємо кодування `utf-8`.

У межах елемента `body` використовується тільки один елемент – `div`, який оформляє блок. Вмістом цього блоку є простий рядок.

4. Елементи у HTML5. Форматування тексту

Низка елементів HTML призначені для форматування текстового вмісту, наприклад, для виділення жирним або курсивом і т. д. Розглянемо ці елементи:

- ``: виділяє текст жирним;
- ``: закреслює текст;

- <i>: виділяє текст курсивом;
- : виділяє текст курсивом, на відміну від тега <i> носить логічне значення, надає виділеному тексту відтінок важливості;
- <s>: закреслює текст;
- <small>: робить текст трохи менше розміром, ніж навколишній;
- : виділяє текст жирним. На відміну від тега призначений для логічного виділення, щоб показати важливість тексту. А не носить характеру логічного виділення, виконує функції тільки форматування.
- <sub>: розміщує текст під рядком;
- <sup>: поміщає текст над рядком;
- <u>: підкреслює текст;
- <ins>: визначає вставлений (або доданий) текст;
- <mark>: виділяє текст кольором, надаючи йому відтінок важливості.

Застосуємо ці елементи:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Форматирование текста в HTML5</title>
  </head>
  <body>
    <p>Форматирование в <mark>HTML5</mark></p>
    <p>Это <b>выделенный</b> текст</p>
    <p>Это <strong>важный</strong> текст</p>
    <p>Это <del>зачеркнутый</del> текст</p>
    <p>Это <s>недействительный</s> текст</p>
    <p>Это <em>важный</em> текст</p>
    <p>Это текст <i>курсивом</i> </p>
    <p>Это <ins>добавленный</ins> текст</p>
    <p>Это <u>подчеркнутый</u> текст</p>
```

```

        <p>X<sub>i</sub> = Y<sup><small>2</small></sup> +
Z<sup><small>2</small></sup></p>
    </body>
</html>

```

Форматирование в **HTML5**

Это **выделенный** текст

Это **важный** текст

Это зачеркнутый текст

Это ~~недействительный~~ текст

Это *важный* текст

Это текст *курсивом*

Это добавленный текст

Это подчеркнутый текст

$$X_i = Y^2 + Z^2$$

Рисунок 1.2 – Результат работы коду

5. Елементи угруповування

Елемент `div` слугує для структуризації контенту на WEB-сторінці для укладення вмісту в окремі блоки. `Div` створює блок, який за замовчуванням розтягується за всією шириною браузера, а наступний після `div` елемент переноситься на новий рядок. Наприклад:

```

<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Документ HTML5</title>
    </head>
    <body>
        <div>Заголовок документа HTML5</div>

```

```
<div>Текст документа HTML5</div>
</body>
</html>
```

Заголовок документа HTML5
Текст документа HTML5

Рисунок 1.3 – Результат роботи коду

Параграфи створюються за допомогою тегів `<p>` і `</p>`, які укладають деякий вміст. Кожен новий параграф розташовується на новому рядку. Застосуємо параграфи:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Документ HTML5</title>
  </head>
  <body>
    <div>Заголовок документа HTML5</div>
    <div>
      <p>Первый параграф</p>
      <p>Второй параграф</p>
    </div>
  </body>
</html>
```

Заголовок документа HTML5
Первый параграф
Второй параграф

Рисунок 1.4 – Результат роботи коду

Елемент `span` обтікає деякий текст за всією його довжиною і слугує переважно для стилізації укладеного в нього текстового вмісту. На відміну від блоків `div` або параграфів `span` не переносить вміст на наступний рядок:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Документ HTML5</title>
  </head>
  <body>
    <div>Заголовок документа HTML5</div>
    <div>
      <p><span style="color:red;">Первый</span> параграф</p>
      <p><span>Второй</span> параграф</p>
    </div>
  </body>
</html>
```

Заголовок документа HTML5

Первый параграф

Второй параграф

Рисунок 1.5 – Результат роботи коду

При цьому слід відзначити, що сам по собі `span` нічого не робить. Так, у другому параграфі `span` ніяк не вплинув на внутрішній текстовий вміст. А в першому параграфі елемент `span` містить атрибут стилю: `style = "color: red;"`, який встановлює для вкладеного тексту червоний колір фону.

При цьому слід відзначити, що елементи `div` і `p` є блоковими, елемент `div` може містити будь-які інші елементи, а елемент `p` – тільки рядкові елементи. На відміну від них елемент `span` є рядковим, тобто як би вбудовує свій вміст у

зовнішній контейнер – той же div або параграф. Але при цьому не слід поміщати блокові елементи в рядковий елемент span.

6. Робота із зображеннями

Для виведення зображень в HTML використовується елемент `img`. Цей елемент надає нам два важливих атрибута:

- `src`: шлях до зображення. Це може бути відносний або абсолютний шлях до файлової системи або адреса в інтернеті;

- `alt`: текстовий опис зображення. Якщо браузер з якихось причин не може відобразити зображення (наприклад, якщо у атрибута `src` некоректно поставлений шлях), то браузер показує замість самої картинки дане текстовий опис.

Атрибут `alt` ще важливий тим, що пошукові системи за текстовим описом можуть індексувати зображення.

Приклад роботи із зображенням:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset=»utf-8»>
    <title>Тег img в HTML5</title>
  </head>
  <body>
    <img src=»dubi.png» alt=»Зимняя равнина» />
  </body>
</html>
```

7. Таблиці та списки

Для створення таблиць у HTML використовується елемент `table`. Кожна таблиця між тегами `<table>` і `</table>` містить рядки, що представлені елементом `tr`. А кожен рядок між тегами `<tr>` і `</tr>` складається з комірок у вигляді елементів `td`.

Приклад створення простої таблиці:

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset=»utf-8»>
    <title>Таблиці в HTML5</title>
  </head>
  <body>
    <table>
      <tr>
        <td>Модель</td> <td>Компанія</td>
<td>Ціна</td>
      </tr>
      <tr>
        <td>Nexus 6P</td> <td>Huawei</td>
<td>49000</td>
      </tr>
      <tr>
        <td>iPhone 6S Plus</td> <td>Apple</td>
<td>62000</td>
      </tr>
    </table>
  </body>
</html>

```

Модель	Компанія	Ціна
Nexus 6P	Huawei	49000
iPhone 6S Plus	Apple	62000

Рисунок 1.6 – Результат роботи коду

Для створення списків у HTML5 застосовуються елементи `` (нумерований список) і `` (ненумерований список):

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Списки в HTML5</title>
  </head>
  <body>
    <h2>Нумерований список</h2>
    <ol>
      <li>iPhone 6S</li>
      <li>Galaxy S7</li>
      <li>Nexus 5X</li>
      <li>Lumia 950</li>
    </ol>
    <h2>Ненумерований список</h2>
    <ul>
      <li>iPhone 6S</li>
      <li>Galaxy S7</li>
      <li>Nexus 5X</li>
      <li>Lumia 950</li>
    </ul>
  </body>
</html>
```


Нумерований список

1. iPhone 6S
2. Galaxy S7
3. Nexus 5X
4. Lumia 950

Ненумерований список

- iPhone 6S
- Galaxy S7
- Nexus 5X
- Lumia 950

Рисунок 1.7 – Результат роботи коду

Для нумерованого списку за допомогою атрибуту `start` можна додатково задати символ, з якого буде починатися нумерація. Наприклад:

```
<h2>list-style-type = decimal</h2>
<ol style="list-style-type:decimal;" start="3">
  <li>iPhone 6S</li>
  <li>Galaxy S7</li>
  <li>Nexus 5X</li>
  <li>Lumia 950</li>
</ol>

<h2>list-style-type = upper-roman</h2>
<ul style="list-style-type:upper-roman;">
  <li>iPhone 6S Plus</li>
  <li>Galaxy S7 Edge</li>
  <li>Nexus 6P</li>
  <li>Lumia 950 XL</li>
</ul>

<h2>list-style-type = lower-alpha</h2>
<ul style="list-style-type:lower-alpha;">
```

```
<li>LG G 5</li>
<li>Huawei P8</li>
<li>Asus ZenFone 2</li>
</ul>
```

list-style-type = decimal

3. iPhone 6S
4. Galaxy S7
5. Nexus 5X
6. Lumia 950

list-style-type = upper-roman

- I. iPhone 6S Plus
- II. Galaxy S7 Edge
- III. Nexus 6P
- IV. Lumia 950 XL

list-style-type = lower-alpha

- a. LG G 5
- b. Huawei P8
- c. Asus ZenFone 2

Рисунок 1.8 – Результат роботи коду

8. Посилання

Посилання, які представлені елементом `<a> `, мають важливе значення – вони забезпечують навігацію між окремими документами. Цей елемент має такі атрибути:

- href: визначає адресу посилання;
- hreflang: вказує на мову документа, на який веде це посилання;
- media: визначає пристрій, для якого призначене посилання;
- rel: визначає відношення між цим документом і ресурсом, на який веде посилання;
- target: визначає, як документ за посиланням має відкриватися;
- type: вказує на mime-тип ресурсу за посиланням.

Найбільш важливим атрибутом є href:

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Посилання</title>
  </head>
  <body>
    <a href="content.html">Підручник по HTML5</a>
  </body>
</html>

```

[Підручник по HTML5](#)

Рисунок 1.9 – Результат роботи коду

Тут для посилання використовується відносний шлях content.html. Тобто в одній папці з цим документом має знаходитися файл content.html, на який буде йти перехід після натискання на посилання.

Також ми можемо поставити внутрішні посилання, які будуть переходити до певних блоків у середині елементів:

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Внутрішні посилання</title>
  </head>
  <body>
    <a href="#paragraph1">Параграф 1</a> | <a
href="#paragraph2">Параграф 2</a> | <a
href="#paragraph3">Параграф 3</a>
    <h2 id="paragraph1">Параграф 1</h2>

```

```
<p>Вміст параграфа 1</p>
<h2 id="paragraph2">Параграф 2</h2>
<p>Вміст параграфа 2</p>
<h2 id="paragraph3">Параграф 3</h2>
<p>Вміст параграфа 3</p>
</body>
</html>
```

[Параграф 1](#) | [Параграф 2](#) | [Параграф 3](#)

Параграф 1

Вміст параграфа 1

Параграф 2

Вміст параграфа 2

Параграф 3

Вміст параграфа 3

Рисунок 1.10 – Результат роботи коду

Щоб визначити внутрішнє посилання, вказується знак решітки (#), після якого йде id того елемента, до якого треба здійснити перехід. У цьому випадку перехід буде йти до заголовків h2.

За замовчуванням посилання вже має деякий колір (один з відтінків синього), окрім того воно має підкреслення. Під час натискання на посилання воно стає активним і набуває червоного кольору, а після переходу за посиланням це посилання може змінити колір в інший колір (зазвичай у фіолетовий). Подібна стилізація задається багатьма браузерами за замовчуванням, але ми можемо її перевизначити. Наприклад:

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Ссылки</title>
    <style>
a:link      {color:blue; text-decoration:none}
a:visited   {color:pink; text-decoration:none}
a:hover     {color:red; text-decoration:underline}
a:active    {color:yellow; text-decoration:underline}
    </style>
  </head>
  <body>
    <a href="index.html">Учебник по HTML5</a>
  </body>
</html>

```

Тут визначені стилі для посилань у різних станах:

- a: link застосовується для посилань у звичайному стані, коли вони не натиснуті й на них не наведений курсор миші;
- a: visited указує на стан посилання, за яким вже був здійснений перехід;
- a: hover указує на стан посилання, на який навели покажчик миші;
- a: active указує на посилання в натиснутому стані.

Стиль color встановлює колір посилання. А стиль text-decoration встановлює підкреслення: якщо значення underline, то посилання підкреслене, якщо none, то підкреслення відсутнє.

Помістивши в середину елемента <a> елемент , можна зробити посилання-зображення:

```

<a href="index.html">
  
</a>

```

Завдання до самостійної роботи

Розробити WEB-сторінку мовою HTML за своїм варіантом.

Варіанти завдань

1. Розробіть дві WEB-сторінки мовою HTML торговельного підприємства та автора. Перша WEB-сторінка має містити інформацію про підприємство (назву, адресу, телефони офісу, час роботи) та прайс-листок підприємства. На другій WEB-сторінці подайте відомості про розробника. Діяльність підприємства – торгівля комп'ютерами.

2. Розробіть дві WEB-сторінки мовою HTML торговельного підприємства та автора. Перша WEB-сторінка має містити інформацію про підприємство (назву, адресу, телефони офісу, час роботи) та прайс-листок підприємства. На другій WEB-сторінці подайте відомості про розробника. Діяльність підприємства – торгівля книгами.

3. Розробіть дві WEB-сторінки мовою HTML торговельного підприємства та автора. Перша WEB-сторінка має містити інформацію про підприємство (назву, адресу, телефони офісу, час роботи) та прайс-листок підприємства. На другій WEB-сторінці подайте відомості про розробника. Діяльність підприємства – торгівля журналами.

4. Розробіть дві WEB-сторінки мовою HTML торговельного підприємства та автора. Перша WEB-сторінка має містити інформацію про підприємство (назву, адресу, телефони офісу, час роботи) та прайс-листок підприємства. На другій WEB-сторінці подайте відомості про розробника. Діяльність підприємства – торгівля побутовою технікою.

5. Розробіть дві WEB-сторінки мовою HTML торговельного підприємства та автора. Перша WEB-сторінка має містити інформацію про підприємство (назву, адресу, телефони офісу, час роботи) та прайс-листок підприємства. На другій WEB-сторінці подайте відомості про розробника. Діяльність підприємства – торгівля мобільними пристроями.

6. Розробіть дві WEB-сторінки мовою HTML торговельного підприємства та автора. Перша WEB-сторінка має містити інформацію про підприємство

(назву, адресу, телефони офісу, час роботи) та прайс-листок підприємства. На другій WEB-сторінці подайте відомості про розробника. Діяльність підприємства – торгівля прикрасами.

7. Розробіть дві WEB-сторінки мовою HTML торговельного підприємства та автора. Перша WEB-сторінка має містити інформацію про підприємство (назву, адресу, телефони офісу, час роботи) та прайс-листок підприємства. На другій WEB-сторінці подайте відомості про розробника. Діяльність підприємства – торгівля торговельним обладнанням.

8. Розробіть дві WEB-сторінки мовою HTML торговельного підприємства та автора. Перша WEB-сторінка має містити інформацію про підприємство (назву, адресу, телефони офісу, час роботи) та прайс-листок підприємства. На другій WEB-сторінці подайте відомості про розробника. Діяльність підприємства – торгівля продовольчими товарами.

9. Розробіть дві WEB-сторінки мовою HTML торговельного підприємства та автора. Перша WEB-сторінка має містити інформацію про підприємство (назву, адресу, телефони офісу, час роботи) та прайс-листок підприємства. На другій WEB-сторінці подайте відомості про розробника. Діяльність підприємства – торгівля одягом.

10. Розробіть дві WEB-сторінки мовою HTML торговельного підприємства та автора. Перша WEB-сторінка має містити інформацію про підприємство (назву, адресу, телефони офісу, час роботи) та прайс-листок підприємства. На другій WEB-сторінці подайте відомості про розробника. Діяльність підприємства – торгівля радіоприладами.

Контрольні питання

1. Що таке HTML?
2. Які інструменти необхідні для роботи з мовою HTML?
3. Які теги використовуються для форматування тексту?
4. Як створити таблицю?
5. Які елементи використовуються для роботи із зображеннями?

Література: [1–7].

Лабораторна робота № 2

Тема. Візуалізація WEB-сторінок HTML за допомогою CSS

Мета: ознайомлення та вивчення принципів візуалізації WEB-сторінок за допомогою CSS.

Короткі теоретичні відомості

1. Основи CSS3

Каскадні таблиці стилів (Cascading Style Sheets) або просто CSS визначають уявлення документа, його зовнішній вигляд. Стил у CSS являє правило, яке вказує WEB-браузеру, як треба формувати елемент. Форматування може включати установку кольору фону елемента, установку кольору і типу шрифту і так далі.

Визначення стилю складається з двох частин: селектор, який указує на елемент, і блок оголошення стилю – набір команд, які встановлюють правила форматування. Наприклад:

```
div{  
    background-color:red;  
    width: 100px;  
    height: 60px;  
}
```

У цьому випадку селектором є div. Цей селектор указує, що цей стиль буде застосовуватися до всіх елементів div. Після селектора у фігурних дужках йде блок оголошення стилю. Між відкритими і закритими фігурними дужками визначаються команди, які вказують, як формувати елемент.

Кожна команда складається з властивості й значення. Так, у наступному виразі:

```
background-color:red;
```

background-color є властивістю, а red – значенням.

Існують різні способи визначення стилів.

Перший спосіб полягає у встановленні стилів безпосередньо в елемент за допомогою атрибута style:


```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Стили</title>
  </head>
  <body>
    <h2 style="color:blue;">Стили</h2>
    <div style="width: 100px; height: 100px;
background-color: red;"></div>
  </body>
</html>

```

Стили



Рисунок 2.1 – Результат роботи коду

Другий спосіб полягає у використанні елемента style в документі HTML. Цей елемент повідомляє браузеру, що дані всередині є кодом CSS, а не HTML:

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Стили</title>
    <style>
      h2{
        color:blue;

```

```

    }
    div{
        width: 100px;
        height: 100px;
        background-color: red;
    }
</style>
</head>
<body>
    <h2>Стили</h2>
    <div></div>
</body>
</html>

```

Другий спосіб робить код HTML чистіше унаслідок винесення стилів в елемент style. Але також є і третій спосіб, який полягає у винесенні стилів у зовнішній файл. Створимо в одній папці з HTML сторінкою текстовий файл, який перейменуємо в styles.css і визначимо в ньому такий вміст:

```

h2{
    color:blue;
}
div{
    width: 100px;
    height: 100px;
    background-color: red;
}

```

Це ті ж стилі, що були в середині елемента style. І також змінимо код HTML-сторінки:

```

<!DOCTYPE html>
<html>
    <head>

```

```
<meta charset="utf-8">
<title>Стили</title>
<link rel="stylesheet" type="text/css"
href="styles.css"/>
</head>
<body>
    <h2>Стили</h2>
    <div></div>
</body>
</html>
```

Отже, визначаючи стилі в зовнішньому файлі, ми робимо код HTML чистіше, структура сторінки відділяється від її стилізації. За таким визначенням стилі набагато легше модифікувати, ніж якби вони були визначені в середині елементів або в елементі style, і такий спосіб є кращим у HTML5. Використання стилів у зовнішніх файлах дозволяє зменшити навантаження на WEB-сервер за допомогою механізму кешування. Оскільки WEB-браузер може кешувати css-файл і за наступним зверненні до WEB-сторінки витягувати потрібний CSS-файл з кешу.

2. Селектори. Селектор елемента

Визначення стилю починається із селектора. Наприклад:

```
div{
    width:50px; /* ширина */
    height:50px; /* висота */
    background-color:red; /* колір фону */
    margin: 10px; /* відступ від інших елементів */
}
```

У цьому випадку селектором є div. Низка селекторів успадковують назву елементів, що форматуються, наприклад, div, p, h2 і т. д. У разі визначення такого селектора його стиль буде застосовуватися до всіх елементів

відповідних цьому селектору. Тобто вище певний стиль буде застосовуватися до всіх елементів `<div>` на WEB-сторінці.

3. Класи

Іноді для одних і тих же елементів потрібна різна стилізація. І в цьому випадку ми можемо використовувати класи. Для визначення селектора класу в CSS перед назвою класу ставиться крапка:

```
.redBlock{  
    background-color:red;  
}
```

Після визначення класу ми можемо його застосувати до елементу за допомогою атрибута `class`. Наприклад:

```
<div class="redBlock"></div>
```

4. Ідентифікатори

Для ідентифікації унікальних на WEB-сторінці елементів використовуються ідентифікатори, які визначаються за допомогою атрибутів `id`. Наприклад, на сторінці може бути головний блок або шапка:

```
<div id="header"></div>
```

Визначення стилів для ідентифікаторів аналогічно визначенню класів, тільки замість точки ставиться символ решітки `#`.

5. Універсальний селектор

Окрім селектора тегів, класів і ідентифікаторів у CSS також є так званий універсальний селектор, який є знаком зірочки (*). Він застосовує стилі до всіх елементів на HTML-сторінці:

```
*{  
    background-color: red;  
}
```

6. Псевдоелементи

Псевдоелементи мають низку додаткових можливостей з вибору елементів WEB-сторінки і схожі на псевдокласи. Список доступних псевдоелементів:

- :: first-letter: дозволяє вибрати першу букву з тексту;
- :: first-line: стилізує перший рядок тексту;
- :: before: додає повідомлення до певного елемента;
- :: after: додає повідомлення після певного елемента;
- :: selection: вибирає вибрані користувачем елементи.

Стилізуємо текст, використовуючи псевдоелементи first-letter і first-line:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Псевдоклассы в CSS3</title>
    <style>
      ::first-letter { color:red; font-size: 25px;}
      ::first-line { font-size: 20px; }
    </style>
  </head>
  <body>
    <p>Но он ничего не видал. Над ним не было ничего
уже, кроме неба, — высокого неба, не ясного, но все-таки
неизмеримо высокого, с тихо ползущими по нем серыми
облаками.</p>
  </body>
</html>
```

Но он ничего не видал. Над ним не было ничего уже, кроме неба, — высокого неба, не ясного, но все-таки неизмеримо высокого, с тихо ползущими по нем серыми облаками.

Рисунок 2.2 – Результат роботи коду

7. Псевдокласи

На додаток до селекторів тегів, класів і ідентифікаторів нам доступні селектори псевдокласів, які несуть додаткові можливості з вибору потрібних елементів.

Список доступних псевдокласів:

- root: дозволяє вибрати кореневий елемент WEB-сторінки;
- link: застосовується до посилань і є посиланням у звичайному стані, за яким ще не здійснено перехід;
- visited: застосовується до посилань і є посиланням, за яким користувач уже переходив;
- active: застосовується до посилань і є посиланням у той момент, коли користувач здійснює за ними перехід;
- hover: є елементом, на який користувач навів покажчик миші. Застосовується переважно до посилань, однак може також застосовуватися і до інших елементів, наприклад, до параграфів;
- focus: є елементом, який отримує фокус, тобто коли користувач натискає кнопку табуляції або натискає кнопкою миші на полі введення (наприклад, текстове поле) ;
- not: дозволяє виключити елементи зі списку елементів, до яких застосовується стиль;
- lang: стилізує елементи на підставі значення атрибута lang;
- empty: вибирає елементи, які не мають вкладених елементів, тобто є порожніми.

8. Властивості у CSS

За допомогою властивостей CSS можна змінювати розміри, колір елементів, формувати текст, задавати границі, внутрішні та зовнішні відступи елементів.

У CSS широке поширення знаходить використання кольорів. Щоб встановити колір тексту, фону або границі, нам треба вказати колір.

Наприклад, визначимо червоний колір для фону елемента div:

```
div{  
    background-color: red;  
}
```

Низка налаштувань кольору дозволяють установити значення для альфа-компоненти, яка відповідає за прозорість. Але також у CSS є спеціальна властивість, яка дозволяє встановити прозорість елементів – властивість `opacity`. Як значення вона приймає число від 0 (повністю прозорий) до 1 (непрозорими):

```
div{  
    width: 100px;  
    height: 100px;  
    background-color: red;  
    opacity: 0.4;  
}
```

Розміри елементів задаються за допомогою властивостей `width` (ширина) і `height` (висота). Значення за замовчуванням для цих властивостей – `auto`, тобто браузер сам визначає ширину і висоту елемента. Можна також явно задати розміри за допомогою одиниць виміру (пікселів, `em`) або за допомогою відсотків:

```
width: 150px;  
width: 75%;  
height: 15em;
```

Для налаштування границі можуть використовуватися відразу кілька властивостей: `border-width`: установлює ширину кордону, `border-style`: задає стиль лінії границі, `border-color`: установлює колір границі.

Властивість `border-style` оформляє тип лінії границі й може приймати одне з таких значень:

- `none`: межа відсутня;
- `solid`: межа у вигляді звичайної лінії;
- `dashed`: штрихова лінія;
- `dotted`: лінія у вигляді послідовності точок;
- `double`: межа у вигляді двох паралельних ліній;
- `groove`: межа має тривимірний ефект;

- inset: межа як би вдавлюється всередину;
- outset: аналогічно inset, тільки межа як би виступає назовні;
- ridge: межа також реалізує тривимірний ефект.

Наприклад:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Блочная модель в CSS3</title>
    <style>
      div{
        width: 100px;
        height:100px;
        border-style: solid;
        border-color: red;
        border-width: 2px;
      }
    </style>
  </head>
  <body>
    <div></div>
  </body>
</html>
```



Рисунок 2.3 – Результат роботи коду

Фон елемента описується в CSS властивістю `background`. Фактично ця властивість є скороченням набору таких властивостей CSS:

- background-color установлює колір фону: background-color: #ff0507;
- background-image як фон установлює зображення: background-image: url(dubi.png);
- background-repeat: установлює режим повторення фонового зображення по всій поверхні елемента;
- background-size: установлює розмір фонового зображення;
- background-position: указує позицію фонового зображення;
- background-attachment: установлює стиль закріплення фонового зображення до елемента;
- background-clip: визначає область, яка вирізається із зображення та використовується як фон;
- background-origin: установлює початкову позицію фонового зображення.

Наприклад:

```
<!DOCTYPE html>

<html>

  <head>

    <meta charset="utf-8">

    <title>Блочная модель в CSS3</title>

    <style>

      div{

        width: 250px;

        height: 200px;

        margin: 10px;

      }

      .colored{

        background-color: #ff0507;

      }

    </style>

  </head>

  <body>

    <div class="colored">

      <div>Блочная модель в CSS3</div>

    </div>

  </body>

</html>
```

```

        .imaged{
            background-image: url(dubi.png);
        }
    </style>
</head>
<body>
    <div class="colored">Первый блок</div>
    <div class="imaged">Второй блок</div>
</body>
</html>

```

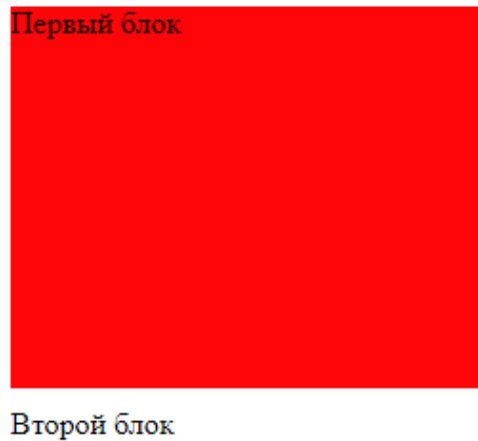


Рисунок 2.4 – Результат работы коду

Властивість box-shadow дозволяє створити у елемента тінь. Приклад:

```

<style>
    div{
        width: 128px;
        height: 96px;
        margin: 20px;
        border: 1px solid #ccc;
        background-color: #eee;
        box-shadow: 10px 4px 10px 3px #888;
    }

```

</style>

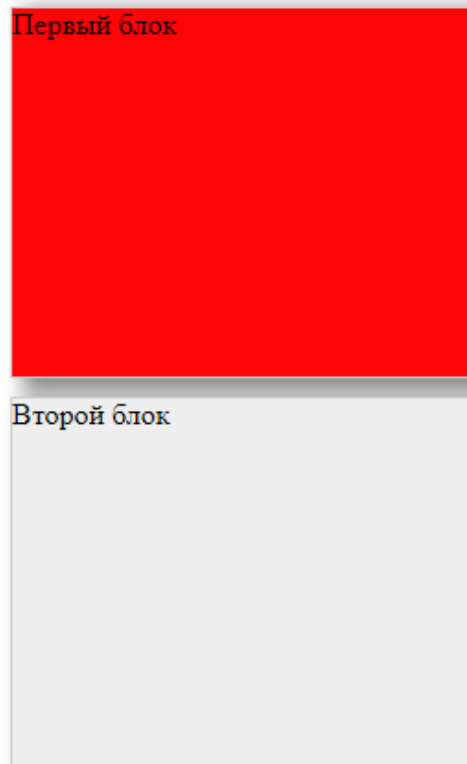


Рисунок 2.5 – Результат роботи коду

Властивість `text-transform` змінює регістр тексту. Воно може набувати таких значень:

- `capitalize`: робить першу букву слова великою;
- `uppercase`: усе слово перекладається до верхнього регістру;
- `lowercase`: усе слово перекладається до нижнього регістру;
- `none`: регістр символів слова ніяк не змінюється.

Властивість `text-decoration` дозволяє додати до тексту деякі додаткові ефекти. Це властивість може набувати таких значень:

- `underline`: підкреслює текст;
- `overline`: надкреслює текст, проводить верхню лінію;
- `line-through`: закреслює текст;
- `none`: до тексту не застосовується декорування.

Приклад форматування тексту:

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Шрифты в CSS3</title>
    <style>
      p.under {
        text-decoration: underline;
      }
      p.over {
        text-decoration: overline;
      }
      p.line {
        text-decoration: line-through;
      }
      p.mixed {
        text-decoration: underline line-through;
      }
      a.none {
        text-decoration: none;
      }
    </style>
  </head>
  <body>
    <div>
      <p class="under">Это подчеркнутый текст.</p>
      <p class="over">Это надчеркнутый текст</p>
      <p class="line">Это зачеркнутый текст</p>
      <p class="mixed">Это подчеркнутый и зачеркнутый
текст</p>
    </div>
  </body>
</html>

```

```
<p>Не подчеркнутая <a href="index.php"
class="none">ссылка<a></p>
</div>
</body>
</html>
```

Это подчеркнутый текст.

Это надчеркнутый текст

~~Это зачеркнутый текст~~

~~Это подчеркнутый и зачеркнутый текст~~

Не подчеркнутая [ссылка](#)

Рисунок 2.6 – Результат работы коду

Нерідко під час створення WEB-сторінок можна зіткнутися із ситуацією, коли вміст блоку займає набагато більше місця, ніж визначено шириною і висотою блоку. У цій ситуації за замовчуванням браузер усе одно відображає вміст, навіть якщо він виходить за межі блоку.

Однак властивість `overflow` дозволяє налаштувати поведінку блоку в подібній ситуації і додати можливість прокрутки. Ця властивість може набувати таких значень:

- `auto`: якщо контент виходить за межі блоку, то створюється прокрутка. В інших випадках смуги прокрутки не відображаються;
- `hidden`: відображається лише видима частина контенту. Контент, який виходить за межі блоку, не відображається, а смуги прокрутки не створюються;
- `scroll`: у блоці відображаються смуги прокрутки, навіть якщо контент весь поміщається в межах блоку і таких смуг прокрутки не потрібно;
- `visible`: значення за замовчуванням, контент відображається, навіть якщо він виходить за межі блоку.

Розглянемо застосування двох значень:

```
<!DOCTYPE html>
```

```

<html>
  <head>
    <meta charset="utf-8">
    <title>Прокрутка в CSS3</title>
    <style>
      .article1{
        width: 300px;
        height: 150px;
        margin:15px;
        border: 1px solid #ccc;
        overflow: auto;
      }
      .article2{
        width: 300px;
        height: 150px;
        margin:15px;
        border: 1px solid #ccc;
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div class="article1">
      <p>Старый дуб, весь преображенный, раскинувшись шатром
сочной, темной зелени, млел, чуть колыхаясь в лучах
вечернего солнца. Ни корявых пальцев, ни болячек, ни
старого недоверия и горя – ничего не было видно. Да, это
тот самый дуб», – подумал князь Андрей, и на него вдруг
нашло беспричинное весеннее чувство радости и
обновления.</p>
    </div>
  </body>
</html>

```

```

</div>

<div class="article2">
<p>Старый дуб, весь преображенный, раскинувшись шатром
сочной, темной зелени, млел, чуть колыхаясь в лучах
вечернего солнца. Ни корявых пальцев, ни болячек, ни
старого недоверия и горя – ничего не было видно. Да, это
тот самый дуб», – подумал князь Андрей, и на него вдруг
нашло беспричинное весеннее чувство радости и
обновления.</p>
</div>
</body>
</html>

```

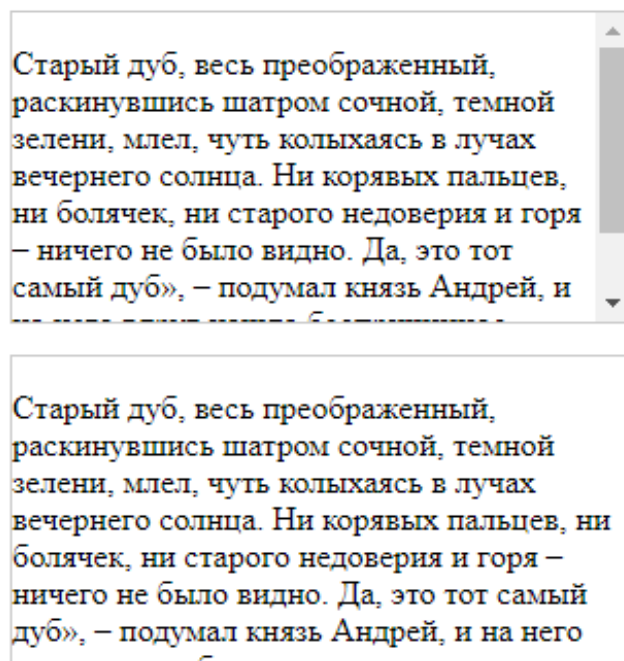


Рисунок 2.7 – Результат работы коду

Завдання до самостійної роботи

Розробити CSS-версію WEB-сторінок мовою HTML, які були створені на попередній лабораторній роботі.

Контрольні питання

1. Що таке CSS?
2. З яких частин складається визначення стилю CSS?
3. Які селектори існують у CSS?
4. Які властивості існують у CSS?
5. Для чого призначена властивість text-transform у CSS?

Література: [1–7].

Лабораторна робота № 3

Тема. Використання блочної верстки для створення макету WEB-сторінки.

Мета: ознайомлення та вивчення принципів використання блочної верстки для створення макету WEB-сторінки.

Короткі теоретичні відомості

1. Блокова модель

Для WEB-браузера елементи сторінки є невеликими контейнерами, або блоками. Такі блоки можуть мати різний вміст – текст, зображення, списки, таблиці та інші елементи. Внутрішні елементи блоків самі є блоками.

Схематично блокову модель можна надати таким чином:

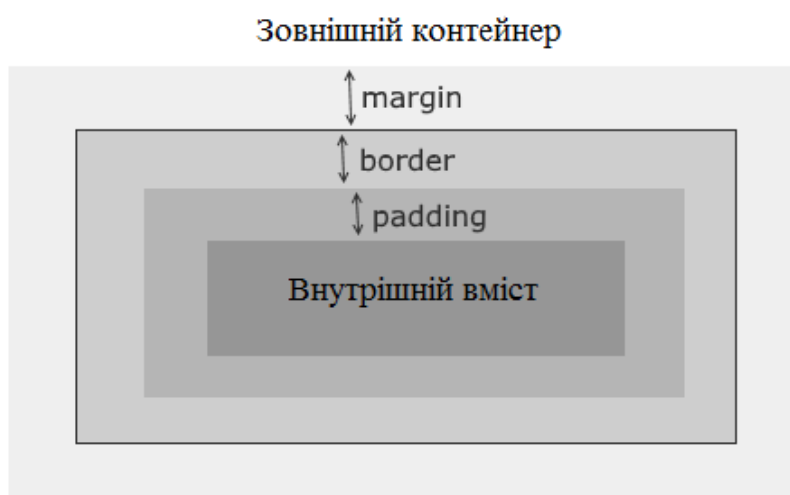


Рисунок 3.1 – Схема блокової моделі

Нехай елемент розташований у якомусь зовнішньому контейнері. Це може бути елемент `body`, `div` і так далі. Від інших елементів він відділяється деякою відстанню – зовнішнім відступом, яке описується властивістю CSS **margin**. Тобто властивість `margin` визначає відстань від границі поточного елемента до інших сусідніх елементів або до границь зовнішнього контейнера.

Далі починається сам елемент. І на початку йде його межа, яка в CSS описується властивістю **border**.

Після границі йде внутрішній відступ, який у CSS описується властивістю **padding**. Внутрішній відступ визначає відстань від границі елемента до внутрішнього вмісту.

Далі йде внутрішній вміст, який також реалізує ту ж блокову модель і також може складатися з інших елементів, які мають зовнішні й внутрішні відступи та границю.

2. Блокова верстка

Зазвичай WEB-сторінка складається з безлічі різних елементів, які можуть мати складну структуру. Тому під час створення WEB-сторінки виникає необхідність позиціонувати ці елементи, стилізувати їх так, щоб вони розташовувалися на сторінці, як належить. Тобто виникає питання створення макета сторінки, її верстки.

Існують різні способи, стратегії і види верстки. Спочатку поширеною була верстка на основі таблиць, оскільки таблиці дозволяють за необхідності дуже легко і просто розділити весь простір WEB-сторінки на рядки і стовпці. Рядками і стовпцями досить легко управляти, в них легко позиціонувати будь-який вміст. Саме це і визначило популярність табличної верстки.

Однак таблична верстка створює не самі гнучкі за дизайном сторінки, що є особливо актуальним аспектом у світі, де немає одного єдиного дозволу екрану, проте є великі екрани на телевізорах, малі екрани на планшетах, дуже маленькі екрани на смартфонах і т. д. Усе це різноманіття екранів таблична верстка не мала можливості задовольнити. Тому поступово їй на зміну прийшла блокова верстка. Блокова верстка – це відносно умовна назва способів і

приймів верстки, коли в більшості WEB-сторінок для розмітки використовується CSS-властивість `float`, а основним будівельним елементів WEB-сторінок є елемент `<div>`, тобто по суті блок. Використовуючи властивість `float` і елементи `div` або інші елементи, можна створити структуру сторінки з декількох стовпців, як під час табличної верстки, яка буде значно гнучкіша.

Приклад використання властивості **`float`** для створення двоколонкової WEB-сторінки. Припустимо, вгорі й внизу у нас будуть стандартно шапка і футер, а в центрі – дві колонки: колонка з меню або сайдбар і колонка з основним вмістом.

Тепер використовуємо `float` для створення двоколонкової WEB-сторінки. Припустимо, вгорі й внизу у нас будуть стандартно шапка і футер, а в центрі – дві колонки: колонка з меню, або сайдбар, і колонка з основним вмістом:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Блочная верстка в HTML5</title>
    <style>
      div{
        margin: 10px;
        border: 1px solid black;
        font-size: 20px;
        height: 80px;
      }
      #header{
        background-color: #ccc;
      }
      #sidebar{
        background-color: #ddd;
      }
      #main{
```

```

        background-color: #eee;
        height: 200px;
    }
    #footer{
        background-color: #ccc;
    }
</style>
</head>
<body>
    <div id="header">Шапка сайта</div>
    <div id="sidebar">Сайдбар</div>
    <div id="main">Основное содержимое</div>
    <div id="footer">Футер</div>
</body>
</html>

```

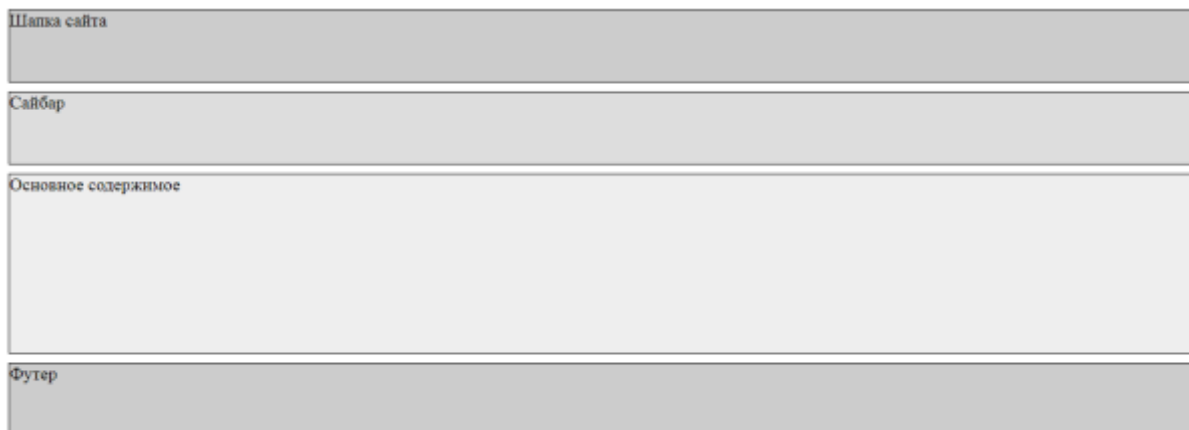


Рисунок 3.2 – Результат роботи коду

Далі, щоб перемістити блок сайдбара вліво відносно блоку основного вмісту й отримати ефект обтікання, нам треба вказати у блоку сайдбара властивість `float: left` і переважну ширину. Ширина може бути фіксованою, наприклад, 150 px або 8 em. Або також можна використовувати відсотки, наприклад, 30 % – 30 % від ширини контейнера `body`. З одного боку, блоками з фіксованою шириною легше управляти, але з іншого – процентні значення

ширини дозволяють створювати більш гнучкі, гумові блоки, які змінюють розміри у разі зміни розмірів вікна браузера.

Останнім кроком є установка відступу блоку з основним вмістом від блоку сайдбара. Оскільки під час обтікання оточувальний блок може обтікати плавальний елемент і праворуч, і знизу, якщо плавальний елемент має меншу висоту, то нам треба встановити відступ, як мінімум рівний ширині плавального елемента. Наприклад, якщо ширина сайдбара дорівнює 150 px, то для блоку основного вмісту можна задати відступ у 170 px, що дозволить створити порожній простір між двома блоками.

Ураховуючи все вище сказане, змінимо стилі блоків сайдбара й основного вмісту в такий спосіб:

```
#sidebar{
    background-color: #ddd;
    float: left;
    width: 150px;
}
#main{
    background-color: #eee;
    height: 200px;
    margin-left: 170px; /* 150px (ширина сайдбара) + 10px
+ 10px (2 отступа) */
}
```

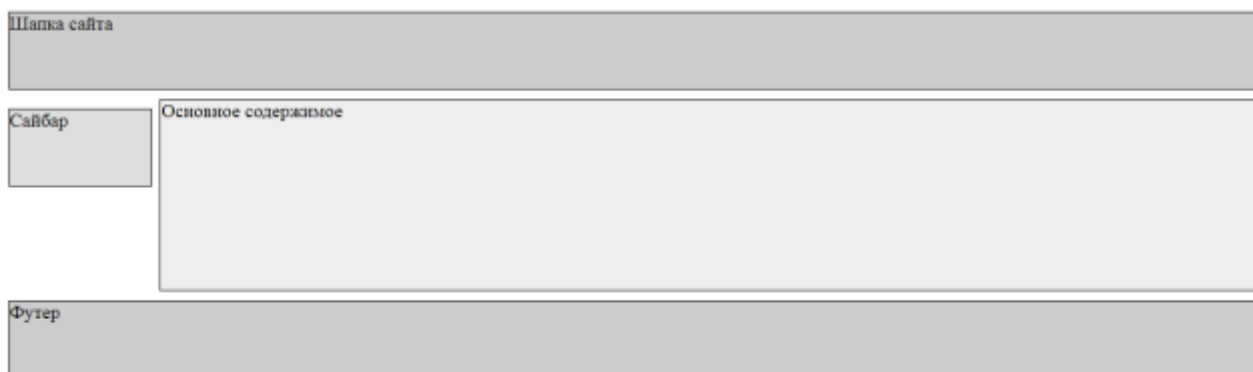


Рисунок 3.3 – Результат роботи коду

Унаслідок цього у нас вийде сайдбар з лівої сторони від основного блоку.

Створення правого сайдбара буде аналогічно, тільки тепер нам треба встановити у сайдбара значення `float: right`, а у блоку основного вмісту – відступ справа:

```
#sidebar{
    background-color: #ddd;
    float: right;
    width: 150px;
}
#main{
    background-color: #eee;
    height: 200px;
    margin-right: 170px;
}
```

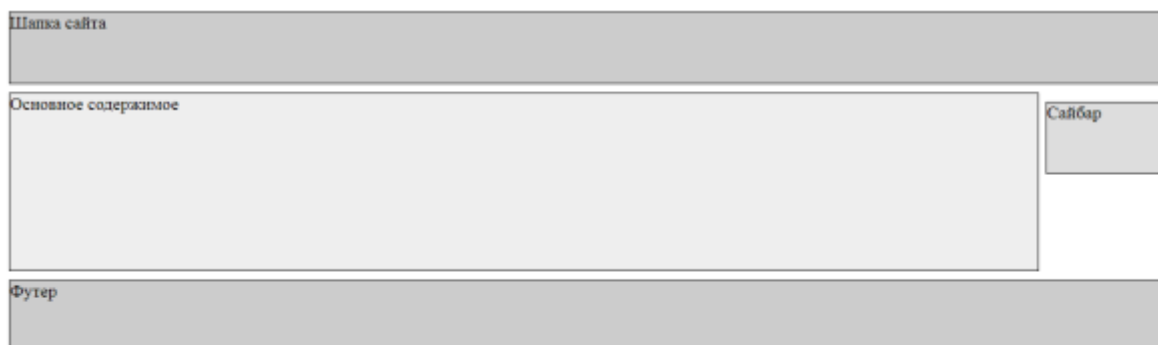


Рисунок 3.4 – Результат роботи коду

3. Властивість **display**

Окрім властивості `float`, яка дозволяє змінювати позицію елемента, в CSS є ще одна важлива властивість – **`display`**. Вона дозволяє управляти блоком елемента і також впливати на його позиціонування щодо сусідніх елементів.

Ця властивість може набувати таких значень:

- `inline`: елемент стає рядковим, подібно словам в рядку тексту;
- `block`: елемент стає блоковим, як параграф;
- `inline-block`: елемент розташовується, як рядок тексту;

- list-item: елемент позиціонується як елемент списку зазвичай з додаванням маркера у вигляді точки або порядкового номера;
- run-in: тип блоку елемента залежить від навколишніх елементів;
- flex: дозволяє здійснювати гнучке позиціонування елементів;
- table, inline-table: дозволяє розташувати елементи у вигляді таблиці;
- none: елемент не видно і він видалений з розмітки HTML.

Отже, значення block дозволяє визначити блоковий елемент. Такий елемент візуально відділяється від сусідніх елементів перенесенням рядка, як, наприклад, елемент параграфа p або елемент div, які за умовчанням є блоковими і під час візуалізації WEB-сторінки візуально переносяться на новий рядок.

Однак елемент span на відміну від елемента div не є блоковим. Тому подивимося, які з ним відбудуться зміни у разі застосування значення block:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <link href="styles.css" rel="stylesheet">
    <title>Свойство display в CSS3</title>
    <style>
      span{
        color: red;
      }
      .blockSpan{
        display: block;
      }
    </style>
  </head>
  <body>
    <div>Это <span>строчный</span> элемент span</div>
```

```

        <div>Это <span class="blockSpan">блочный</span>
элемент span</div>
    </body>
</html>

```

Это строчный элемент span
 Это
 блочный
 элемент span

Рисунок 3.5 – Результат работы коду

Ще одне значення – inline-block – є елементом, який володіє ознаками блочного і строкового елементів. Відносно сусідніх зовнішніх елементів такий елемент розцінюється як рядковий. Тобто він не відділяється від сусідніх елементів переводом рядка. Однак відносно до вкладених елементів він розглядається як блоковий. І до такого елемента застосовуються властивості width, height, margin.

```

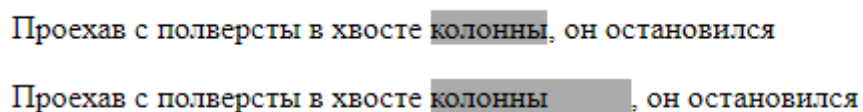
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Свойство display в CSS3</title>
        <style>
            span{
                width: 100px;
                height: 30px;
                background-color: #aaa;
            }
            .inlineBlockSpan{
                display: inline-block;
            }
        </style>

```

```

</head>
<body>
    <p>Проехав с полверсты в хвосте
<span>колонны</span>, он остановился</p>
    <p>Проехав с полверсты в хвосте <span
class="inlineBlockSpan">колонны</span>, он остановился</p>
</body>
</html>

```



Проехав с полверсты в хвосте колонны, он остановился

Проехав с полверсты в хвосте колонны, он остановился

Рисунок 3.6 – Результат работы коду

Значення table по суті перетворює елемент у таблицю. Приклад списку:

```

<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Свойство display в CSS3</title>
        <style>
            ul{
                display: table;
                margin: 0;
            }
            li{
                list-style-type: none;
                display: table-cell;
                padding: 10px;
            }
        </style>

```



```

</head>
<body>
  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
  </ul>
</body>
</html>

```

Item 1 Item 2 Item 3

Рисунок 3.7 – Результат роботи коду

Завдання до самостійної роботи

Створити WEB-сайт за поданим макетом на самостійно обрану тему, використовуючи властивості, описані у теоретичній частині.

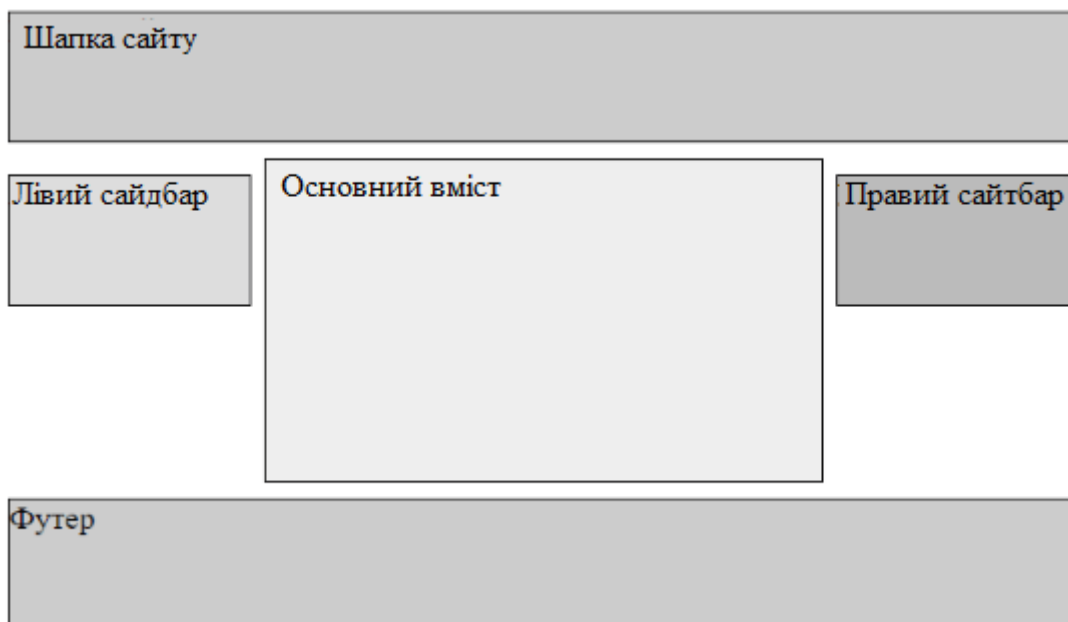


Рисунок 3.8 – Макет WEB-сайту для завдання

Контрольні питання

1. Які існують способи верстки макету?

2. Що собою являє блокова модель?
3. Які властивості CSS використовуються для блокової верстки?
4. Яких значень може набувати властивість display?
5. Як змінює елемент значення table властивості display?

Література: [1–7].

Лабораторна робота № 4

Тема. Створення адаптивного дизайну

Мета: ознайомлення та вивчення принципів створення адаптивного дизайну WEB-сторінок.

Короткі теоретичні відомості

1. Адаптивний дизайн

Зараз більшого поширення знаходять різні гаджети – смартфони, планшети, «розумний годинник» та інші пристрої, які дозволяють виходити в інтернет, переглядати вміст сайтів. За деяким оцінюванням уже мало не половина інтернет-трафіку генерується подібними гаджетами, дозвіл екрана яких відрізняється від дозволу екранів стандартних комп'ютерів. Подібне поширення гаджетів несе нові можливості з розвитку WEB-сайтів, залучення нових відвідувачів, просуванню інформаційних послуг і т. д. Але разом з тим з'являються і нові проблеми.

Головна проблема полягає в тому, що стандартна WEB-сторінка буде по-різному виглядати для різних пристроїв з різним дозволом екрану. Первинним розв'язанням цієї проблеми було створення спеціальних версій для мобільних пристроїв. На початку поширення мобільних телефонів користувачі могли через телефон по протоколу WAP отримувати доступ до спеціальних war-сайтів, які були написані на мові WML – мовою на основі XML, схожею з HTML.

Однак розвиток самих гаджетів, їх можливостей призвело до того, що зараз мобільні телефони надають куди більші можливості з отримання та відображення вмісту сайтів, а в написанні подібних сайтів використовуються ті

ж HTML5 та CSS3, що і для звичайних сайтів. Окрім того, поява все більшої кількості різноманітних пристроїв призвело до того, що WEB-сторінки необхідно підлаштовувати не тільки під невеликі екрани смартфонів або планшетів, але і під величезні екрани повноформатних широкоекранних телевізорів або гігантських планшетів типу Surface Hub, які також можуть мати доступ до інтернету.

І для розв'язання проблеми сумісності WEB-сторінок із самими різними дозволами самих різних пристроїв виникла концепція адаптивного дизайну. Її суть полягає в тому, щоб належно масштабувати елементи WEB-сторінки залежно від ширини екрану.

2. Метатег Viewport

Видима область на екрані браузера описується поняттям Viewport. По суті viewport є областю, в яку WEB-браузер намагається «вставити» WEB-сторінку. Наприклад, браузер Safari на iPhone і iPod визначає ширину viewport, що за замовчуванням дорівнює 980 пікселям. Тобто, отримавши сторінку і вписавши у viewport з шириною 980 пікселів, браузер стискає її до розмірів мобільного пристрою. Наприклад, якщо ширина екрану смартфона становить 320 пікселів, то до цього розміру потім буде стиснута сторінка. І до всіх елементів сторінки буде застосовано коефіцієнт масштабування, що дорівнює 320/980.

Щоб уникнути подібної не надто приємною картини, слід використовувати метатег viewport. Він має таке визначення:

```
<meta name="viewport" content="параметри_метатега">
```

В атрибуті content мета-тега ми можемо визначити такі параметри.

Таблиця 4.1 – Параметри метатега viewport

Параметр	Значення	Опис
width	Приймає цілочисельне значення в пікселях або значення device-width	Установлює ширину області viewport
height	Приймає цілочисельне значення в пікселях або значення device-height	Установлює висоту області viewport

Продовження таблиці 4.1

initial-scale	Число з плавальною точкою від 0.1 та вище	Задає коефіцієнт масштабування початкового розміру viewport. Значення 1.0 задає відсутність масштабування
user-scalable	no/yes	Указує, чи може користувач за допомогою жестів масштабувати сторінку
minimum-scale	Число с плавальною точкою від 0.1 та вище	Задає мінімальний масштаб розміру viewport. Значення 1.0 задає відсутність масштабування
maximum-scale	Число с плавальною точкою від 0.1 та вище	Задає максимальний масштаб розміру viewport. Значення 1.0 задає відсутність масштабування

3. Media Query

Іншим важливим елементом в побудові адаптивного дизайну є правила Media Query, які дозволяють визначити стиль залежно від розмірів браузера користувача. Наприклад, щоб застосувати стиль лише до мобільних пристроїв, ми можемо написати так:

```
<html>
  <head>
    <title>Адаптивная WEB-сторінка</title>
    <meta name="viewport" content="width=device-width">
    <link rel="stylesheet" type="text/css"
href="desktop.css" />
    <link rel="stylesheet" type="text/css" media="(max-
device-width:480px)" href="mobile.css" />
  </head>
```

<body>

Значення атрибута `media (max-device-width: 480px)` говорить нам про те, що стилі з файлу `mobile.css` будуть застосовуватися до тих пристроїв, максимальна ширина екрану яких становить 480 пікселів – тобто фактично це і є мобільні пристрої. За допомогою ключового слова `and` можна комбінувати умови, наприклад:

```
<link rel="stylesheet" type="text/css" media="(min-width:481px) and (max-width:768px)" href="mobile.css" />
```

Цей стиль буде застосовуватися, якщо ширина браузера знаходиться в діапазоні від 481 до 768 пікселів.

За допомогою директиви `@import` можна визначити один `css`-файл і імпортувати в нього стилі для певних пристроїв:

```
- @import url(desktop.css);  
- @import url(tablet.css) (min-device-width:481px)  
and (max-device-width:768);  
- @import url(mobile.css) (max-device-width:480px);
```

Також можна не розділяти стилі за файлами, а використовувати правила **CSS3 Media Query** в одному файлі `css`:

```
body {  
    background-color: red;  
}  
/*Далі інші стилі*/  
@media (max-device-width:480px) {  
    body {  
        background-color: blue;  
    }  
    /*Далі інші стилі*/  
}
```

Функції, що застосовуються у **CSS3 Media Query**:

– `aspect-ratio`: відношення ширини до висоти області відображення (браузера);

- device-aspect-ratio: відношення ширини до висоти екрану пристрою;
- max-width / min-width і max-height / min-height: максимальна і мінімальна ширина і висота області відображення (браузера);
- max-device-width / min-device-width і max-device-height / min-device-height: максимальна і мінімальна ширина і висота екрану мобільного пристрою;
- orientation: орієнтація (портретна або альбомна).

Наприклад, ми можемо задати різні стилі для різних орієнтацій мобільних пристроїв:

```
/*Стилі для портретної орієнтації*/
@media only screen and (orientation: portrait){
    }
/*Стилі альбомної орієнтації*/
@media only screen and (orientation: landscape){
    }
```

Завдання до самостійної роботи

Розробити адаптивний дизайн WEB-сайту, створеного на попередній лабораторній роботі.

Контрольні питання

1. Що таке адаптивний дизайн та які проблеми він розв'язує?
2. Які елементи дозволяють зробити дизайн адаптивним?
3. Що таке Media Query?
4. Які функції застосовуються у CSS3 Media Query?
5. Як задати різні стилі для різних орієнтацій мобільних пристроїв за допомогою Media Query?

Література: [1–7].

Лабораторна робота № 5

Тема. Застосування CSS-фреймворків для створення WEB-сторінок

Мета: ознайомлення та вивчення принципів створення WEB-сторінок, використовуючи CSS-фреймворки.

Короткі теоретичні відомості

CSS, або каскадні таблиці стилів, – потужний інструмент, який дозволяє змінювати зовнішній вигляд сайту. Але головна проблема, з якою доведеться зіткнутися під час роботи з технологією CSS, – коли з’являється необхідність написання великої кількості коду. Звичайно, написання великої кількості коду може не мати великої кількості труднощів. Але, коли елементів на WEB-сайті стає багато, коду теж може ставати дуже багато. Писати таку велику кількість коду може бути не дуже доцільно.

Для того, щоб уникнути цієї великої кількості роботи і прискорити процес верстки WEB-сторінок, були створені спеціальні бібліотеки, або CSS-фреймворки. По суті, CSS-фреймворк – це просто файл CSS, який стандартно підключається до вашої WEB-сторінки.

Цей CSS файл містить уже написаний за вас набір стилів, які ви зможете застосовувати до елементів на вашому WEB-сайті після того, як ця бібліотека буде підключена до вашої WEB-сторінки. Після того, як цей файл фреймворка буде підключений до WEB-сторінки, ви можете додавати до ваших елементів не якісь конкретні стилі CSS, а просто додавати класи.

Наприклад:

```
<div class="box"></div>
```

Додавши клас, ви говорите, як CSS фреймворк має оформити той чи інший WEB-елемент.

І головне, для чого варто використовувати фреймворки, – це зберігання часу. Додати для якогось елемента певний клас набагато простіше, ніж написати десятки рядків коду. Друга перевага – стандартизація вашого коду. Якщо ви хочете зробити кнопку, ви вже знаєте, який клас вам для неї потрібно призначити. Це не залежить від того, з яким із сайтів ви працюєте. Скрізь все працює однаково.

Плюси CSS-фреймворків:

- кросбраузерність;

- можливість створити коректний HTML макет навіть не дуже досвідченому фахівцеві;

- однаковість коду;

- збільшення швидкості розробки.

Мінуси:

- залежність від стилю CSS бібліотеки;

- надмірний код.

Найбільш популярні фреймворки:

Bootstrap – цей фреймворк є наймовірно популярним і затребуваним, його представили ще на початку 2011 року. Адаптивність (адаптивна верстка) – його головна перевага. Bootstrap дозволяє створювати проекти з наймовірно адаптивним, стильним дизайном – проект буде автоматично підлаштовуватися, враховуючи розмір екрану комп'ютера або мобільного пристрою користувача, що переглядає сайт. До переваг відноситься: велика кількість стилів, шаблонів, посторінковий дизайн – це істотно полегшує створення сайту.

Bootstrap став популярним через величезну кількість переваг, у ньому практично відсутні недоліки. Це не тільки HTML / CSS-фреймворк, у Bootstrap також включені плагіни і готові стилі JS / JQuery.

Semantic UI – використовується для створення переносимих інтерфейсів. Цей фреймворк можна назвати досить молодим, проте варто відзначити його постійний розвиток. У ньому можна знайти величезну кількість кнопок і інших елементів, необхідних для роботи – зображення, іконки, написи.

Foundation – цей фреймворк є одним з популярних у сегменті front-end-фреймворків. Останні версії відрізняються поліпшеним функціоналом для сучасних мобільних пристроїв. Завдяки семантичному підходу є можливість використання SCSS, написання більш чистого коду в HTML. Цей фреймворк є ідеальним у ситуації, коли потрібно швидке прототипування.

Для підключення фреймворку необхідно додати `<link>` у `<head>`. Для стилізації елемента необхідно додати клас із підключеної бібліотеки. Приклад застосування фреймворку Bootstrap для стилізації елемента button:

```
<!DOCTYPE html>
```



```

<html>
<head>
  <meta charset="utf-8" />
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
integrity="sha384-
JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ
0Z" crossorigin="anonymous">
</head>
<body>
  <button type="button" class="btn btn-primary">Primary</button>
  <button type="button" class="btn btn-secondary">Secondary</button>
  <button type="button" class="btn btn-success">Success</button>
  <button type="button" class="btn btn-danger">Danger</button>
  <button type="button" class="btn btn-warning">Warning</button>
  <button type="button" class="btn btn-info">Info</button>
  <button type="button" class="btn btn-light">Light</button>
  <button type="button" class="btn btn-dark">Dark</button>
</body>
</html>

```

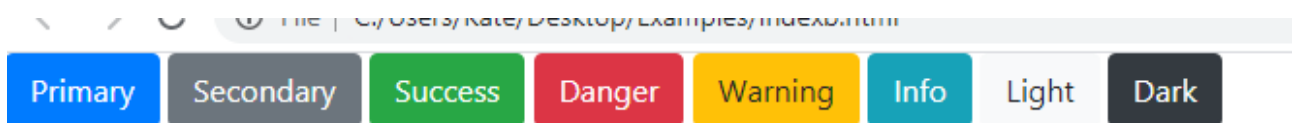


Рисунок 5.1 – Результат роботи коду

Приклад застосування елементів icons із бібліотеки Bootstrap:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>

```

```

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.6.3/css/all.css" integrity="sha384-
UHRtZLI+pbxtHCWp1t77Bi1L4ZtiqrqD80Kn4Z8NTSRyMA2Fd33n5dQ8lWUE00s
/" crossorigin="anonymous"></head>

    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></sc
ript>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>

    <body>
    <div class="container">
        <h1>My Icons <i class="fas fa-heart"></i></h1>
        <p>An icon along with some text: <i class="fas fa-thumbs-up"></i></p>
    </div>

    <div class="container">
        <p>Others:</p>
        <i class="fas fa-cloud"></i>
        <i class="fas fa-coffee"></i>
        <i class="fas fa-car"></i>
        <i class="fas fa-file"></i>
        <i class="fas fa-bars"></i>
    </div>
</body>
</html>

```

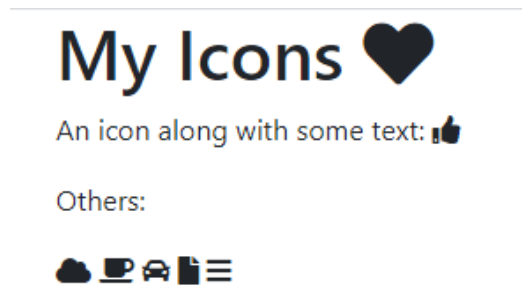


Рисунок 5.2 – Результат роботи коду

Приклад застосування каруселі, використовуючи бібліотеку Bootstrap:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link                                rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></sc
ript>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
  <style>
/* Make the image fully responsive */
.carousel-inner img {
  width: 100%;
  height: 100%;
}
</style>
```

```

</head>
<body>
<div id="demo" class="carousel slide" data-ride="carousel">
  <!-- Indicators -->
  <ul class="carousel-indicators">
    <li data-target="#demo" data-slide-to="0" class="active"></li>
    <li data-target="#demo" data-slide-to="1"></li>
    <li data-target="#demo" data-slide-to="2"></li>
  </ul>
  <!-- The slideshow -->
  <div class="carousel-inner">
    <div class="carousel-item active">
      
    </div>
    <div class="carousel-item">
      
    </div>
    <div class="carousel-item">
      
    </div>
  </div>
  <!-- Left and right controls -->
  <a class="carousel-control-prev" href="#demo" data-slide="prev">
    <span class="carousel-control-prev-icon"></span>
  </a>
  <a class="carousel-control-next" href="#demo" data-slide="next">
    <span class="carousel-control-next-icon"></span>
  </a>
</div>
</body>
</html>

```

Приклад застосування Media Objects з бібліотеки Bootstrap:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link
                                                                    rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></sc
ript>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</head>
<body>
<div class="container mt-3">
  <h2>Nested Media Objects</h2>
  <p>Media objects can also be nested (a media object inside a media
object):</p><br>
  <div class="media border p-3">
    
    <div class="media-body">
      <h4>John Doe <small><i>Posted on February 19, 2016</i></small></h4>
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua.</p>
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua.</p>
```

```

<div class="media p-3">
  
  <div class="media-body">
    <h4>Jane Doe <small><i>Posted on February 20 2016</i></small></h4>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua.</p>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua.</p>
  </div>
</div>
</div>
</div>
</div>
</div>
</body>
</html>

```



Рисунок 5.3 – Результат роботи коду

Завдання до самостійної роботи

Змінити стиль WEB-сайту, створеного на попередній лабораторній роботі, застосувавши до нього один з CSS-фреймворків.

Контрольні питання

1. Для чого призначені CSS-фреймворки?
2. Плюси та мінуси застосування CSS-фреймворків.
3. Які існують найбільш популярні CSS-фреймворки?
4. Як підключити фреймворк Bootstrap?
5. Які переваги має фреймворк Bootstrap?

Література: [1–7].

З КРИТЕРІЙ ОЦІНЮВАННЯ ЗНАНЬ СТУДЕНТІВ

Перелік лабораторних робіт складається з п'яти робіт, які спрямовані на перевірку знань студентів з теорії та практики розв'язання прикладних задач.

Виконання лабораторних робіт і написання звітів є обов'язковим для всіх студентів. За п'ять робіт студент може отримати 40 балів.

За кожную лабораторну роботу виставляють:

– **8 балів**, якщо студент виконав лабораторну роботу, написав звіт і захистив його та правильно і вичерпно відповів на всі запитання.

– **7 балів**, якщо студент виконав лабораторну роботу, написав звіт і захистив його та правильно відповів на всі питання. При цьому допускаються незначні неточності при відповіді на контрольні питання;

– **6 балів**, якщо студент виконав лабораторну роботу, написав звіт і захистив його та відповів на всі питання. При цьому допускаються помилки при відповіді на два контрольні питання;

– **5 бали** виставляється, якщо студент виконав лабораторну роботу, написав звіт і захистив його, але відповіді на контрольні питання не повні;

– **4 бали** виставляється, якщо студент виконав лабораторну роботу, написав звіт і захистив його задовільно, але частково відповів на контрольні питання викладача;

– **3 бали**, якщо студент виконав лабораторну роботу, написав звіт, але не захистив його;

– **2 бали**, якщо студент неповністю виконав лабораторну роботу;

– **1 бал**, якщо студент частково виконав лабораторну роботу;

– у решті випадків виставляють **0 балів**.

СПИСОК ЛІТЕРАТУРИ

1. Бородаєв Д. В. WEB-сайт як об'єкт графічного дизайну: монографія. Харків: Септима ЛТД, 2006. 288 с.
2. Мельник Р. А. Програмування веб-застосунків. Львів: Львівська політехніка, 2018. 248 с.
3. Роббінс Дж. Характеристики HTML 5. Кишеньковий довідник. Київ: Діалектика, 2020. 208 с.
4. Куусуль Н. М. Використання PHP. Самовчитель. Київ: Діалектика, 2006. 272 с.
5. Гончаров А. Ю. WEB-дизайн: HTML, JavaScript и CSS. Карманный справочник. Москва: КУДИЦ–ПРЕСС, 2007. 320 с.
6. Коржинский С. Н. Настольная книга WEB-мастера: эффективное применение HTML, CSS, JavaScript. Москва: КноРус, 2000. 320 с.
7. Алешин Л. И. Телекоммуникационные технологии для библиотек. Москва: Литера, 2009. 352 с.

Зразок оформлення титульної сторінки звіту

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО
КАФЕДРА АВТОМАТИЗАЦІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ЗВІТ
ІЗ ЛАБОРАТОРНИХ РОБІТ
З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ
«WEB-ДИЗАЙН»

Виконав студент групи _____

ПІБ студента

Перевірив посада, ПІБ викладача

КРЕМЕНЧУК 20__

Методичні вказівки щодо виконання лабораторних робіт з навчальної дисципліни «WEB-дизайн» для студентів денної форми навчання зі спеціальності 122 – «Комп'ютерні науки» освітньо-професійної програми «Комп'ютерні науки» освітнього ступеня «Бакалавр»

Укладачі: к. т. н., доц. І. Г. Оксанич,
асист. К. С. Король

Відповідальний за випуск старш. викл. Т. В. Горлова

Підп. до др. 19 01. 2024. Формат 60×84 1/16. Папір тип. Друк ризографія.
Ум. друк. арк. 2,75. Наклад 3 прим. Зам. № 002/5 Безкоштовно.

Редакційно-видавничий відділ
Кременчуцького національного університету
імені Михайла Остроградського
вул. Першотравнева, 20, м. Кременчук, 39600