Міністерство освіти та науки України
Чернівецький національний університет
імені Юрія Федьковича

Основи інтернет-технологій

**Методичні вказівки та завдання
до лабораторних робіт**

Чернівці
Чернівецький національний університет
2023

УДК 004.774(076.5)
О-751

Рекомендовано до друку Вченою радою факультету
математики та інформатики Чернівецького національного університету
імені Юрія Федьковича
(протокол № 6 від « 25 » січня 2023 року)

Рецензенти:
Піддубна Лариса Андріївна, доцент кафедри математичного моделювання;
Матвій Олександр Васильович, доцент кафедри математичного моделювання.

О-751 Готинчан Т.І. Основи інтернет технологій: методичні вказівки та завдання до лабораторних робіт / Укл: Т.І. Готинчан, – Чернівці : Чернівецький нац. ун-т, 2023. – 48 с.

Видання містить методичні вказівки та завдання до лабораторних робіт, які охоплюють основні принципи вступу до веброзробки за допомогою HTML і CSS.

Для студентів спеціальностей «Комп'ютерні науки» та «Системний аналіз»

УДК 004.774(076.5)

Зміст

Вступ

Реалії сьогодення людства вимагають створення різноманітних вебпродуктів, які допомагають людині у різних сферах її діяльності. Веброзробка постійно змушує розробників постійно вивчати щось нове, щоб бути кращим і затребуваним. Її величезний плюс – це "низький" поріг входу у професію. Так, вивчати потрібно буде багато всього, і робити це потрібно постійно. Але для того, щоб «почати», потрібно вивчити ті три стовпи веб-розробки:

- HTML;
- CSS;
- JavaScript.

Перераховані вище технології – лише необхідна база. HTML – стандартизована мова розмітки сторінок в інтернеті, CSS – каскадна таблиця стилів, що відповідає за зовнішній вигляд, мова програмування JavaScript відповідає за реагування елементів на дії користувача.

Найкраще, якщо освоєння основ веброзробки відбувається в умовах реального проекту – тоді відбувається освоєння й супутніх інструментів: графічних редакторів та редакторів коду, інструментів розробника у браузері тощо. Тому студентам рекомендується верстати реальні проєкти за вебмакетами.

Уміння аналізувати вебмакети у графічному редакторі, виділяти елементи та зв’язки між ними, а вибудовувати адаптивну верстку під різні пристрої допомагають студентам спеціальностей «Комп’ютерні науки» та «Системний аналіз» галузі «Інформаційні технології» у майбутньому професійно створювати вебпродукти різного рівня складності.

Методичні вказівки містять завдання до лабораторних робіт з курсу «Основи інтернет технологій», а також теоретичні відомості. Передбачено, що студенти виконують індивідуальні проєкти. При чому кожна лабораторна робота є основою для наступної.

Мета і завдання навчальної дисципліни

Мета навчальної дисципліни полягає в опануванні майбутніми фахівцями теоретичних знань і практичних навичок, необхідних для вирішення питань, пов'язаних із макетуванням та верстkою вебсторінок у глобальній мережі інтернет з використанням сучасних технологій, а саме навчити студентів прочитувати макети вебсторінок, використовувати семантичні теги з необхідними атрибутами для їхньої розмітки, підбирати правила і технології стилізації, ефективно їх оформлювати.

Дисципліна формує такі компетентності [1, 2]:

- за ОП «Комп'ютерні науки»:

Загальні компетентності:
ЗК1. Здатність до абстрактного мислення, аналізу та синтезу.
ЗК2. Здатність застосовувати знання у практичних ситуаціях.
ЗК7. Здатність до пошуку, оброблення та аналізу інформації з різних джерел.
ЗК8. Здатність генерувати нові ідеї (креативність).

Фахові компетентності спеціальності:
ФК3. Здатність до логічного мислення, побудови логічних висновків, використання формальних мов і моделей алгоритмічних обчислень, проектування, розроблення й аналізу алгоритмів, оцінювання їх ефективності та складності, розв'язності та нерозв'язності алгоритмічних проблем для адекватного моделювання предметних областей і створення програмних та інформаційних систем.
ФК10. Здатність застосовувати методології, технології та інструментальні засоби для управління процесами життєвого циклу інформаційних і програмних систем, продуктів і сервісів інформаційних технологій відповідно до вимог замовника.

Наведені результати навчання за відповідною дисципліною співвідносяться із ***такими програмними результатами навчання***:
ПРН1. Застосовувати знання основних форм і законів абстрактно-логічного мислення, основ методології наукового пізнання, форм і методів вилучення, аналізу, обробки та синтезу інформації в предметній області комп'ютерних наук.

- ОП «Системний аналіз»:

Загальні компетентності:
ЗК1. Здатність до абстрактного мислення, аналізу та синтезу.
ЗК2. Здатність застосовувати знання у практичних ситуаціях.
ЗК7. Здатність до пошуку, оброблення та аналізу інформації з різних джерел.
ЗК11. Здатність генерувати нові ідеї (креативність).
ЗК14. Здатність оцінювати та забезпечувати якість виконуваних робіт

Фахові компетентності спеціальності:
ФК8. Здатність організовувати роботу з аналізу та проектування складних систем, створення відповідних інформаційних технологій та програмного забезпечення.
ФК11. Здатність системно аналізувати свою професійну і соціальну діяльність, оцінювати накопичений досвід.

Наведені результати навчання за відповідною дисципліною співвідносяться із такими ***програмними результатами навчання*****:**
ПР8. Володіти сучасними методами розробки програм і програмних комплексів та прийняття оптимальних рішень щодо складу програмного забезпечення, алгоритмів процедур і операцій.
ПР13. Проектувати, реалізовувати, тестувати, впроваджувати, супроводжувати, експлуатувати програмні засоби роботи з даними і знаннями в комп’ютерних системах і мережах.

Постановка завдання

Верстка сайту

Розробка вебсайтів починається з правильної постановки завдання. Необхідно вирішити, які завдання стоять перед майбутнім ресурсом, які саме результати роботи очікуються у результаті. Важливо вже на початкових етапах розробки досить точно визначити, які дії має зробити користувач, який відвідав ресурс та ознайомився з його вмістом.

Чітко поставлена мета допоможе створити вебсайт, максимально ефективним. Слід продумати, що є головним, а що – другорядним.

Тільки після визначення головної мети вирішується питання структури сайту. Залежно від завдань підбирається вміст сайту: текст, зображення, за необхідності медіа-контент. А також складається головне меню, обговорюються елементи навігації тощо. Після цього створюється макети вебсторінок, на підставі яких й верстаються самі сторінки та вікна вебсайту.

Кожен розробник повинен прагнути, щоб сайт був зручним для користувача та вирішував поставлені завдання.

Завдання проєкту:

1. Виберіть тематику свого проєкту. Наприклад,
   - Мій рідний край;
   - ІТ-технології;
   - Онлайн-фірма;
   - Онлайн-школа за певними напрямками;
   - Подорожі та дозвілля;
   - Тема за вибором.
2. У вашому проєкті має бути не менше 3-4 сторінок. Обов’язкові:
   - головна (index.html);

- опитувальник;
- сторінка новин або статей за тематикою проєкту.

Додатковими сторінками можуть бути, наприклад,

- ваші досягнення (сертифікати, нагороди, тощо);
- елементи діяльності (пропонований перелік);
- тощо.

Зауваження. Реєстрація та авторизація може бути зреалізована як на головній сторінці, так і окремій.

3. Усі сторінки мають бути зв’язані між собою.
4. Верстка вебсторінок сайту має бути семантичною та чутливою до розмірів вікна перегляду на різних пристроях.
5. Каскадні таблиці стилів застосовуються до всіх сторінок сайту. Не створювати для кожної сторінки окремий файл CSS із повторювальними правилами.
6. Готовий проєкт розмістити на ресурсі netlify.com

Завдання та вказівки до лабораторних робіт

Лабораторна робота № 1.

Розмітка вебсторінок сайту

Короткі теоретичні відомості

HTML – це мова розмітки. Вона відповідає за вміст вебсторінки, а не за його візуалізацію. Стилізація здійснюється за допомогою CSS.

Поточним стандартом HTML є Living Standard, який ведеться організацією WHATWG [3]. Довідкова інформація по кожному html-елементу наведена в [1 – 3] з переліку інформаційних ресурсів.

Основними елементами HTML є теги та атрибути тегів.

Тег - це синтаксична одиниця мови HTML, яка виділяє або створює елемент. Є:

- ***Подвійні теги*** - показують початок і кінець елемента. Початок елемента позначається відкриваючим тегом <...>, а кінець – закриваючим </ ...>.
- ***Одинарні теги*** просто не мають пари. Наприклад, тег розриву рядків
 або горизонтальної лінії <hr>.

Атрибут використовується для визначення характеристик html-елемента і поміщається усередині тегу елемента, що позначає початок, тобто лише у <...>. Зазвичай атрибути складаються з двох частин – це ім'я і значення:

- Ім'я – це властивість, яку потрібно встановити.
- Значення – це значення, яке потрібно встановити для властивості. Значення атрибута завжди поміщається у лапки. Атрибути можуть бути зі значеннями і без них – логічними.

Сучасний підхід у верстці – це семантика. Семантична верстка – підхід до розмітки, який опирається не на вмісті сторінки, а на змістовому значенні кожного блоку та логічну її структуру.

Семантичні теги структури:

<article>

- Значення: незалежна, смислова одиниця, що відокремлюється, наприклад коментар, твіт, статтю, блог тощо.
- Особливості: бажаний заголовок усередині.

<section>

- Значення: смисловий розділ документа. Невіддільний, на відміну від <article>.
- Особливості: бажаний заголовок усередині.

<aside>

- Значення: побічний, непрямий для сторінки контент.
- Особливості: може мати власний заголовок. Може зустрічатись кілька разів на сторінці. Не призначений лише для бічної панелі.

<nav>

- Значення: навігаційний розділ із посиланнями на інші сторінки або частини сторінки.
- Особливості: використовується для основної навігації, а не всіх груп посилань. Основною є навігація чи ні – на розсуд верстальника. Наприклад, меню у підвалі сайту можна не обертати в <nav>. У підвалі зазвичай з'являється короткий список посилань (наприклад, посилання на головну сторінку, авторські права та умови) – це не є основною навігацією, семантично для такої інформації призначений <footer> сам собою.

<header>

- Значення: вступна частина смислового розділу або всього сайту, зазвичай містить підказки та навігацію. Найчастіше повторюється на всіх сторінках сайту.
- Особливості цих елементів може бути кілька на сторінці.

<main>

- Значення: основний зміст сторінки, що не повторюється на інших сторінках.
- Особливості: має бути один на сторінці, виходячи з визначення. Не можна включати у нього те, що повторюється на сторінці.

<footer>

- Значення: заключна частина смислового розділу або всього сайту зазвичай містить інформацію про авторів, список літератури, копірайт і так далі. Найчастіше повторюється на всіх сторінках сайту.
- Особливості: цих елементів може бути кілька на сторінці. Тег <footer> не обов'язково має бути наприкінці розділу. Може використовуватись не лише як підвал сайту.

<address>

- Значення: блок, в якому розміщується контактна інформація.
- Особливості: співставляється з найближчим батьківським тегом article, footer, або всім документом;

Теги для форматування j:

- Теги заголовків: <h1> ... <h6>;
- Для форматування тексту: <b>, , <i>, , <sub>, <sup>, <u>,<ins>, <del>;
- Теги для введення неформатованого тексту: <pre>, <code>, <samp>;

- Теги для цитат і визначень: <abbr>, <blockquote>, <q>, <cite>, <dfn>;
- Абзаци і засоби перенесення тексту: <p>,
;

Теги для оформлення списків однорідних елементів:

- <ul, ol, dl> – маркований, нумерований список і список визначень;
- – дочірні елементи ol, ul$
- <dt, dd> – дочірні елементи dl.

Теги для посилань та гіперпосилань: <link>, <a>

Приклад оформлення якоря

```
<p><a href="#team "> посилання </a></p>
…
<p id =”team”>якір</p>
```

Тег для зображення: <img>.

```
<img src="images/image.png" alt="Пояснення зображення">
```

Приклад зображення, що є посиланням:

```
<a href="#"> <img src="image.png" alt="картинка" > </a>
```

Теги медіаконтенту: <video>, <audio>, <source>.

Приклад виведення відеоконтенту

```
<video width="400" height="300" controls="controls"
poster="video/duel.jpg">
    <source src="video /duel.ogv" type='video/ogg;
codecs="theora, vorbis"'>
    <source src="video /duel.mp4" type='video/mp4;
codecs="avc1.42E01E, mp4a.40.2"'>
    <source src="video /duel.webm" type='video/webm;
codecs="vp8, vorbis"'>
    video не підтримується вашим браузером.
    <a href="video /duel.mp4">Скачайте відео</a>
</video>
```

Можливі атрибути тегів audio та video:

- autoplay="autoplay" – автостарт програвання;
- muted=“muted“ – вимикає звук;
- controls="contols" – відображати панель керування з кнопками;
- loop="loop" – циклічне програвання;
- preload="preload" – завантажувати файл у пам’ять одразу після завантаження сторінки;
- poster="URL-адреса" – адреса картинки для відображення замість відео, поки воно не завантажиться.

Загальні теги, що не мають семантики:

- – для стилізації блоків чи розмітки другорядної інформації;
- – для стилізації рядкового вмісту чи розмітки другорядної інформації.

Шаблон розмітки сторінки:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible"
content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Заголовок сторінки</title>
    <link href=“css/style.css” rel=“stylesheet”>
</head>
<body>
    <header>
        <h1>Головний заголовок сайту</h1>
        <p>Текст</p>
```

```
        <nav>
            <ul>
                <li><a href="index.html">Поточна сторі-
нка</a></li>
                <li><a href="other.html">Посилання на
іншу сторінку</a></li>
            </ul>
        </nav>
    </header>
    <!-- коментар -->
    <main>
        <article>
            <section>
                <h2>Заголовок другого рівня</h2>
                <p>Опис</p>
                <img src="images/image.png"
alt="Пояснення зображення">
                <p>Текст абзацу</p>
            </section>
            ...
            <section>
                <h2>Заголовок другого рівня іншої сек-
ції</h2>
                <p>Текст абзацу</p>
            </section>
        </article>
    </main>
    <footer>
        <p>Інформація про розробника сайту</p>
    </footer>
     <script src="scripts/script.js" defer></script>
</body>
</html>
```

Теги форм:

- <form> є контейнером для форми, розміщеної на вебсторінці;
- <fieldset>, <legend> – для групування елементів форми;
- <select>, <optgroup>, <option>, <datalist> – для формування виданих списків;
- <input> – поле введення. Атрибут type визначає тип кнопки:
  - text – введення довільних символів;
  - password – для введення паролів;
  - email – перевірка чи є адресою електронної пошти;
  - checkbox – перетворює поле введення під прапорець;
  - radio –використовується для створення
  - групи радіо-кнопок (перемикачів), що описують набір
  - взаємозв'язаних параметрів;
  - та інші.
- <textarea> ... </ textarea> використовується, коли потрібно створити великі текстові поля.
- <button> – дозволяє створити клікабельні кнопки. Всередину елемента <button> можна помістити контент - текст або зображення. Атрибут type визначає тип кнопки:
  - button - клікабельна кнопка, можна призначити обробник за допомогою JavaScript
  - reset - кнопка скидання, повертає первісне значення форми (спрацьовує лише у блоці <form>)
  - submit - кнопка для відправки даних форми.

Приклад форми авторизації

```
<form action="#" method="post">
     <fieldset>
        <p> <label> Логін <input id type="email"
placeholder="email@gmail.com" name="login"
tabindex="1"> </label></p>
```

```
        <p> <label> Пароль <input type="password"
placeholder="Не менше 8 символів" name="pass"
tabindex="2"></label> </p>
      </fieldset>
      <fieldset>
         <button type="submit"
tabindex="3">Увійти</button>
         <button type="reset">Очистити</button>
      </fieldset>
 </form>
```

Завдання та хід виконання

1. Виберіть тематику свого проєкту.
2. Підберіть або напишіть самостійно текстовий вміст до кожної сторінки.
3. Підберіть або самостійно створіть зображення, аудіо- або відео контент, які використовуватимете у вмісті.

Зауваження. Бажано у тексті передбачити заголовки, списки, таблиці, цитати, розшифрування абревіатур, виділення тексту, тощо. Обов'язкове вставлення зображень, а за змістом й аудіо- чи відео фрагментів. Якщо є програмний код, то від має бути відповідно відображений. Зазначте у супровідній документації посилання на ресурси, звідки взяті зображення, аудіо та відео. Якщо текст також запозичений, то у розмітці передбачити посилання на джерело.

4. Зображення оптимізувати, наприклад, за допомогою ресурсів:
   - https://squoosh.app/ – для растрових зображень;
   - https://jakearchibald.github.io/svgomg/ – векторних зображень.
5. Створіть папку проєкту. У середині папки створити внутрішні папки відповідно окремо із макетами сторінок, зображеннями, відео та аудіо, файлами стилів, супровідною документацією.

Зауваження. У процесі роботи над проєктом наповнюйте ці папки необхідним вмістом.

6. Оберіть або продумайте макет кожної сторінки і створіть його, наприклад, засобами figma [7] для пристроїв різних розмірів.

Зауваження. Ви можете використати будь-який уподобаний вами макет головної сторінки, що є у вільному доступі. Обов'язково вказати посилання на ресурс макету у супровідній документації про джерела ресурсів.Наприклад, https://www.free-css.com/free-css-templates, https://nicepage.com/website-templates, тощо.

7. Опрацюйте способи під'єднання іконок, наприклад, як у https://www.w3schools.com/icons/icons_reference.asp.

8. Опрацюйте мнемоніку для вставлення у HTML-розмітку, наприклад за https://unicode-table.com/en/html-entities/ .

9. Відповідно до макету сторінок та сценарію інтерактивності підберіть відповідні теги HTML5.

10.Відкрийте онлайн-довідник по HTML (наприклад, https://css.in.ua/) або підручники (наприклад, https://www.w3schools.com/html/default.asp, https://developer.mozilla.org/en-US/docs/Web/HTML) та прочитайте інформацію про теги, які використовуватимете в роботі. Використовуйте й необхідні атрибути.

Зауваження. Уникайте рядкової стилізації та встановлення розмірів без потреб. Зосередьтеся на розмітці.

11.Запустіть середовище роботи зі HTML сторінками (наприклад, **Visual Studio Code**, Sublime Text, NodePad++ або інші).

Зауваження. Для створення деяких громіздких елементів, зокрема таблиць, багаторівневих списків можна використовувати онлайн HTML-редактори (наприклад, https://html5-editor.net/), а для перевірки правильності коду – валідатори (наприклад, https://validator.w3.org). Проте **не стилізувати** сам текст, лише розмітка.

12.На основі вашого тексту розмітьте HTML-сторінки, використовуючи семантичні теги та необхідні атрибути.

13.Продумайте переміщення між елементами сторінок та самими сторінками. Зв'яжіть необхідні елементи та сторінки.

14. Захистіть лабораторну роботу та розмістіть на сторінці курсу.

Запитання для повторення

1. Що таке тег? Яким він може бути? Як зображається?
2. Що таке атрибут? Його призначення та запис.
3. Що таке семантична верстка? Чому вона важлива?
4. Які семантичні теги визначають структуру вебсторінки?
5. Які теги використовують для розмітки тексту?
6. Призначення тегів заголовків? Які вимоги до їхнього використання?
7. За допомогою яких тегів можна вказати посилання та гіперпосилання?
8. Як розмітити медіа-контент?
9. Які теги не мають семантики? Для чого їх використовують?
10. Які теги використовують для розмітки форми?

Лабораторна робота № 2

Основи CSS. Форматування тексту. Фони та рамки

Короткі теоретичні відомості

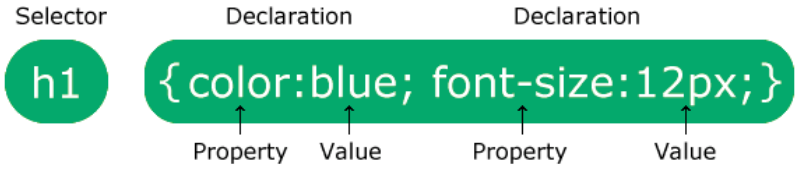
CSS (Cascading Style Sheets) – формальна мова для опису зовнішнього вигляду документа, написаного мовою розмітки.

Основне призначення CSS:

- розділити на синтаксичному рівні структуру документа від його візуального оформлення;
- підвищити швидкість завантаження сторінки, оскільки таблиці стилів, які написані в окремому файлі, кешуються браузером;
- можливість адаптивного оформлення під різні типи пристроїв.

Правило CSS складається з двох блоків:

- один блок – це селектори, які зіставляються елементам на сторінці, які будуть потім стилізовані відповідно до правил;
- другий блок – це так звані оголошення або описи, які складаються з пари властивість - значення.



Коментарі можуть розташовуватися тільки всередині спеціальної послідовності символів / * ... * /.

Є три основних способи підключення CSS-стилів.

Перший спосіб полягає в тому, що всі необхідні CSS-правила записуються всередині значення атрибута style у елемента. Використовується найчастіше з боку скриптів.

```
<p style = "text-align: justify">
    Lorem ipsum dolor sit amet consectetur adipisicing.
</p>
```

Другий спосіб – це розмістити всі CSS-стилі всередині тега <style>, який, в свою чергу, розміщується всередині тегу <head>. Такий підхід використовується у випадку Critical CSS.

```
<head>
        ...
    <style>
      h2 {
          color: red;
          font-family: Arial, Helvetica, sans-serif;
          }
    </style>
</head>
```

Третій спосіб – це спосіб підключення через зовнішній файл з розширенням .css. Такий спосіб найпрактичніший, оскільки дозволяє застосовувати файл зі стилями для різних HTML-документів. Файли зі стилями кешуються браузерами.

```
<head>
        ...
   <link rel="stylesheet" href="style.css">
</head>
```

Селектор -– це формальний опис того елемента або групи елементів, до яких застосовується указане правило стилю.

Селектори, як правило ділять на прості та комбіновані, які складаються з простих за допомогою знаку комбінатора.

Основні види селекторів:

1) селектор елемента p{…}
2) селектор універсального елемента *{…}
3) селектор ідентифікатора #id{…}

```
<div id="warning">…..</div>
        #warning {
             color: red;
        }
```

4) селектор класу .main{…}

```
<div class="warning">…..</div>
       .warning {
          color: red;
        }
```

5) селектор нащадка застосовують стилі до елементів, розташованих усередині елемента-контейнера.

```
ul li {
            list-style: none;
            display: inline-block;
       }
```

6) селектор псевдокласу. Псевдоклас – це особливий клас, не прикріплений до певного HTML-тегу, а визначає особливий стан елемента.

```
button {
            background-color: white;
            border: 1px solid brown;
        }
button:hover {
            background-color: pink;
            border: none;
        }
```

7) селектор псевдоелемента**.** Псевдоелемент в CSS дозволяє стилізувати певну частину обраного елемента.

```
p::first-line {
            color: darkblue;
           }
```

8) Групування селекторів

```
h1,h2,p {
            font-family: Verdana;
            color: dimgray;
        }
```

Існують й комбінування селекторів, з якими можна ознайомитись у [1-3] інформаційних джерел.

Пріоритетність стилів

Інлайнові стилі мають вищий пріоритет, ніж внутрішні та зовнішні.

Специфічність CSS-селекторів, спадкування і каскад для таблиць стилів

Правило специфічності: *кожному селектору в CSS зіставляється тричислове значення, яке називається* ***специфічністю.***

Для **обчислення** цього значення необхідно зробити такі дії:

1) Порахувати число селекторів ідентифікатора (#id).
2) Порахувати число селекторів по класу / псевдоклас (.class).
3) Порахувати число селекторів, які відповідають елементу / псевдоелементу (наприклад, div).
4) Універсальний селектор * має **нульову** специфічність.

За допомогою ключового слова **!important,** яке додається відразу після значення властивості, можна визначити правило, яке матиме пріоритет над усіма іншими правилами.

Спадкування – це механізм, за допомогою якого стилі застосовуються не тільки до вказаних елементів, але так само до їхніх нащадків.

Каскадування – це механізм, який керує кінцевим результатом у ситуації, коли до одного елементу застосовуються різні CSS-правила.

Існує три критерії, які визначають порядок застосування властивостей:

- найперше правило: **!important** має найвищий пріоритет;
- друге – це специфічність правила, тобто застосовується правило з більш високою специфічністю серед кількох конфліктуючих;
- і найостанніше: якщо попередні правила не дозволили вирішити конфлікт (наприклад конфліктуючі правила не мали оголошень **!important** і мають однакову специфічність), браузер дивитиметься на порядок оголошення цих правил усередині однієї або декількох таблиць стилів.

Одиниці вимірювання довжини

Для завдання розмірів елементів веб-сторінки використовуються одиниці вимірювання довжини, які в свою чергу можна поділити на

- **абсолютні** одиниці вимірювання довжини;
- **відносні** одиниці вимірювання довжини.

Базовою абсолютною одиницею вимірювання довжини в CSS є **піксель** (1px). Числовими значеннями пікселя можуть бути як цілі, так і дробові значення.

Важливо пам'ятати, *що кількість пікселів на сторінці визначено налаштуваннями екрану, і відповідатиме реальним пікселям екрану тільки в тому випадку, якщо вони збігаються*.

До **відносних** одиницях виміру довжини можна віднести:

- em - залежить від розміру шрифту елемента, до якого застосовується. Якщо у елемента заданий шрифт розміром 10px, то 1em буде дорівнює 10px;

- rem - залежить від розміру шрифту кореневого елементу сторінки (тег <html>);
- % - відсоток буде від значення властивості батька з тією ж назвою.
- одиниці області перегляду (viewport):

Усього одиниць вимірювання viewport є **чотири** і вони мають такі визначення.

- **Одиниця vw** -– це одиниця, яка дорівнює 1/100 (або по іншому 1%) від ширини області перегляду.
- **Одиниця vh** -– це одиниця, яка дорівнює 1/100 (або по іншому 1%) від висоти області перегляду.
- **Одиниця vmin** – 1/100 найменшого значення області перегляду.
- **Одиниця vmax** – 1/100 максимального значення області перегляду.

Властивості форматування тексту

Під'єднувати шрифти можна, наприклад, з ресурсу GoogleFonts (https://fonts.google.com) у head HTML-сторінки.

```
<link
href="https://fonts.googleapis.com/css?family=Lato|Robo
to:300,400,500&display=swap" rel="stylesheet">
```

Властивість **font-family** використовується для вибору шрифту.

Властивість **font-weight** задає насиченість шрифту.

Властивість **font-size** вказує бажану висоту гліфів з шрифту.

Властивість **font-style** дозволяє вибрати стиль накреслення для шрифту.

Властивість **text-decoration** додає оформлення тексту у вигляді його підкреслення, перекреслення, лінії над текстом.

Властивість **text-transform** стилізує текст.

Властивість **text-shadow** дозволяє додати одну або більше тіней для тексту.

Властивість **text-align** визначає горизонтальне вирівнювання тексту в межах блочного елемента (контейнера).

Властивість **letter-spacing** дозволяє збільшувати або зменшувати відстань між буквами в тексті.

Властивість **word-spacing** дозволяє регулювати відстанейь між словами.

Властивість **line-height** дозволяє регулювати відстань між рядками.

Властивість **color** визначає колір тексту елемента.

```
h1 {
   font-family: 'Courier New', Courier, monospace;
   font-weight: 900;
   font-size: 2rem;
   font-style: italic;
   text-transform:uppercase;
   text-align: center;
   text-shadow: 5px 5px #808080;
   color: dimgray;
   }
p{
   font-size: 1.5em;
   line-height: 1.2;
   letter-spacing: 1.1;
   word-spacing: 0.9;
   }
a{
   text-decoration: none;
   }
```

Властивість **text-overflow** вказує, що має статися, коли текст переповнює містить елемент.

Властивість **vertical-align** керує вертикальним вирівнюванням рядкових елементів

```
span {vertical-align: super;}
```

Фони і рамки

Властивість **background** дозволяє описати в одному оголошенні такі властивості фону: **background-color, background-image, background-position, background-size, background-repeat,** які можна задавати й окремо.

Властивість **background-attachment** дозволяє визначити чи позиція фонового зображення буде фіксованою у вікні перегляду, чи прокручується разом із блоком, що містить його.

```
div {
   background: silver url(https://) center/cover no-
repeat;
   background-attachment: fixed;
   }
```

CSS визначає два типи градієнтів:

- **Лінійний градієнт** (Linear Gradient) - плавний перехід від кольору до кольору по прямій лінії.
- **Радіальний градієнт** (Radial Gradient) - плавний перехід від кольору до кольору з однієї точки в усі напрямки.

```
div {
  background-image: linear-gradient(black, white);
  width: 200px;
  height: 200px;
}
```

Властивість **border-style** задає рамку, яка вимальовуватиметься поверх фону елемента.

Властивість **border-color** задає колір рамок усіх сторін одночасно.

Властивість **border-width** задає ширину рамки.

Властивість **border** дозволяє об'єднати в собі такі властивості: border-width, border-style, border-color.

Властивість **border-radius** задає заокругленість кутів рамки.

Властивість **outline** дозволяє в одному оголошенні встановити ширину, стиль і колір зовнішнього межі елемента.

```
div {
    border: 5px solid #fafafa;
    border-radius: 5px;
  }
```

Завдання та хід виконання

Частина 1. Тренування

1. Скачайте файли заготовки з Moodle і розмістіть їх в одній папці. Відкрийте папку у редакторі коду.
2. Подивіться зразок, який ви маєте отримати, у файлі sample.jpg.
3. Запустіть у браузері файл index.html без стильового оформлення. Ознайомтесь зі структурою розмітки тексту у редакторі коду.
4. Перегляньте у редакторі коду файл style.css.
5. Під'єднайте файл стилів до файлу розмітки.
6. За потреби змініть онлайн url зображення в абзацах на адресу зображення у папці.
7. Прочитайте інформацію про стилі, які використовуються у роботі.
8. Замініть слово selectors на відповідні прості чи комбіновані селектори, опис яких зазначений у коментарях. Ви маєте сторінку оформити за зразком, який наведений у файлі sample.jpg.

Зауваження. Для створення таблиць стилів можна застосовувати й інші підходи. Ваша мета навчитись застосовувати різні прийоми написання таблиць стилів.

9. Наведіть курсор на елементи правої панелі та зображення. Зверніть увагу як змінилось їхнє стильове оформлення. Проаналізуйте де саме це описано у style.css.
10. Нижче заголовка додайте горизонтальне меню-навігацію (перехід на зовнішні сторінки).

У файлі style.css проаналізуйте стилі у блоці навігації і допищіть відповідно розмітку у файлі index.html, щоб отримати панель навігації вигляду

11.Відокремте змістові блоки стилів за змістом коментарями як це зроблено для навігаційної панелі.

Частина 2. Форматування тексту у власному проєкті

1. Оформіть свій сайт, застосувавши розмітку з першої лабораторної роботи.
2. Створіть папку styles у середині папки проєкту.
3. Створіть файл з розширенням .css у середині цієї папки.
4. Під’єднайте файл зовнішніх таблиць стилей до кожної зі сторінок.
5. Під’єднайте необхідні шрифти та файл іконок.
6. Працюєте лише над стильовим оформленням тексту, іконок та зображень, підібравши відповідні властивості відповідно до макету. Використовуйте відносні одиниці вимірювання em чи rem для шрифтів і img {width: 100%; max-width: 100%;} для гнучкої верстки.
7. Опрацюйте тему псевдокласи, псевдоелементи та функції CSS.
8. Напишіть правила стилізації масштабування та фільтрації зображень та іконок, при наведенні курсора на них.
9. Оптимізуйте ваші правила стилізації, враховуючи наслідування та каскадування.
10.Зверніть увагу на форматування посилань та заголовків. Особливо щодо наслідування властивостей.

11.Використайте :root та var(), щоб оптимізувати підтримку можливих змін зі стилізацією тексту, кольорів.

12.Стилізуйте написи до елементів форми.

13.Оформіть вбудовану валідацію елементів форми за допомогою CSS.

14.Оформіть роботу. Продемонструйте роботу викладачу

15.Завантажте роботу чи посилання на неї на сторінку курсу

Запитання для повторення

1. Призначення CSS?
2. Синтаксис правила CSS.
3. Що таке селектор і що може бути селектором?
4. Які способи приєднання стилів є? Їхній пріоритет.
5. Переваги та недоліки кожного способи подання стилізації.
6. Що таке специфічність селектора? Як обчислити число специфічності?
7. Що таке успадкування?
8. Як задати успадкування значення властивості? Кольору?
9. Що таке каскадування? Як воно діє?
10. Які властивості застосовуються для форматування тексту?
11. Чи усі вони успадковуються і чи для всіх елементів?
12. Як можна задати колір? Тексту? Фону?
13. Яка відмінність між лінійним та радіальним градієнтами?
14. Як можна задати тінь для тексту та рамки?
15. Як можна стилізувати замки блоків?

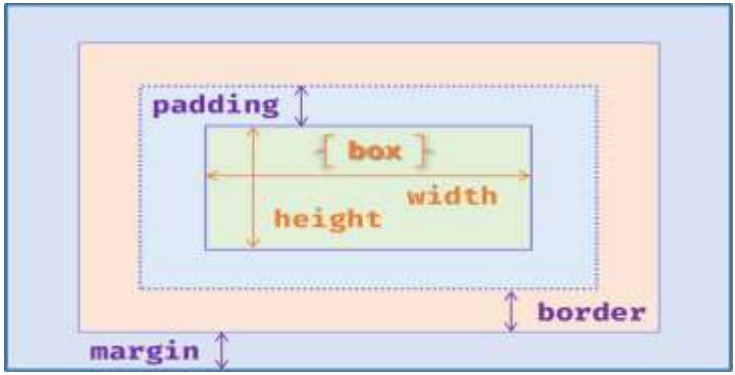
Лабораторна робота № 3

Блокова модель. Позиціювання. Z-index. Flex. Grid

Короткі теоретичні відомості

Блокова модель

У CSS пропонується, що кожен елемент створює один або кілька прямокутних блоків, які називаються контейнерами елементів.



Властивість **display** дає змогу змінити розташування елементів з вертикального на горизонтальне положення або навпаки.

Значення властивості display, таке як **inline** робить елементи рядковими.

Значення властивості display, такі як **block, list-item, table** роблять елементи блоковими.

Значення властивості display таке як **inline-block** роблять елементи рядково-блоковими.

Значення властивості none приховує елемент, наче б його не було.

Властивість **visibility** зі значеннями visible, hidden при приховані елемента залишає порожнє місце.

Властивість **width** визначає ширину вмісту в блоці.

Властивість **height** визначає висоту вміст блоку

Поля, відступи і межі (рамки) елементів визначаються за годинниковою стрілкою. Початок відліку згори.

Властивість **box-sizing** дозволяє впливати на алгоритм визначення загального розміру блоків. Властивість набуває два основних значення:

- content-box – (значення за замовчуванням) задають ширину і висоту контенту і не включають у себе значення відступів, полів і кордонів.
- border-box – коли властивості width і height включають у себе значення полів і меж, але не включають відступів (margin).

Властивість margin може набувати як додатних, так і від'ємних значень. Також їй властиве поняття колапсу відстаней по вертикалі.

Щоб розташувати блоковий контейнер з із шириною, меншою за розмір вікна перегляду по центру по горизонталі, достатньо задати значення auto для властивості margin.

```
div { margin: 0 auto; }
```

Позиціонування елементів

Позиціонування елементів дозволяє за допомогою CSS-властивостей витягувати елемент зі звичайного потоку документа і розміщувати його незалежно в необхідній частини документа. Додатковою властивістю позиціонування є властивість **z-index**, яка

показує на якому рівні розташований елемент щодо інших у так званій осі-Z

Для того, що б змінити стандартну поведінку існує CSS-властивість **position**. Ця властивість на сьогодні може набувати всього п'ять константних значень:

- static - (значення за замовчуванням)
- relative
- absolute
- fixed
- sticky

А також для за потреби задати властивості:

- left - яке позначає координати елемента зліва
- right - координати справа
- top - координати зверху
- bottom - координати знизу

Значення **static** встановлено для все елементів, для яких не вказано інших варіантів позиціонування, тобто це поведінка по-замовчуванню.

Значення **fixed** робить елементи фіксованими відносно лівого боку батьківського елемента чи body, залежно задані властивості координат чи ні.

Значення **relative** робить елементи відносними свого положення. Також використовується без зазначення координат для подальшого використання position: absolute.

Значення **absolute** робить елементи абсолютно закріпленими до батьківського елемента.

Значення **sticky** робить елементи «липкими» до певної позиції.

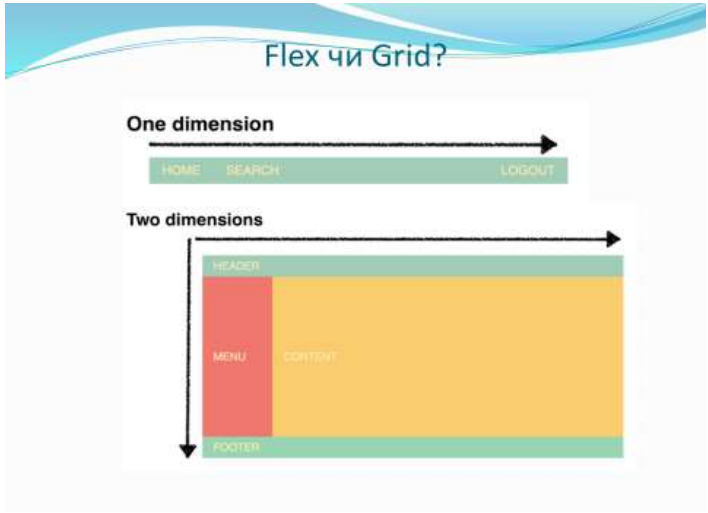
Наприклад, реалізація кнопки вгору

```
<div class="to-up">Up</div>
.to-up {
    width:20px;
    height:20px;
    border-radius:50%;
    position: fixed;
```

```
    bottom: 10px;
    right: 10px;
}
```

Flex, Grid

Технології Flex, Grid дозволяють розміщувати елементи і перебудову їхнього положення відносно розміру вікна перегляду.



Flexbox керує розташуванням елементів відносно осі, а Grid – відносно двох осей.

Відстані між елементами задаються властивістю gap.

Оголошення flexbox відбувається властивістю display

```
display: flex;  або display: inline-flex;
```

Основні поняття Flexbox:

flex-контейнер, flex-елемент, головна і поперечна осі.

flex-контейнер встановлює поведінку гнучкого форматування для його вмісту, тобто для дочірніх елементів, які є flex-елемент

Властивості, що задаються для **контейнера:**

flex-direction задає напрямок розташування елементів вздовж головної осі.

flex-wrap зазначає чи будуть елементи перепливати на наступний рядок у разі не поміщення їх в одному рядку.

justify-content дає можливість гнучко розподіляти вільний простір усередині flex-контейнера між flex-елементами **за *головною віссю***

align-items дає можливість гнучко розподіляти вільний простір усередині flex-контейнера між flex-елементами **за *поперечною віссю***

align-content дозволяє гнучко розподіляти простір всередині flex-контейнера контейнера **за *поперечною віссю*** для окремих рядків цього flex-контейнера, коли flex-елементи переносяться на додаткові рядки, якщо не поміщаються в один ряд.

Властивості, що задаються для **flex-елементів**:

align-self використовується, щоб вирівнювати ***по поперечній*** осі окремо взяті flex-елементи.

order впливає на порядок відображення flex-елементів усередині контейнера.

flex-grow задає коефіцієнт збільшення ширини flex-елементаю

flex-shrink задає коефіцієнт зменшення ширини flex-елемента.

flex-basis задає базовий розмір flex-елемента по головній осі.

Наприклад, розташування головного заголовку по середині банера

```
<div>
    <h1> Головний заголовок</h1>
</div>

div{
   height: 150px;
   display: flex;
   justify-content: center;
   align-items: center;
}
```

Grid дає змогу виставити сітку для розмітки, а відносно сітки розташовувати елементи всередині неї та вирівнювати по головній та поперечній осі.

Оголошення flexbox відбувається властивістю display
`display: grid;` або `display: inline-grid;` або `display: subgrid;`

Одиницею вимірювання у grid крім традиційних, є fr (fraction).

Властивості, що задаються для **структури**:

grid-template-columns – кількість стовпчиків.

grid-template-rows – кількість рядків.

column-gap, row-gap, gap – відстань між елементами

grid-area – оголошення областей.

grid-template-areas описує шаблон сітки за допомогою вказання назв зон, що були оголошені за допомогою grid-area.

grid-auto-columns, **grid-auto-rows** визначають розмір треків, що створюються автоматично.

grid-auto-flow автоматичне розміщення браузером елементів.

Властивості, що задаються для **grid-контейнера:**

justify-content вирівнює сітку відносно горизонтальної осі.

align-content вирівнює сітку відносно вертикальної осі.

justify-items вирівнює **контент всередині комірки** відносно **горизонтальної** осі.

align-items вирівнює **контент всередині комірки** відносно **вертикальної** осі.

Властивості, що задаються для **grid-елементів**:

grid-**column**: grid-column-start / grid-column-end;
визначає з якого стовпця по який розміщуватиметься елемент. Аналогічно для рядків **grid-row**: grid-row-start / grid-row-end.

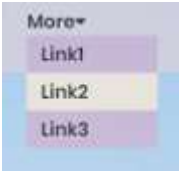
align-items

justify-self вирівнює поточний елемент відносно горизонтальної осі.

align-self вирівнює поточний елемент відносно вертикальної осі.

Завдання та хід виконання

1. До меню навігації допишіть підменю, використовуючи до нього позиціювання absolute, а до відповідного меню relative. При цьому використайте або іконку, або спецсимвол HTML для позначення розкриття меню (наприклад, знак стрілки або трикутника).



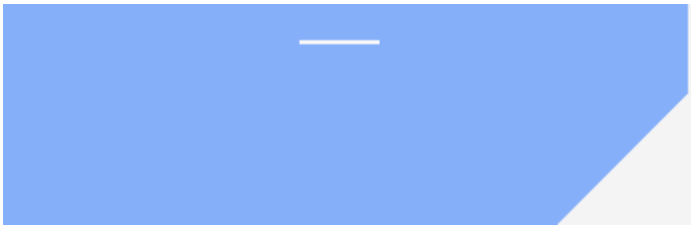
2. Оформіть кнопку навігації вгору, яка **зафіксована** у лівому нижньому кутку вікна сторінки при прокрутці сторінки. При натисненні на неї потрапляємо на блок навігації. Кнопка має відображатись поверх інших елементів при прокрутці вгору.

Зауваження. При оформленні кнопки можна використовувати іконки, зображення, спецсимволи, а також зображення, створене лише CSS (властивість transform)

3. Оформіть зображення на допоміжній частині так, щоб при завантаженій сторінки зображення не було видно або тьмяним, а натомість по середині був текст про нього. А вже при наведенні на нього курсора текст ховався, а зображення з'являлось. Наприклад, як

Зауваження. 1. Тут знову доцільно використати такий самий підхід як у пункті 1. Вирівнювання тексту посередині можна здійснити за допомогою `transform: translate(-50%, -50%).`

2. Для ефекту зрізаного кута можна використати властивості



```
background:linear-gradient(-45deg, transparent 32px, #1769ff  0);
```

Оформіть елемент пошуку. Наприклад, як

4. Використовуючи Flexbox і Grid, розмістіть відповідно до макету елементи на сторінках.

5. Перевірте за допомогою інструментів розробника відображення сторінки під різні пристрої.

6. Оформіть роботу. Продемонструйте роботу викладачу.

7. Захистіть лабораторну роботу та розмістіть на сторінці курсу.

Запитання для повторення

1. Призначення властивості display?
2. Які значення набуває властивість box-sizing?
3. Властивість position? Застосування.
4. Коли властивість z-index можна застосовувати?
5. Як можна вирівняти блок посередині сторінки по горизонталі?
6. Призначення і застосування flexbox?
7. Призначення і застосування grid?

Лабораторна робота № 4

Чутливий дизайн. Елементи анімації. Використання бібліотек

Короткі теоретичні відомості

Гнучка верстка (Elastic layout) означає можливість змінювати свої розміри в залежності від розміру вікна браузера. Для цього використовують відносні одиниці вимірювання довжини. У відносних одиницях, як правило, задають відступи, поля, розмір шрифту.

Адаптивна верстка (Adaptive Layout) означає підлаштування елементів сайту під розмір екрану, при цьому можливі зміни розміру шрифту, розташування об'єктів, колір і тощо. Зазвичай використовують медіа-вирази (@media). Можливе застосування JavaScript, для створення складних сценаріїв адаптивного поведінки

Поєднання обох верстоk використовується у чутливій верстці (**Responsive web design**)

Responsive web design = adaptive web design + прогресивні покращення.

Адаптивний дизайн орієнтується на контрольних (переломних) точках. Тобто макети завантажуються при певних розмірах вікна браузера пристрою. Цими точками, наприклад, можуть бути:

320, 480, 760, 960, 1200, 1600.

Зауваження. 1. У бібліотеці Bootsrap використовується інша система точок розбиття. 2. Використання flexbox та дозволяє автоматично перелаштовувати елементи на сторінці.

Медіа-вирази (@-media правила) змінюють стилі на підставі характеристик пристрою, пов'язаних з відображенням контенту, включаючи тип, ширину, висоту, орієнтацію і дозвіл екрана.

У медіа-виразах потрібно указувати точки перелому та за потреби типи пристроїв.

Типи пристроїв:

- all – для усіх типів пристроїв (за замовчуванням)
- print – для принтерів
- screen – для екранів комп'ютерів
- tv – для телеекранів.

Кілька умов одного медіа-виразу можна скомбінувати з допомогою логічних операторів:

- not -– для заперечення умов медіа запиту
- and – об'єднання декількох умов
- only – приховує медіа-вирази від браузерів, які їх не підтримують
- , (кома) – використовується як логічне або, вказує на те, що хоча б одна з умов має виконається.

Наприклад,

- правила стилізації для пристроїв з розміром вікна перегляду більщих, ніж 750px.

```
@media (min-width: 750px) {
    html{ font-size: 14px;}

    img{width: 80%;}
 }
```

- правила стилізації для пристроїв з розміром вікна перегляду від 320px до 640px.

```
 @media (min-width: 320px) and (max-width: 640px) {
    .card {
        min-width: 80%;
        margin: 2vw auto;
        font-size: 0.8em
    }
}
```

- правила стилізації для моніторів комп'ютерів довільного розміру і пристроїв з розміром вікна перегляду більших, ніж 768px.

```
@media only screen and (min-width: 768px){
    .block {
        background-color:white;
    }
}
```

Зауваження. Зазвичай медіа-вирази пишуть зверху вниз або навпаки. Так, якщо основна верстка для декстопних пристроїв, то зверху.

Переходи CSS дозволяють плавно змінювати значення властивостей протягом певного часу.

Щоб створити ефект переходу, необхідно вказати дві речі:

- властивість CSS transition;
- тривалість ефекту.

Властивості, які задаються при переході:

Властивість **transition-delay** визначає затримку (у секундах) для ефекту переходу.

Властивість **transition-timing-function** визначає криву швидкості ефекту переходу.

Властивість **transition-property** визначає ім'я властивості CSS, для якого призначений ефект переходу (ефект переходу почнеться, коли змінюється зазначена властивість CSS).

Задавати t**ransition** потрібно саме на елементі. А на дії стан кінцевої зміни

```
div {
        width: 100px;
        height: 100px;
        background: red;
        transition: width 2s, height 2s, transform 4s;
         }
```

```
  div:hover {
      width: 300px;
      height: 300px;
      transform: rotate(180deg);
   }
```

CSS дозволяє анімувати елементи HTML без використання JavaScript. Для цього слід використати правило **@keyframes** та властивість **animation**.

Властивості, які задаються при анімації:

Властивість **animation-delay** визначає затримку для початку анімації.

Властивість **animation-direction** визначає, чи має відтворюватися анімація вперед, назад або чергуючи цикли.

Властивість **animation-iteration-count** вказує, скільки разів анімація повинна запускатися.

Властивість **animation-timing-function** визначає криву швидкості анімації.

Властивість **animation-fill-mode** може замінити цю поведінку.

Наприклад, поступова зміна кольорів елемента. Кадри змінюються у певні проміжки часу, які задаються % від тривалості ефекту.

```
@keyframes example {
  0%   {background-color:red; left:0px; top:0px;}
  25%  {background-
color:yellow; left:200px; top:0px;}
  50%  {background-
color:blue; left:200px; top:200px;}
  75%  {background-
color:green; left:0px; top:200px;}
  100% {background-color:red; left:0px; top:0px;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
```

```
  position: relative;
  background-color: transparent;
  animation-name: example;
  animation-duration: 10s;
  animation-delay: 4s;
  animation-iteration-count: 3;
  animation-direction: alternate;
}
```

На сьогодні створено багато бібліотек, які пропонують готові шаблони HTML-конструкцій із власними стилями, що значно спрощує верстку. Проте, якщо використовувати готові шаблони без додавання власних стилів, то багато сайтів виглядали б однаково. Фреймворк Bootstrap можна використовувати й як бібліотеку шаблонів. Для цього слід прочитати документацію у [8] та під'єднати необхідні файли, які зазначені у вступі:

1) <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65" crossorigin="anonymous">
2) <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KFdBIe4zVF0s0G1M5b4hcpxyD9F7jL+jjXkk+Q2h455rYXK/7HAuoJl+0I4" crossorigin="anonymous"></script>
3) <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js" integrity="sha384-oBqDVmMz9ATKxIep9tiCxS/Z9fNfEXiDAYTujMAeBAsjFuCZSmKbSSUnQlmh/jp3" crossorigin="anonymous"></script>
4) <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.min.js" integrity="sha384-cuYeSxntonz0PPNlHhBs68uyIAVpIIOZZ5JqeqvYYIcEL727kskC66kF92t6Xl2V" crossorigin="anonymous"></script>

Зауваження. 3) і 4) використовуються замість 2), якщо не планується використовувати спадні списки, спливаючі вікна чи спливаючі підказки, тоді можна заощадити кілька кілобайтів, не включаючи Popper.

Крім того, ресурс [8] дає змогу не лише копіювати готові шаблони, але й переходити в онлайн редактор щоб продивитись результат.

Наприклад, можна вибрати кнопку із низки пропонованих

Одразу надається розмітка, де у правому вехньому куті є дві кнопки:

– копіювання всього пропонованого коду;

– перехід на https://stackblitz.com/ для демонстрації прикладів.

HTML

```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</butto
<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>

<button type="button" class="btn btn-link">Link</button>
```

Завдання та хід виконання

1. Напишіть медіа-запити для гнучкості дизайну.
2. Кнопка кнопки Sign in на пристроях малих розмірів має бути нерухомою.
3. При натисненні Sign in має з'являтись модальне вікно з формою автентифікації.
4. Створіть адаптивне меню. Для зображення значків меню можна використовувати іконки чи спецсимволи.
5. Додайте до кнопки вгору властивість плавного переходу.
6. Створіть секцію зворотного зв'язку з відгуками декількох користувачів про сайт.
7. Додайте анімацію до певних елементів, не порушуючи концепцію сторінки.
8. Перейдіть на ресурс https://getbootstrap.com/docs/5.2/getting-started/introduction/ . Продумайте які шаблони елементів ви б хотіли використати.
9. Прочитайте які необхідно файли під'єднати. Перейдіть на сторінку старт https://getbootstrap.com/docs/5.2/getting-started/introduction/ та скопіюйте посилання.
10. Створіть нову сторінку, яку ви наповнюватимете елементами бібліотеки Bootstrap. Підключіть до неї необхідні файли за скопійованими посиланнями.
11. Наповніть шаблони, пропоновані бібліотекою Bootstrap власним вмістом.
12.
13. Відредагуйте код перед розміщенням сторінки на хостингу.
14. Зареєструйтесь на хостингу за уподобанням, наприклад. на netlify.com
15. Розмістіть та ресурсі свій проєкт.
16. Продемонструйте роботу викладачу.
17. Захистіть лабораторну роботу та розмістіть на сторінці курсу архів проєкту та посилання на хостинг.

Запитання для повторення

1. Що таке гнучка верстка?
2. Що таке адаптивна верстка?
3. Яка верстка називається чутливою.
4. Як можна організувати чутливу верстку?
5. Яке призначення властивості transition?
6. Як можна організувати анімацію?
7. Що потрібно зробити, щоб створити анімований елемент?

Література

1. Освітня програма «Інформаційні технології та управління проєктами першого (бакалаврського) рівня вищої освіти» [Електронний ресурс] – Режим доступу : https://mathmod.chnu.edu.ua/media/1721/op_knbak_2021.pdf
2. Освітня програма «Системний аналіз першого (бакалаврського) рівня вищої освіти» [Електронний ресурс] – Режим доступу : https://mathmod.chnu.edu.ua/media/1723/ op_sabak_2021.pdf
3. HTML Living Standart. [Електронний ресурс] – Режим доступу : https://html.spec.whatwg.org/
4. HTML5 підручник/ [Електронний ресурс] – Режим доступу : https://w3schoolsua.github.io/html/index.html
5. CSS. Notes for Professionals. [Електронний ресурс] – Режим доступу : https://books.goalkicker.com/CSSBook/
6. HTML5. [Електронний ресурс] – Режим доступу : https://books.goalkicker.com/HTML5Book/.
7. Figma. [Електронний ресурс] – Режим доступу : https://www.figma.com/
8. Bootstrap. [Електронний ресурс] – Режим доступу : https://getbootstrap.com/docs/
9. John Dean. Web Programming with HTML5, CSS, and JavaScript. [Електронний ресурс] – Режим доступу : https://www.pdfdrive.com/web-programming-with-html5-css-and-javascript-e187657772.html
10. Head First HTML and CSS: A Learner's Guide to Creating Standards-Based Web Pages. [Електронний ресурс] – Режим доступу : https://www.pdfdrive.com/head-first-html-and-css-a-learners-guide-to-creating-standards-based-web-pages-e158237724.html

Інформаційні ресурси

1. Український вебдовідник. [Електронний ресурс] – Режим доступу : https://css.in.ua/
2. HTML, CSS. [Електронний ресурс] – Режим доступу : https://www.w3schools.com/default.asp
3. Web technology for developers/ [Електронний ресурс] – Режим доступу до ресурсу : https://developer.mozilla.org/en-US/docs/Web....

Навчальне видання

Укладач:

Готинчан Тетяна Іванівна

Основи інтернет-технологій

Методичні вказівки та завдання до лабораторних робіт

Відповідальний за випуск **Черевко І.М.**
Комп'ютерний набір **Готинчан Т.І.**