МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

# МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт

## «Основи веб-технологій»

з дисципліни «Основи веб-технологій»
для студентів 122 спеціальності «Комп’ютерні науки»

Затверджено
редакційно-видавничою
радою університету,
протокол № __ від __.__.2023 р.

Харків
НТУ «ХПІ»
2023

Методичні вказівки до виконання лабораторних робіт з дисципліни «Основи веб-технологій» для студентів 122 спеціальності «Комп’ютерні науки» / уклад. А.О. Лисенко, І.В. Шуба. – Харків : НТУ «ХПІ». 2023 – 45 с.

Укладачі: А.О. ЛИСЕНКО, І.В. ШУБА

Рецензент М.А. ГРИНЧЕНКО

Кафедра стратегічного управління

# ЗМІСТ

**ВСТУП**.....4
**ЛАБОРАТОРНА РОБОТА №1**.....5
**ЛАБОРАТОРНА РОБОТА №2**.....9
**ЛАБОРАТОРНА РОБОТА №3**.....15
**ЛАБОРАТОРНА РОБОТА №4**.....18
**ЛАБОРАТОРНА РОБОТА №5**.....25
**ЛАБОРАТОРНА РОБОТА №6**.....33
**ЛАБОРАТОРНА РОБОТА №7**.....37
**КОНТРОЛЬНІ ЗАПИТАННЯ**.....40
**РЕКОМЕНДОВАНІ ДЖЕРЕЛА**.....41
**ДОДАТОК А**.....42
**ДОДАТОК Б**.....44

# ВСТУП

Веб-технології стали невід'ємною складовою нашого цифрового світу, забезпечивши новий спосіб, яким ми спілкуємось, працюємо та вивчаємо нові знання. Дисципліна «Основи веб-технологій» дозволяє усвідомити основні концепції, технології та принципи, які лежать в основі вебу, щоб створювати ефективні та привабливі веб-сайти та додатки.

Мета цих методичних вказівок – надати студентам глибоке розуміння основних технологій вебу та вміння застосовувати їх для розробки та підтримки веб-проектів. Ви отримаєте можливість ознайомитися з ключовими інструментами, мовою програмування JS та техніками, необхідними для створення функціональних та естетичних веб-рішень. Крім того, лабораторні роботи спрямовані на розвиток практичного досвіду у створенні веб-сайтів та роботі з їх складовими елементами.

Методичні вказівки складаються з семи лабораторних завдань, кожне з яких орієнтоване на певний аспект веб-технологій. Кожне завдання включає опис постановки задачі, необхідні теоретичні матеріали, приклади коду та практичні поради з вирішення конкретної проблеми. Крім того, вказівки містять рекомендації з ефективного використання ресурсів та джерел для додаткового самостійного вивчення.

Для успішного виконання лабораторних завдань потрібно мати базові знання з програмування та розуміння основ веб-розробки. Також важливо мати бажання експериментувати та вдосконалювати власні навички. Виконання кожної лабораторної роботи вимагає систематичного підходу та уваги до деталей, що дозволить вам отримати максимальну користь та навички у цій захоплюючій області.

**ЛАБОРАТОРНА РОБОТА №1**
**«Структура й принципи функціонування Веб»**

*Мета*: Познайомитись із структурою та специфікою розробки та функціонування веб. Розглянути типи сайтів їх структуру та реалізацію.

*Знати*: Структуру сайтів, принципи їх функціонування та особливості розробки користувацької та серверної частин.

*Вміти*: Розрізняти типи сайтів, їх структуру, знати будову сайтів та розрізняти фронтенд та бекенд розробку.

**Теоретичні відомості**

Веб (веб-сайти та веб-додатки) мають свою структуру та принципи функціонування, які допомагають забезпечити ефективну та зручну інтернет-присутність. Основні складові та принципи веб-систем:

1) Клієнт-серверна модель.

Один з основних принципів веб-архітектури - це клієнт-серверна модель. За цим принципом веб-додатки поділяються на дві основні компоненти: клієнт та сервер. Клієнт – це програмне забезпечення, яке запускається на боку користувача (наприклад, браузер), і він забезпечує інтерфейс для взаємодії з користувачем. Сервер - це програмне забезпечення, яке запускається на боку сервера та обробляє запити від клієнта, здійснює доступ до баз даних, логіку додатку та повертає результати до клієнта.

2) HTTP протокол.

Для обміну даними між клієнтом та сервером використовується протокол HTTP (Hypertext Transfer Protocol). HTTP забезпечує стандартизований механізм для взаємодії з веб-серверами, включаючи відправку запитів (GET, POST, PUT, DELETE тощо) та отримання відповідей (HTML сторінки, JSON дані тощо).

3) HTML, CSS та JavaScript.

HTML (Hypertext Markup Language), CSS (Cascading Style Sheets) та JavaScript - це основні технології для створення клієнтської сторони веб-додатків. HTML використовується для створення структури веб-сторінок, CSS - для стилізації та оформлення сторінок, а JavaScript - для динамічного управління поведінкою сторінок, інтерактивністю та взаємодією з сервером.

4) Бази даних.

Веб-додатки часто використовують бази даних для зберігання, організації та отримання даних. Серверна частина веб-додатків взаємодіє з базою даних для збереження та зчитування інформації.

Фронтенд та бекенд. Веб-додатки можуть бути розділені на фронтенд (клієнтську частину) та бекенд (серверну частину). Фронтенд забезпечує

інтерфейс для користувачів та взаємодію з сервером, а бекенд забезпечує обробку запитів, логіку додатку та доступ до баз даних.

Інтерактивність та взаємодія:

Одні з основних переваг веб-додатків полягають у їхній здатності взаємодіяти з користувачами. Веб-додатки можуть реагувати на взаємодію користувача, виконувати дії на стороні клієнта та сервера та оновлювати інформацію без необхідності перезавантаження сторінки.

Загалом, веб-продукти мають складну структуру, що базується на взаємодії між клієнтами та серверами, використанні стандартних протоколів та технологій та можливості взаємодії та динамічного оновлення інформації.

***Види сайтів***:

1. Сайт-візитівка.
2. Комерційні сайти:
   а) сайт-вітрина
   б) промо-сайти
   в) інтернет-магазин
3. Інформаційні сайти:
   а) тематичні інформаційні сайти
   б) новинні сайти
   в) блоги
4. Освітні ресурси:
   а) каталоги освітніх ресурсів
   б) платформи для он-лайн курсів
   в) електронні підручники
   г) он-лайн словники, енциклопедії
   д) сервіси для створення презентацій, інфографіки тощо.
5. Соціальні проєкти (блоги, соціальні мережі).
6. Веб-сервіси:
   а) каталоги
   б) пошукові системи
   в) поштові сервіси
   г) сайт хостинг
   д) дошки оголошень

Розробка веб-сайту може бути складним процесом з використанням різних інструментів (рис.1), який включає кілька етапів. Нижче наведені загальні етапи розробки веб-сайту:

1) Перший етап - це збір і аналіз вимог до веб-сайту. Це означає розуміння цілей сайту, цільової аудиторії, функціональних вимог, дизайну, технічних вимог, бюджету та графіку розробки.

2) Планування. На цьому етапі створюється детальний план розробки, включаючи визначення технологій, вибір платформи, структуру сторінок, складність функціоналу, терміни виконання і розподіл обов'язків між командою розробників.
3) Дизайн веб-сайту охоплює створення макетів, вибір кольорів, шрифтів, графіки та інтерфейсу. Важливо, щоб дизайн був орієнтованим на користувача і відповідав бренду або цілі компанії.
4) Розробка. На цьому етапі розробляються функціональність та динамічні елементи веб-сайту. Програмісти працюють зі збереженням даних, створенням серверної та клієнтської частин, інтеграцією з базами даних та іншими сторонніми системами.
5) Тестування. Після завершення розробки веб-сайту відбувається тестування. Веб-сайт перевіряється на різних пристроях та браузерах, щоб впевнитися, що він працює належним чином і є безпечним. Також виконуються тестування з використанням різних сценаріїв, щоб переконатися, що функціонал працює належним чином.
6) Випробування та оптимізація продуктивності. Забезпечення швидкої роботи веб-сайту і мінімізація часу завантаження сторінок є важливим аспектом. Перевіряються продуктивність, оптимізуються файли, кешування, використовуються CDN (мережі доставки контенту) для покращення швидкості сайту.
7) Реліз та розгортання. Після успішного завершення тестування сайт готовий до релізу. Розгортання означає розміщення сайту на живому сервері і доступність його для відвідувачів.
8) Підтримка та оновлення. Після запуску сайту команда займається підтримкою, виправленням помилок та оновленнями для забезпечення безперебійної роботи та покращення функціональності.

Ці етапи розробки можуть варіюватися в залежності від розміру проекту, специфікацій та методологій роботи команди.


Рис. 1 – Інструменти для створення сайті

**Завдання**

1. Обрати собі для роботи протягом семестру вид та тематику сайту.
2. Продумати його структуру, дизайн, наповнення. Зробити макет.

Варіанти тематик:

1) Сайт – тлумачний словник з ілюстраціями
2) Сайт про собак
3) Сайт з фотографіями та описом квітів з обласних центрів України
4) Сайт – кулінарна книга з рецептами української кухні
5) Інтернет-магазин дитячих іграшок
6) Інтернет-магазин книг
7) Сайт – довідник з музичних термінів
8) Сайт-візитка одного з казкових героїв
9) Сайт туристичної агенції для подорожей Україною
10) Сайт про автомобілі
11) Сайт про екологію
12) Сайт-довідник віртуальні подорожі Україною
13) Сайт – довідник про мови програмування
14) Сайт – довідник про музичні напрямки
15) Сайт – довідник про самогоноваріння
16) Сайт, присвячений улюбленому письменнику
17) Сайт, присвячений улюбленій музичній групі чи співаку/співачці
18) Сайт наукових новин
19) Сайт анекдотів та карикатур
20) Сайт аніме
21) Сайт – довідник з мови програмування C++
22) Сайт, присвячений певній країні (на вибір студента)
23) Сайт, присвячений різним породам котів
24) Інтернет-магазин настільних ігор
25) Інтернет-магазин квітів
26) Сайт, присвячений певній політичній організації (на вибір студента)
27) Інтернет-магазин будівельних матеріалів
28) Сайт фірми з продажу та ремонту смартфонів
29) Сайт спортивного клубу
30) Сайт кінотеатру
31) Сайт кондитера
32) Сайт для дошкільного навчального закладу

**ЛАБОРАТОРНА РОБОТА №2**
**«Базові теги HTML»**

*Мета*: Познайомитись із синтаксисом, основними тегами й атрибутами мови html.
*Знати*: Структуру html-документу, основні теги мови й методи форматування тексту.
*Вміти*: Компонувати текстову інформацію при створенні гіпертекстових документів з використанням простих текстових редакторів. Формувати гіпертекстові посилання й списки й створювати на цій основі зв'язані гіпертекстові сторінки.

## Теоретичні відомості

HTML (англ. HyperText Markup Language) – стандартна мова розмітки гіпертекстових документів (веб-сторінок) в Інтернеті. Мова HTML інтерпретується браузерами. Отриманий в результаті інтерпретації відформатований текст відображається на екрані монітора комп'ютера або мобільного пристрою. Розробкою та впровадженням стандартів для мережі Інтернет, в тому числі й стандартів для мови HTML займається організація W3C (англ. World Wide Web Consortium (https://www.w3.org/) – консорціум Всесвітньої павутини).

Структура сайту відображає його організацію та ієрархію сторінок, що включаються в його склад. Гарно продумана структура сайту сприяє зручності навігації, поліпшує користувацький досвід та сприяє зростанню популярності та відвідуваності сайту. Ось деякі основні компоненти структури сайту:

1) Головна сторінка (Homepage). Є вхідною точкою на сайт і відображає основні елементи та інформацію про компанію, продукти або послуги, які вона пропонує. Це перша сторінка, яку бачать відвідувачі сайту, тому важливо забезпечити привабливий дизайн та зручну навігацію.

2) Меню та навігація. Зазвичай сайти мають глобальне меню або панель навігації, яка дозволяє користувачам легко переходити між різними сторінками та розділами сайту. Меню може бути розташоване у верхній частині сторінки (*header*), боковому барі (*sidebar*) або у нижній частині (*footer*). Добре організована навігація допомагає користувачам знайти потрібну інформацію швидко та легко.

3) Розділи та підрозділи. Сайт може бути розділений на різні розділи та підрозділи, що відповідають різним тематикам або функціоналу. Наприклад, у випадку компаній це можуть бути, наприклад, розділи «Про нас», «Продукти», «Контакти» та інші. Підрозділи можуть деталізувати інформацію в межах основного розділу.

4) Статичні та динамічні сторінки. Сайти можуть включати як статичні, так і динамічні сторінки. Статичні сторінки є постійними та незмінними, їх вміст

визначається наперед. Динамічні сторінки створюються з допомогою шаблонів та баз даних, їх вміст залежить від користувача або контексту та може змінюватися в реальному часі.

5) Контактна інформація. На сайті повинна бути надана контактна інформація, яка дозволяє відвідувачам зв'язатися з компанією, наприклад, контактна форма, адреса, електронна пошта, телефонні номери тощо.

6) Посилання та внутрішні сторінки. Важливо створити зручну ієрархію посилань між різними сторінками сайту, щоб відвідувачі могли легко переходити між різними сторінками. Внутрішні посилання допомагають відвідувачам знайти пов'язану інформацію.

HTML складається з різних елементів, які називаються тегами. Теги допомагають вказати браузеру, як повинен відображатися вміст сторінки. Теги бувають одинарні та парні (контейнер) відкриваючий і закриваючий.

Теги в HTML можна класифікувати за різними критеріями. Ось декілька основних типів тегів в HTML (рис.2), враховуючи їх функціональне призначення:

***Теги структури***:

<*!DOCTYPE*>: Декларація типу документа, яка вказує на версію HTML, використовується на початку кожного HTML-документа.

<*html*>: Визначає початок HTML-документа.

<*head*>: Містить інформацію про документ, таку як заголовок сторінки, метатеги, зовнішні файли CSS та ін.

<*body*>: Містить весь вміст сторінки, який буде відображений на веб-сторінці, такі як текст, зображення, посилання тощо.

***Текстові теги:***

<*p*>: Визначає абзац тексту.

<*h1*> to <*h6*>: Теги заголовків різних рівнів.

<*a*>: Визначає посилання.

<*span*>: Використовується для вибіркового застосування стилів до текстових елементів.

<*strong*>: Виділяє текст жирним шрифтом.

<*em*>: Виділяє текст курсивом.

***Теги списків***:

<*ul*>: Використовується для створення неупорядкованого списку (буллет-списку).

<*ol*>: Використовується для створення упорядкованого списку (нумерованого списку).

<*li*>: Елемент списку.

***Теги для зображень та медіа:***

<*img*>: Відображає зображення на сторінці.

<*video*>: Відтворює відео.

<*audio*>: Відтворює аудіо.

***Теги для форм та вводу:***

<*form*>: Визначає форму для збору даних від користувача.

<*input*>: Визначає поле вводу для форми.

<*textarea*>: Визначає поле для введення тексту більшого обсягу.

***Теги для таблиць:***

<*table*>: Визначає таблицю.

<*tr*>: Визначає рядок таблиці.

<*td*>: Визначає комірку таблиці.

***Теги мета-інформації***:

<*title*>: Визначає заголовок сторінки, який відображається на вкладці браузера або у списку закладок.

<*meta*>: Визначає метатеги, які містять додаткову інформацію про сторінку (наприклад, мову, автора, ключові слова).

***Теги для групування елементів:***

<*div*> : є блоковим елементом і призначений для виділення фрагмента документу з метою зміни його стилю. Щоб не описувати кожен раз стиль всередині тега, можна виділити стиль в зовнішню таблицю стилів, а для тегу додати атрибут class або id з ім'ям селектора

<*span*>: призначений для визначення стилю малих елементів документа. За допомогою тега  можна виділити частину інформації всередині інших тегів і встановити для неї свій стиль. Наприклад, всередині абзацу <p> можна змінити колір і розмір першої літери, якщо додати початковий і кінцевий тег  і визначити для нього стиль тексту. Щоб не описувати кожен раз стиль всередині тега, можна виділити стиль в зовнішню таблицю стилів, а для тегу додати атрибут class або id з ім'ям селектора.

<*!--……-->*: Коментар, текст який має довідковий характер для розробників.

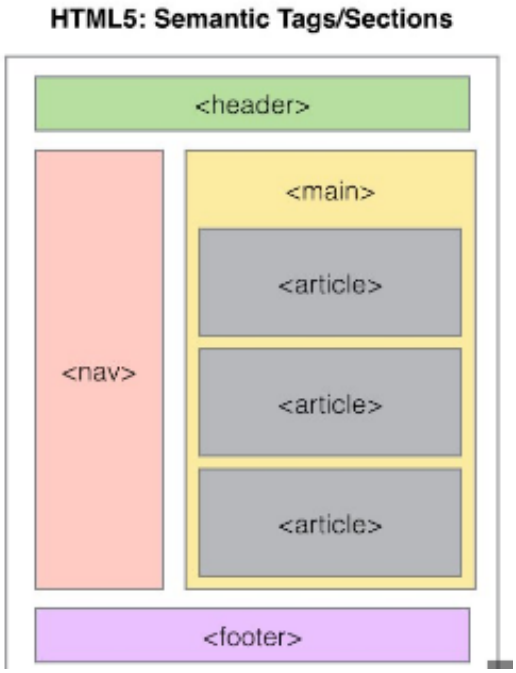



Рис. 2 – Семантичні теги в HTML

| **Тег** | **Опис** | |
|---|---|---|
| Header | Заголовок документа або розділу.<br>Зазвичай це велика смуга вгорі сторінки, з великим заголовком та/або логотипом. Тут наводиться загальна інформація про веб-сайт, що не змінюється від сторінки до сторінки. | |
| Nav | Контейнер для навігаційних посилань.<br>Посилання на основні розділи сайту; зазвичай це кнопки, посилання чи вкладки. Як і заголовок, навігація залишається незмінною на всіх сторінках сайту. | |
| Section | Розділ в документі. | Велика область в центрі сторінки, що містить, в основному, унікальний контент даної сторінки, наприклад відео, розповідь або карту, яку ви хочете переглянути, або заголовки новин і т.д. одна з частин сайту, яка безперечно змінюватиметься від сторінки до сторінки! |
| Article | Незалежна самодостатня стаття. | |
| Aside | Бічна панель. Зазвичай містить деяку другорядну інформацію, посилання, цитати, рекламу тощо. Зазвичай вона відноситься до вмісту в основному контенті. | |
| Footer | Нижній колонтитул документа або розділу.<br>Смуга в нижній частині сторінки, яка зазвичай містить повідомлення про авторські права або контактну інформацію. Це місце для розміщення загальної інформації. | |

Теги в HTML мають певну будову (рис.3 а 4), яка допомагає браузеру правильно інтерпретувати та відображати вміст веб-сторінок. Кожен тег складається з певних компонентів, а саме:

1) ***Відкриваючий тег*** – це початок тегу і він вказує браузеру, коли починається діяльність тега. Відкриваючий тег визначається шляхом поставлення знака "<" і назви тегу. Наприклад, * *.
2) ***Атрибути***. Деякі теги можуть мати атрибути, які надають додаткову інформацію або налаштування тега. Атрибути вказуються всередині відкриваючого тега і можуть мати значення. Наприклад,

   *<img src="image.jpg" alt="Зображення">*.
3) ***Закриваючий тег*** - це закінчення тегу, і він вказує браузеру, коли закінчується діяльність тега. Закриваючий тег має такий же вигляд, як відкриваючий, але містить додатковий знак "/". Наприклад, .

Однак, не всі теги вимагають закриваючого тега, а такі теги називаються самозакриваючими або одиночними тегами. У таких випадках, тег складається лише з відкриваючого тегу, і він закінчується знаком "/>". Наприклад: <br> (перенесення рядка), <hr> (горизонтальна лінія), <img>.

```
<p>Це параграф з текстом.</p>
```

а)

```
<img src="image.jpg" alt="Зображення">
```

б)

```
<p>Це перший рядок.<br>Це другий рядок.</p>
```

в)

Рис. 3 – Приклад тегів: а) звичайний тег; б) тег з атрибутом; в) самозакриваючого тега <br>


Рис.4 – Будова тега з атрибутом

**Завдання**

Згідно обраного виду та тематики сайту створити веб-сторінки, використовуючи лише інструменти мови HTML, які містять обов'язкові складові:

1) текстові елементи (не менше трьох видів заголовків, текстові блоки з контентом відповідно тематиці сайту; таблицю; не менше; відео та аудіоконтент;
2) ілюстрації – не менше п’яти зображень різного розміру;
3) списки – два види списків (нумерований і маркований);
4) таблиця;
5) посилання, переходи між сторінками сайту (не менше 5 сторінок).
6) Елементи розмітки веб-сторінки.

**ЛАБОРАТОРНА РОБОТА №3**
«Хостинг сайту та веб-сервери»

*Мета*: Познайомитись веб-серверами та способами розміщення веб-сайту в мережі Інтернет.
*Знати*: Спосіб розміщення веб-сайту в мережі Інтернет та основи .
*Вміти*: Завантажувати та налаштовувати веб-серверу Apache та писати код веб-серверу на Python.

## Теоретична частина

Хостинг сайту та веб-сервери – це основні компоненти, які дозволяють зробити сайт доступним у Інтернеті.

***Хостинг*** – це послуга, яка надається провайдерами (хостинг-провайдерами) та дозволяє зберігати веб-сайт на їх серверах і зробити його доступним для користувачів через Інтернет. Коли власник сайту купує хостинг, він отримує доступ до серверного простору, де може розміщувати файли сайту, бази даних, електронну пошту та інші ресурси.

Типи хостингу можуть варіюватися в залежності від потреб сайту:

1) спільний хостинг це коли сайт розміщується на сервері з іншими сайтами. Він є більш економічним, але ресурси таких серверів діляться між різними користувачами.
2) віртуальний приватний сервер (VPS). Власник сайту отримує окремий віртуальний сервер зі своїм набором ресурсів та контролем, але фізичний сервер ділиться між декількома VPS.
3) Виділений сервер. Власник сайту отримує повний контроль та доступ до фізичного сервера, що надає максимальну продуктивність, але вимагає більшого бюджету.

**Веб-сервер** – це програмне забезпечення, яке працює на сервері та обробляє запити від користувачів, що надійшли через Інтернет. Він зберігає веб-сайт, його статичні та динамічні ресурси та відповідає на запити користувачів, надсилаючи їм відповідні файли, дані або результати виконання запитів.

Для більшості веб-сайтів, що зберігаються на хостингу, використовується веб-сервер Apache, Nginx або IIS (Internet Information Services). Вони забезпечують обробку HTTP-запитів, статичних файлів, виконання скриптів та взаємодію з базами даних для динамічного створення сторінок.

***Apache*** – це один з найпоширеніших веб-серверів, який використовується для обробки HTTP-запитів та надання веб-сторінок користувачам через Інтернет. Він є одним з найстаріших та найбільш надійних веб-серверів та грає важливу роль у хостингу веб-сайтів.

Деякі характеристики та особливості Apache:

1) Apache є вільним програмним забезпеченням з відкритим вихідним кодом, що означає, що його можна безкоштовно використовувати, модифікувати та поширювати його за умови дотримання ліцензії Apache License.
2) підтримка різних платформ, наприклад може працювати на різних ОС, таких як Windows, macOS, Linux та інші.
3) модульність - Apache побудований на модульній архітектурі, що дозволяє легко додавати та видаляти функціональність через модулі. Це дозволяє налаштовувати сервер залежно від потреб веб-проекту.
4) висока надійність та продуктивність, він може обробляти велику кількість запитів і роздавати сторінки швидко.
5) підтримка великої кількості протоколів. Apache підтримує HTTP, HTTPS, FTP, і багато інших протоколів, що дозволяє використовувати його для різних типів додатків та сервісів.
6) Налаштування за допомогою файлу конфігурації. Apache конфігурується за допомогою текстового файлу конфігурації, що дозволяє здійснювати різні налаштування сервера.

Крім веб-серверів, існують інші види серверів, такі як сервери баз даних (наприклад, MySQL, PostgreSQL), сервери електронної пошти (наприклад, Postfix, Exim) тощо. Кожен з них грає свою роль у забезпеченні повного функціонування веб-сайту.

Також, для написання простих веб-серверів можна використовувати мову програмування Python за допомогою вбудованих бібліотек, таких як ***http.server*** або ***socket***. Приклад коду, який демонструє базовий механізм роботи веб-сервера, що використовує бібліотеку http.server пердставлено на рисунку 3. Запустити код можна, зберігши його у файлі з розширенням .py через командний рядок.

```python
import http.server
import socketserver

# Вказуємо порт, на якому буде працювати сервер
PORT = 8000

# Задаємо конфігурацію сервера
Handler = http.server.SimpleHTTPRequestHandler

# Запускаємо сервер на вказаному порту
with socketserver.TCPServer(("", PORT), Handler) as httpd:
    print(f"Сервер запущено на порту {PORT}")
    # Постійно обслуговуємо клієнтські запити
    httpd.serve_forever()
```

Рис. 5 – Код простого веб-серверу, який буде обслуговувати файли в поточному каталозі.

**Завдання**

1) Завантажити та налаштувати Apache, ознайомившись з мережевою документацією на сайті. Налаштувати віртуальний сервер. Розмістити розроблені раніше сторінки веб-сайту на веб-сервері.
2) Розмістити розроблені сторінки веб-сайту на сервері написаному Python

**ЛАБОРАТОРНА РОБОТА №4**
«Основи CSS»

*Мета*: Познайомитися із синтаксисом та основними елементами CSS.
*Знати*: Способи створення різних дизайнів сайту за допомогою CSS.
*Вміти*: Встановлювати стилі для HTML-елементів за допомогою селекторів CSS

## Теоретична частина

***CSS*** (Cascading Style Sheets) - це мова стилізації, яка використовується для оформлення та розмітки веб-сторінок. За допомогою CSS можна задавати зовнішній вигляд та візуальний стиль елементів HTML, таких як текст, зображення, фон, розміри, відступи, кольори та інші аспекти веб-сторінки.

CSS дозволяє створювати красиві та привабливі веб-дизайни, поліпшує користувацький досвід та забезпечує консистентний вигляд веб-сайту на різних пристроях і браузерах. Для використання CSS на веб-сторінці, включати стилі можна безпосередньо в HTML-документ або розміщувати їх у зовнішніх CSS-файлах та посилатися на них у HTML.

Селектор – ім'я стилю, у якому зазначені параметри форматування.

Типи селекторів:

- селектори тегів (стилізують всі теги <p>),
- селектори ідентифікаторів (стилізують елемент з вказаним ідентифікатором, наприклад, #header),
- селектори класів (стилізують всі елементи з вказаним класом, наприклад, .highlight) тощо.

Властивості – визначають стилі, які будуть застосовуватися до вибраних елементів. Кожна властивість має значення, яке визначається за допомогою двокрапки. Наприклад:

***color: red***; (встановлює колір тексту).
***font-size: 16px***; (встановлює розмір шрифту).

Приклад найпоширеніших селекторів:

- ✓ color: колір тексту;
- ✓ text-align: де елементи розміщуються на сторінці;
- ✓ background-color: можна встановити будь-який колір;
- ✓ width: ширина у пікселях або відсотках від сторінки;
- ✓ height: висота у пікселях або відсотках сторінки;
- ✓ padding: скільки місця слід залишити всередині елемента;
- ✓ margin: скільки місця слід залишити поза елементом;
- ✓ font-family: тип шрифту для тексту на сторінці;
- ✓ font-size: кегль шрифту у пікселях;

✓ border: тип межі (суцільна, штрихова тощо) та колір.

CSS має концепцію ***каскадування***, що дозволяє задавати багато стилів для одного елемента, а потім визначити, який стиль має переважати. Порядок підключення CSS-файлів або місце оголошення стилів впливає на їх пріоритет. Каскадні стилі дозволяють задавати різні стилі для елементів в залежності від їх контексту та розташування на сторінці.

Синтаксис CSS має структуру (рис.6):

*Селектор {*
*властивість1: значення;*
*властивість2: значення;*
...
*властивістьN: значення;*
*}*

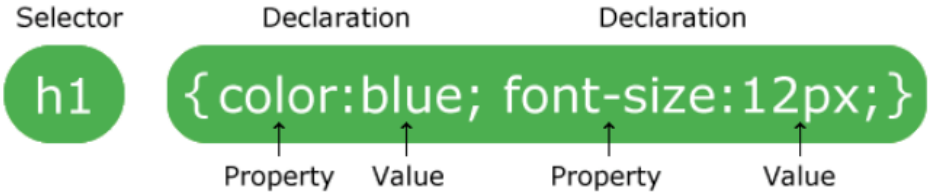



Рис. 6 – Будова селектору CSS[1]

CSS має наступний пріоритет застосування:
1. рядковий стиль;
2. id;
3. клас;
4. тип елемента.

**Вставка в документ CSS-стилів**

1. Посиланням в HTML документі на .css файл, що містить визначення стилів. При такому підході Ви можете посилатися на той самий файл стилів з будь-якої кількості документів. Тоді, при зміні стилів у цьому файлі всі документи поміняють свій вигляд;

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

[1] CSS Syntax and Selectors // https://html5css.ru/css/css_syntax.php

2. Визначити всі стилі, що використовуються на початку сторінки, а потім використовувати. При такому підході можна змінювати стилі на початку документа (один раз) і зміни відіб'ються у всіх місцях сторінки, де ці стилі використовуються. Наприклад:

```
<head>
<style type="text/css">
 H1 {
 font-size: 120%;
 font-family: Verdana, Arial, Helvetica, sans-serif;
 color: #336
 }
</style>
</head>
```

3. Вставляти вказівки на правила форматування безпосередньо в тег. Звичайно це використовується для унікального форматування, що більше (крім як у цьому місці документа) ніде не потрібно. Наприклад (додається кольоровий заголовок третім способом):

```
<!DOCTYPE html>
<html lang="en">
 <head>
 <title>Hello!</title>
 </head>
 <body>
 <h1 style="color: blue; text-align: center;">A Colorful
Heading!</h1>
 Hello, world!
 </body>
<html>
```

**Властивості CSS**

Існує дуже багато властивостей CSS, всі ми розглянути не зможемо, але так само, як і елементи HTML, як правило, їх легко знайти за запитом, наприклад, «змінити шрифт на синій CSS». Деякі з найпоширеніших:

- color: колір тексту;
- text-align: де елементи розміщуються на сторінці;
- background-color: можна встановити будь-який колір;
- width: ширина у пікселях або відсотках від сторінки;
- height: висота у пікселях або відсотках сторінки;
- padding: скільки місця слід залишити всередині елемента;
- margin: скільки місця слід залишити поза елементом;

 font-family: тип шрифту для тексту на сторінці;
 font-size: кегль шрифту у пікселях;
 border: тип межі (суцільна, штрихова тощо) та колір.

**Спадкування**. HTML документ має ієрархічну структуру. Тіло документа вкладене в теги <BODY>...</BODY>. В середині документа можуть бути розділи, вкладені в теги <DIV>...</DIV>. Усередині розділів можуть бути ще розділи й т.д. Ідея спадкування полягає в тому, що всі формати, які визначаються для <BODY>, будуть успадковуватися всіма розділами. Можна перевизначити в розділі деякі формати, а всі інші він успадкує від <BODY>. Таким способом, розділ, вкладений в інший розділ, успадковує формати розділу, що його містить, але може щось перевизначити для себе.

Вкладені розділи успадковують форматування батьків і можуть додавати своє (а якщо треба, то й змінювати батьківське). Однак, як тільки вкладений розділ завершився й триває батьківський, всі його формати відновлюються.

***Адаптивність веб-сайту*** (або Responsive Web Design) – це підхід до веб-розробки, який дозволяє автоматично адаптувати вигляд і макет веб-сторінки до різних розмірів екранів на різних пристроях, таких як комп'ютери, планшети і смартфони. Головною метою адаптивного дизайну є забезпечення оптимального користувацького досвіду незалежно від розміру екрану.

Основні принципи адаптивного дизайну включають:

1) гнучкість – верстка сторінки розрахована на те, щоб елементи можна було адаптувати та перегруповувати залежно від розміру екрану. Це може включати перестановку елементів, зміну розмірів, приховування деяких елементів або зміну розміщення контенту.
2) використання CSS медіа-запитів можна задавати різні стилі для різних пристроїв та екранів. Це дозволяє забезпечити різний зовнішній вигляд та розташування елементів залежно від ширини екрана.
3) мобільні меню. Для малих екранів часто використовуються мобільні меню, які згортаються в іконку бургер-меню, щоб зекономити місце та забезпечити більший зручність користувачам.
4) Гнучке зображення. Використання гнучких зображень (зображення, що адаптуються до розміру екрану) допомагає забезпечити оптимальне відображення зображень на різних пристроях.

Способи покращення зовнішнього вигляду сайта:

1) додавання у заголовок HTML-файлів рядок коду, який вказує мобільному пристрою використовувати область перегляду тієї ж ширини, що і ширина пристрою, що використовується, а не значно більшу.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

2) застосування медіазапитів. Медіазапити – це способи зміни стилю сторінки на основі того, як саме переглядається сторінка.

```
<!DOCTYPE html>
<html lang="en">
 <head>
 <title>Screen Size</title>
 <style>
 @media (min-width: 600px) {
 body {
 background-color: red;
 }
 }
 @media (max-width: 599px) {
 body {
 background-color: blue;
 }
 }
 </style>
 </head>
 <body>
 <h1>Welcome to the page!</h1>
 </body>
</html>
```

3) використання атрибута CSS – ***flexbox*** дозволяє легко перенести елементи до наступного рядка, якщо вони не вміщуються горизонтально.

```
<!DOCTYPE html>
<html lang="en">
 <head>
 <title>Screen Size</title>
 <style>
 #container {
 display: flex;
 flex-wrap: wrap;
 }
 #container > div {
 background-color: green;
```

```
font-size: 20px;
margin: 20px;
padding: 20px;
width: 200px;
}
</style>
</head>
<body>
<div id="container">
<div>Some text 1!</div>
<div>Some text 2!</div>
<div>Some text 3!</div>
<div>Some text 4!</div>
<div>Some text 5!</div>
<div>Some text 6!</div>
<div>Some text 7!</div>
<div>Some text 8!</div>
<div>Some text 9!</div>
<div>Some text 10!</div>
<div>Some text 11!</div>
<div>Some text 12!</div>
</div>
</body>
</html>
```

***Валідність HTML-верстки*** –це її відповідність стандартам організації TheWorldWideWebConsortium (W3C). Відсутність помилок в розмітці документу - це один з головних показників якості розмітки. Перевірка розмітки на помилки та відповідність стандарту може бути проведена автоматично за допомогою он-лайн сервісу W3C чи за допомогою різних програм «валідаторів»

**Завдання**

1) Для сторінок веб-сайту, створеного у попередніх робота застосувати CSS для візуального оформлення сторінок:

- всі стилі сайту мають бути прописані у файлі style.css;
- для різних частин сторінки використати різні кольори фону, на одній із сторінок реалізувати;
- для текстових блоків використати стилі: три кольори, два шрифти, три розміри елементів тексту;
- для зображень додати тінь, скруглення кутів, на одній із сторінок реалізувати обтікання рисунків текстом.

2) Використовуючи медіазапити здійснити адаптивну верстку так щоб при зменшенні екрану пристроїв (настільний комп’ютер, ноутбук, планшет, смартфон) змінювалися:

- фон сторінки;
- розмір та розміщення рисунків;
- розмір тексту та його розміщення.

3) Перевірити відображення макету в різних браузерах та виправити за необхідності помилки.

**ЛАБОРАТОРНА РОБОТА №5**
«JS у веб-технологіях»

*Мета*: Познайомитися із синтаксисом, основними елементами мови JavaScript.
*Знати*: Структуру HTML документа, основні елементи мови JavaScript.
*Вміти*: Вміти писати прості сценаріїв JavaScript та використання готових JS-бібліотек.

## Теоретична частина

### Вставка в код сторінки

JavaScript являє собою повністю інтерпретуєму мову опису сценаріїв із програмним кодом скрипта, що поставляється користувачеві у відкритому вигляді.

1) Безпосередньо в тіло HTML документа за допомогою спеціального тегу <SCRIPT>. . .</SCRIPT> . Браузер аналізує вміст, що перебуває між цими тегами й для виконання сценарію пускає в хід той або інший інтерпретатор. Проте, для усунення можливих різночитань і колізій є можливість указувати мову і його версію безпосередньо в тілі тегу скрипта:

```
<SCRIPT language JavaScript 1.1> ...
</SCRIPT>
```

2) Розміщення скриптів в окремих файлах. Цей підхід найбільш прийнятний при використанні скриптів досить великого обсягу, а також для зберігання службових процедур. Якщо ми зберігаємо наш скрипт у файлі з ім'ям filel.js, то включити його в код сторінки ми можемо в такий спосіб:

```
<SCRIPT src="file.JS" X/SCRIPT>
```

У цьому випадку між тегами опису скрипта нічого вставляти не потрібно - необхідний код перебуває адже у файлі filel.js. Ніщо не заважає нам указувати назву мови і його версію в тегу опису скрипта й у цьому випадку.

Використання скриптів у тілі HTML-документа

```
<HTML>
<HEAD> <! - - - Тут дані заголовка ---i-> </HEAD>
<BODY>
<! ---i- HTML-Код -i->
<SCRIPT> // Код скрипта </SCRIPT>
<! ---i- HTML-Код-i->
<SCRIPT language VBScript> // Код скрипта мовою VBScript
</SCRIPT>
<! ---i- HTML-Код-i>
<SCRIPT language JavaScript 1.3 > // Код скрипта мовою JavaScript
</SCRIPT>
```

```
<! ---i- HTML-Код-i->
<! ---i- Підключаємо скрипт із зовнішнього файлу ---i->
<SCRIPT src=" script-l.js" language JavaScript 1.1 > </SCRIPT>
<! ---i-А якщо браузер не знає про скриптах ? ---i>
<NOSCRIPT>
<CENTER> <H1> !!! Ваш браузер не має підтримки скриптов. Радимо
обзавестися новим програмним забезпеченням !!! </H1>
</CENTER>
</NOSCRIPT>
</BODY>
</HTML>
```

**Синтаксис JavaScript**.

JavaScript є справжньою мовою програмування з усіма властивими йому атрибутами: змінними, операторами контролю процесу, командами ввод/виводу й ін. Синтаксис JavaScript багато в чому схожий із синтаксисом C++. У той же час конструкції JavaScript досить спрощені, що дозволяє освоювати його людям, що не є досить глибокими знавцями програмування. Для написання коду програм на JavaScript використовується стандартний набір латинських символів. У рядкових виразах можна використовувати символи національних алфавітів. Застосовуються також спеціальні символи, такі як \n - новий рядок, \t - табуляція, \b - вибій, \r - повернення каретки. Як коментар використовується послідовність символів //. У назвах операторів й іменах змінних заголовні й малі літери є рівнозначними

Більшість операторів JavaScript подібні до відповідних в C++:

(+) додавання, зчеплення рядків;
(-) вирахування;
(*) множення;
(/) розподіл;
( %) залишок від цілочисельного розподілу;
(++) інкремент;
( --) декремент;
(a+=b) a=a+b;
( a-a-=b) a= a-b;
(&) логічний AND (довгий варіант);
(&&) логічний AND (короткий варіант);
(|) логічний OR (довгий варіант);
(||) логічний OR (короткий варіант);
(!) логічне заперечення;
(>) більше;
(<) менше;
(>>) зрушення праворуч;
(<<) зрушення ліворуч.

```javascript
// Змінні (variables)
let name = 'John'; // Декларація змінної let
var age = 30; // Старий спосіб декларації змінної var (не рекомендовано)
const PI = 3.14; // Константа (незмінювана змінна)

// Типи даних
let num = 10; // Число
let str = 'Hello'; // Рядок
let bool = true; // Булевий тип (true або false)
let arr = [1, 2, 3]; // Масив
let obj = { name: 'John', age: 30 }; // Об'єкт

// Нульова та неіснуюча змінні
let nullVar = null; // Призначення змінній значення null
let undefinedVar; // Значення за замовчуванням - undefined (не призначено)

// Операції змінних
let x = 5;
let y = 10;
let sum = x + y; // Додавання
let product = x * y; // Множення
let isGreater = x > y; // Порівняння (поверне false)

// Конкатенація рядків
let firstName = 'John';
let lastName = 'Doe';
let fullName = firstName + ' ' + lastName; // Результат: "John Doe"
```

Рис 7 – Скрін частини коду де виражено змінні

javascript Copy code

```javascript
let num = 10;

// Умовний оператор if
if (num > 0) {
  console.log('Число додатне');
} else if (num < 0) {
  console.log('Число від\'ємне');
} else {
  console.log('Число дорівнює нулю');
}

// Тернарний оператор
let result = num > 0 ? 'Число додатне' : 'Число невід\'ємне';
console.log(result);
```

Рис.8 – Умовні конструкції в JS

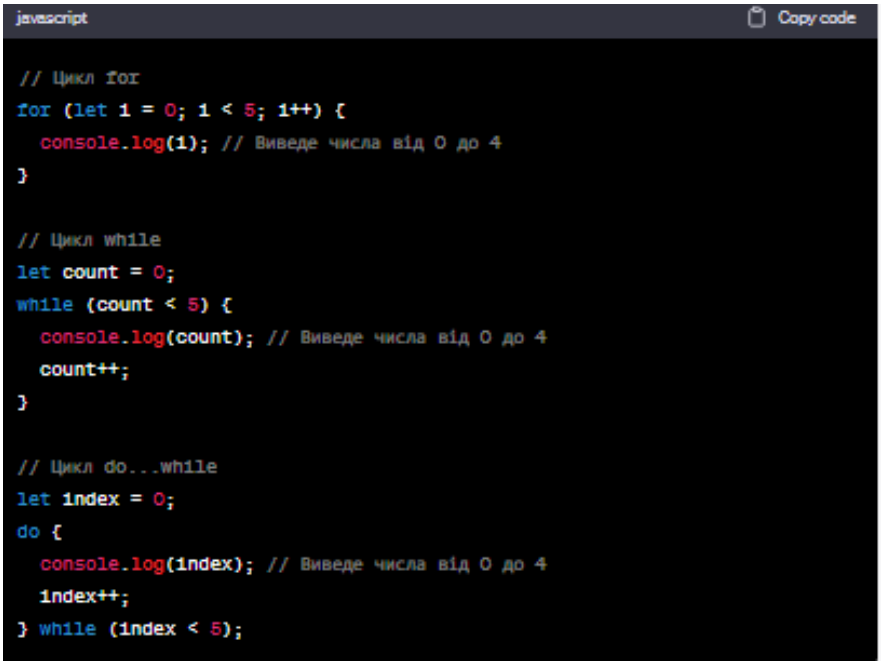



Рис.9 – Цикли в JS

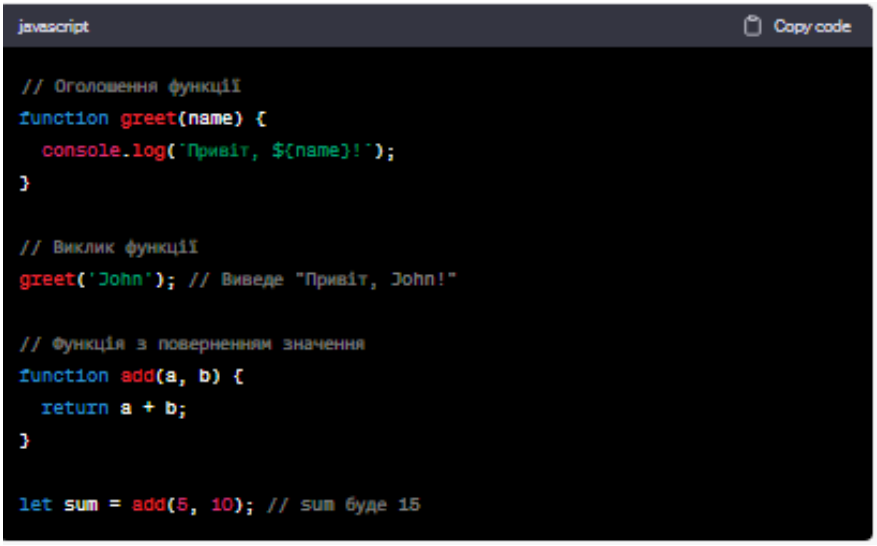



Рис.10 – Функції в JS

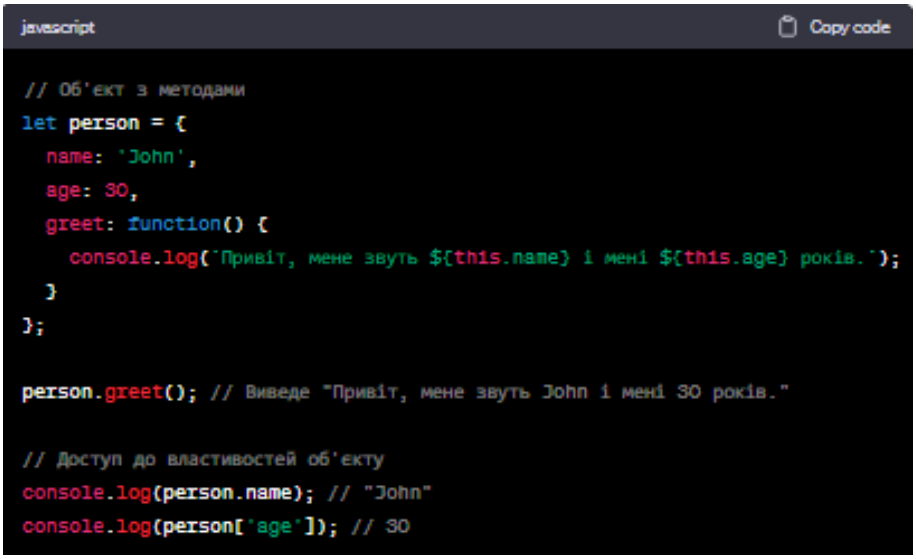



Рис.11 – Об’єкти і методи в JS

**Масиви в JS**

Завдання масивів відбувається за допомогою оператора Array (масиви є об'єктами):

*mas = new Array()*[2];

```javascript
let numbers = [1, 2, 3, 4, 5]; // Масив чисел
let fruits = ['apple', 'banana', 'orange']; // Масив рядків
let mixedArray = [10, 'hello', true, {name: 'John'}, [1, 2, 3]];
```

Рис. 12 – Оголошення масиву

Нумерація елементів масиву починається з нульового, однак при ініціалізації в дужках вказується розмірність масиву, тобто число елементів, під які виділяється пам'ять. При цьому розмірність і тип елементів масиву може мінятися в процесі виконання програми, тобто в мові JavaScript не буде помилки «вихід за межі масиву». При звертанні до неіснуючих елементів (раніше не заданим), їх вміст буде приймати значення "undefined".

*mas2 = new Array("A", "B", "C")*[3]

Звертання до елементів масиву відбувається шляхом вказівки ім'я масиву й індексу елемента. При цьому вказівка індексу елемента більшого, ніж зазначено при оголошенні масиву веде до збільшення розмірності масиву, а не до помилки.

```javascript
let numbers = [1, 2, 3, 4, 5];

console.log(numbers[0]); // Виведе 1
console.log(numbers[2]); // Виведе 3
```

Рис. 12 – Отримання значення за допомогою індексів

```javascript
let numbers = [1, 2, 3, 4, 5];

console.log(numbers.length); // Виведе 5 - кількість елементів у масиві
```

Рис.13 – Визначення довжини масиву

[2] заданий масив mas, що має невизначену розмірність, якщо у дужках поставити число, то воно буде визначати розмір масиву

[3] Масив із трьох елементів

**Операції з масивами**

```javascript
let fruits = ['apple', 'banana', 'orange'];

fruits[1] = 'grape'; // Змінить значення другого елементу на 'grape'
console.log(fruits); // Виведе ['apple', 'grape', 'orange']
```

Рис. 14 – Зміна елементу масиву за індексом

```javascript
let fruits = ['apple', 'banana', 'orange'];

fruits.push('kiwi'); // Додає елемент 'kiwi' в кінець масиву
console.log(fruits); // Виведе ['apple', 'banana', 'orange', 'kiwi']

fruits.pop(); // Видаляє останній елемент масиву
console.log(fruits); // Виведе ['apple', 'banana', 'orange']
```

Рис. 15 - Додавання та видалення елементів в масиві

```javascript
let numbers = [1, 2, 3, 4, 5];

numbers.slice(1, 3); // Вирізає підмасив з індексу 1 до 2 (не включно)
// Результат: [2, 3]

numbers.splice(1, 2); // Видаляє 2 елементи з початку масиву з індексу 1
// Результат: [1, 4, 5]

numbers.join('-'); // Об'єднує елементи масиву в рядок з роздільником '-'
// Результат: "1-4-5"

numbers.reverse(); // Перевертає порядок елементів у масиві
// Результат: [5, 4, 1]
```

Рис. 16 – Операції з масивами

**Виведення в JavaScript**

У JavaScript виведення даних на екран може відбуватися різними способами.

Найпоширенішими способами виведення є використання функції ***console.log()*** для виведення в *консоль* та використання методу ***alert()*** для виведення повідомлення у *спливаючому вікні* на сторінці. При цьому оператори

виводу оптимізовані для найбільш зручного їх використання. Найбільш простим є застосування оператора ***Alert()***. Аргументом оператора може бути будь-який рядковий вираз.

```javascript
let message = 'Hello, world!';
console.log(message); // Виведе 'Hello, world!' у консоль
```

Рис. 17 – Код для виведення в консоль

```javascript
let message = 'Це повідомлення на сторінці';
alert(message); // Створить спливаюче вікно з текстом 'Це повідомлення на сторін
```

Рис.18 – Код для виведення у спливаючому вікні на сторінці

Виведення в HTML (рис.19) при цьому відбувається зміна вмісту елементів DOM. В даному випадку, JavaScript вибирає елемент з ідентифікатором '***output***' за допомогою ***document.getElementById()*** та змінює його вміст за допомогою ***.innerHTML***.

```html
<!DOCTYPE html>
<html>
<head>
    <title>Виведення в JavaScript</title>
</head>
<body>
    <div id="output"></div>

    <script>
        let message = 'Це текст у div елементі';
        document.getElementById('output').innerHTML = message;
    </script>
</body>
</html>
```

Рис. 19

**Завдання**

Використовуючи раніше розроблені веб-сторінки виконати завдання за допомогою **JavaScript**:

1. На першій сторінці реалізуйте «карусель» із 10-ти зображень та кнопки перемикання. При чому рух зображень має бути реалізовано в двох напрямках (прямому і зворотньому).

2. Реалізуйте щоб при наведені курсору на об’єкт (наприклад, рисунок логотипу, зображення, таблиця тощо) відбувалося збільшення об’єкту при наведенні на нього курсору й повернення від збільшеної копії до зменшеного після зняття курсору (рекомендовано використати оброблювачі подій *OnMouseOver* і *OnMouseOut*).

3. Напишіть скрипт для реалізації багатомовного інтерфейсу (додайте англомовну першу сторінку) і користувач може обирати мову відображення інформації на сторінці.

4. Напишіть скрипт для математичних операцій - калькулятор на вашому сайті згідно тематики сайту. Наприклад, якщо тематика сайту туризм, то реалізувати розрахунок вартості поїздки з урахуванням вартості квитків, проживання, послуг страхування тощо; якщо сайт фотографа – розрахунок вартості фотосесії в залежності від кількості фото та тривалості фотосесії.

5. Реалізуйте скрипт, який випадковим чином виводить на екран (цитата відомої особи, передбачення, анекдот тощо.) інформацію (інформацію потрібно помістити в масив і винести в окремий файл скрипта (*.js)).

6. Додайте до вашого сайту ігровий елемент згідно тематики. Наприклад, тематична вікторина, гра «Шибениця», «Гонки», «Хрестики-нулики», міні-квест.

7. Додайте до сайту блок, який буде відображати час та дату у реальному вимірі.

**ЛАБОРАТОРНА РОБОТА №6**
«Реєстраційна форма для веб-сайту»

*Мета*: Познайомитися із синтаксисом, основними елементами для створення реєстраційної форми на сайті мовами HTML, CSS, JavaScript.
*Знати*: Структуру HTML документа, основи CSS та основні елементи мови JavaScript при створенні реєстраційних форм.
*Вміти*: Створювати реєстраційні форми з використанням мов HTML, CSS, JavaScript.

## Теоретична частина

Сучасні веб-застосунки використовують клієнт-серверну архітектуру, яка за останні десятиліття стала одним із основних способів організації розподілених програмних систем. При такій організації клієнтом виступає веб-браузер, а у якості серверного за стосунку виступає веб-сервер. Задачею веб-браузера є візуалізація HTML-сторінок та мультимедійних об‘єктів, з яких вони складаються. В свою чергу вебсервер виконує роль оброблювача запитів від віддалених веб-клієнтів. Обмін інформацією відбувається із використанням методів HTTP: GET, POST, PUT, HEAD, TRACE, OPTIONS, CONNECT, DELETE.

Веб-форма є одним із способів взаємодії користувача з веб-сайтом. Вона дозволяє користувачам вводити та надсилати дані на сервер для обробки або збереження. Форми можуть містити різні елементи, такі як текстові поля, радіокнопки, прапорці, список вибору, кнопки, тощо. При використанні форм на сайт можна створювати облікові записи електронної пошти, переглядати та купувати товари в інтернет-магазинах, здійснювати фінансові транзакції та багато іншого.

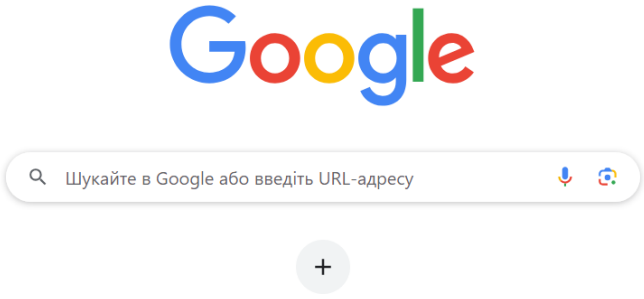
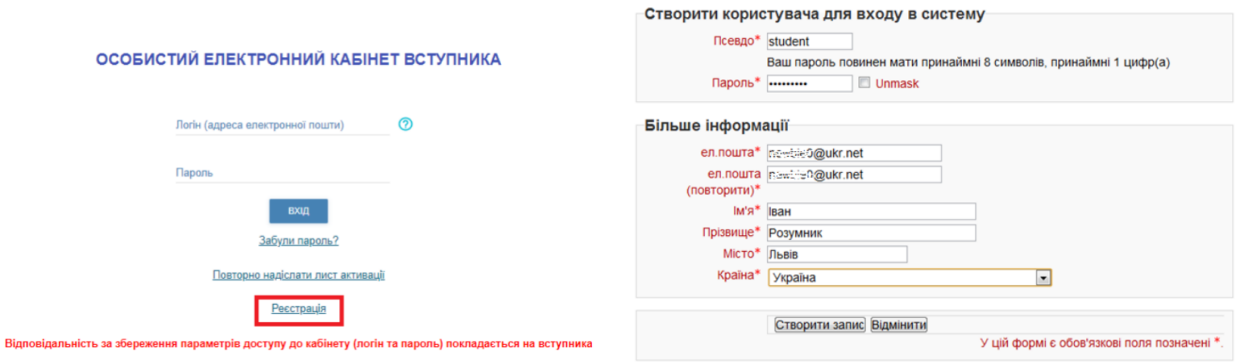



Рис. 20 – Приклад найпростішої форми




а) б)

Рис. 21 – Приклад форм а) і б) на сайті

Таблиця 1 - Основні елементи веб-форми

| Тег | Опис |
|---|---|
| ***<form>*** | Цей контейнер визначає форму на сторінці. В атрибуті action вказується URL або шлях до серверного скрипта, який оброблятиме дані форми. Атрибут method вказує метод відправки даних (наприклад, "post" або "get") |
| ***<label>*** | Використовується для позначення назви елемента форми |
| ***<input>*** | Один з основних елементів форми, що дозволяє вводити текстові дані, вибирати прапорці, радіокнопки тощо. Атрибут type визначає тип елемента |
| ***<select>*** | Використовується для створення списку вибору. Зазвичай використовується разом із <option> для створення пунктів вибору |
| ***<textarea>*** | Використовується для введення багаторядкового тексту |
| ***<button>*** | Використовується для створення кнопки відправки або інших дій |
| ***<fieldset>*** | Поділяється на логічні фрагменти складної реєстраційної форми |

```html
<!DOCTYPE html>
<html>
<head>
    <title>Зразок веб-форми</title>
</head>
<body>
    <form action="/server-script" method="post">
        <!-- Елементи форми будуть розміщені тут -->
        <label for="name">Ім'я:</label>
        <input type="text" id="name" name="name" required>

        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required>

        <label for="gender">Стать:</label>
        <input type="radio" id="male" name="gender" value="male">
        <label for="male">Чоловік</label>
        <input type="radio" id="female" name="gender" value="female">
        <label for="female">Жінка</label>

        <label for="interests">Інтереси:</label>
        <select id="interests" name="interests" multiple>
            <option value="programming">Програмування</option>
            <option value="design">Дизайн</option>
            <option value="music">Музика</option>
        </select>

        <input type="submit" value="Надіслати">
    </form>
</body>
</html>
```

Рис.22 – Скрін коду форми

При відправці форми, дані будуть надіслані на сервер, і на сервері можна обробити ці дані та виконати відповідні дії.

Варто відзначити, що реальній обробці даних форми на стороні сервера може використовуватися мова програмування, така як PHP, Python, Node.js або інші, в залежності від ваших вподобань і технічних вимог.

Всі об'єкти форми, незалежно від того, яким чином здійснюється доступ до них, мають свої набори властивостей, як і документ, вікно. Властивості об'єкта form, зокрема, що випливають:

- action. (URL. По суті, такий, як атрибут action елемента <form>);
- method. (такий, як атрибут method елемента <form>);
- name. (ім'я форми (такий, як атрибут name елемента <form>));

 length. (число елементів input, textarea й select у формі);
 target. (цільове вікно або фрейм);
 elements. (масив, у якому містяться всі елементи input, textarea і select).

Для більшої наочності та зменшення кількості помилок при заповненні форми доцільно застосувати HTML5-атрибут ***autofocus***, який можна вставити в елемент <input> або <textarea> (але тільки один елемент форми), наприклад:

*<input id="name" placeholder="Іван Іванченко" autofocus>*

Для того, щоб звернути увагу користувача на цю вимогу за допомогою якихось своїх візуальних ознак можна, наприклад, виділити рамку поля кольором і поставити біля нього зірочку. Або використання регулярних виразів ***Регулярний вираз*** — це шаблон для зіставлення із зразком, закодований згідно з певними синтаксичними правилами. Вони використовуються для пошуку в тексті рядків, які відповідають певному шаблону. Наприклад, за допомогою регулярного виразу можна перевірити, що поштовий індекс містить правильну кількість цифр, або в адресі електронної пошти є знак @, а його доменне розширення містить, принаймні, два символи.

**Завдання**

1. Реалізувати на головній сторінці форму, яка приймає анкетні дані користувача з використанням різних типів полів для вводу даних (текст, телефон, календар, email тощо); різні варіанти вибору інформації в формі SELECT, RADIO, CHECKBOX. Написати JS-сценарій, який виведе у діалоговому вікні пояснювальну інформацію щодо введення даних до форми.

2. Всі поля обов'язкові для заповнення. Здійснити попередню перевірку коректності даних (наприклад, якщо це поле для email то введена інформація обов’язково повинна мати @, а поле для телефону при введені літер має видавати помилку), які вносяться користувачем у форму у вигляді окремого JS- сценарію, для цього слід застосувати механізм регулярних виразів RegExp.

3. До розробленої форми застосувати елементи стилю CSS для надання формі естетичного вигляду. Наприклад, колір фону, скруглення кнопок «Відправки»/ «Скидання»/ «Видалення», додати ефект зміни кольору при наведенні курсору на кнопку чи текстове поле тощо.

4. Перевірити правильність даних потрібно також на стороні сервера. У випадку помилки знову відображається форма з відповідним повідомленням про помилку. Ті поля, які були заповнені вірно повинні зайняти своє значення.

**ЛАБОРАТОРНА РОБОТА №7**

«Фронтенд та бекенд фреймворки (*Angular, JQuery, React*)»

*Мета*: Познайомитися із фреймворками для фронтенд і бекенд розробки.
*Знати*: Види фреймворків, їх особливості та сфери застосування.
*Вміти*: Використовувати фреймворки при виконанні проєктів.

## Теоретична частина

Фронтенд та бекенд фреймворки - це програмні засоби, які допомагають розробникам побудувати та розвивати веб-додатки. Кожен з них виконує різні функції та забезпечує різні аспекти веб-додатків. Давайте розглянемо їх ближче:

Фронтенд фреймворки призначені для розробки клієнтської частини веб-додатків, тобто того, що відображається та інтерактивно взаємодіє з користувачем в його браузері. Ці фреймворки допомагають розробникам побудувати інтерфейс користувача з HTML, CSS та JavaScript, а також забезпечують інструменти для ефективної взаємодії з сервером.

До популярних фронтенд фреймворків відносять:

***React***. Розроблений компанією Facebook, React є одним із найпопулярніших та потужних фреймворків для створення інтерфейсів користувача.

***Angular***. Розроблений компанією Google, Angular також надає багатий набір інструментів для розробки веб-додатків.

***Vue.js***. Ще один популярний фреймворк, який має легку вагу та простий у використанні.

Бекенд фреймворки призначені для розробки серверної частини веб-додатків, яка обробляє запити від клієнтської сторони, виконує логіку додатку та звертається до бази даних. Бекенд фреймворки допомагають забезпечити безпеку, масштабованість та ефективність серверної частини додатку.

Наприклад:

***Node.js.*** Node.js є середовищем виконання JavaScript на стороні сервера та дозволяє розробникам будувати ефективні та масштабовані серверні додатки.

***Django***. Фреймворк, написаний на Python, який спрощує розробку серверної частини додатків та забезпечує велику кількість вбудованих функцій.

***Ruby on Rails***. Це фреймворк для розробки веб-додатків на мові Ruby, який підтримує швидку розробку та конвенції над конфігурацією.

Обидва типи фреймворків, фронтенд та бекенд, можуть працювати разом для створення повноцінних веб-додатків, де фронтенд відповідає за інтерфейс користувача, а бекенд забезпечує обробку даних та бізнес-логіку.

Бібліотека jQuery Validate - це плагін для jQuery, який дозволяє легко валідувати форми на стороні клієнта, тобто перевіряти правильність введених користувачем даних, перш ніж вони будуть надіслані на сервер. Вона надає зручний спосіб валідації форм, дефініюючи правила та повідомлення про помилки.

Щоб почати використовувати бібліотеку jQuery Validate, спочатку необхідно підключити саму бібліотеку jQuery і після цього підключити бібліотеку jQuery Validate. Всі файли можна завантажити з офіційного сайту бібліотеки або використовувати версію, що розміщена на CDN (Content Delivery Network). Нижче наведений приклад коду для використання jQuery Validate з підключенням через CDN.

```
<!DOCTYPE html>
<html>
<head>
  <title>Приклад валідації форми з jQuery Validate</title>
  <!-- Підключення бібліотеки jQuery -->
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <!-- Підключення бібліотеки jQuery Validate -->
  <script
src="https://cdn.jsdelivr.net/jquery.validation/1.19.3/jquery.validate.min.js"></script>
</head>
<body>
  <form id="myForm">
    <label for="name">Ім'я:</label>
    <input type="text" id="name" name="name" required>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>

    <input type="submit" value="Надіслати">
  </form>

  <script>
    // Виконання скрипту після завантаження сторінки
    $(document).ready(function() {
      // Валідація форми
      $('#myForm').validate({
        rules: {
          name: 'required', // Ім'я має бути обов'язково заповнене
```

```
            email: {
                required: true, // Email має бути обов'язково заповнений
                email: true // Перевірка правильності формату email
            }
        },
        messages: {
            name: 'Будь ласка, введіть своє ім\'я',
            email: {
                required: 'Будь ласка, введіть свій email',
                email: 'Будь ласка, введіть коректний email'
            }
        },
        submitHandler: function(form) {
            // Виконується, якщо форма пройшла валідацію
            form.submit(); // Відправка форми
        }
    });
});
</script>
</body>
</html>
```

У даному прикладі ми створили форму з полями «Ім'я» та «Email» і використали бібліотеку jQuery Validate для валідації. У визначенні методу validate() ми задали правила для кожного поля форми (rules) та повідомлення про помилки (messages). Коли користувач намагається надіслати форму, бібліотека перевіряє правильність введених даних і відображає повідомлення про помилки, якщо вони є. Якщо форма пройшла валідацію успішно, виконується метод submitHandler, який відправляє форму на сервер.

**Завдання**

1) Реалізувати перевірку коректності даних, які вносяться користувачем у форму із використанням бібліотеки jQuery validate.

2) Застосувати один із фронтенд фрейморків для раніше створеного елементу на веб-сторінці, наприклад, слайдер, калькулятор, анімацію.

## КОНТРОЛЬНІ ЗАПИТАННЯ

1) Основи HTML (синтаксис, структура документа).
2) Основні теги HTML, робота з текстом, списки.
3) Створення посилань.
4) Зображення та їх властивості в HTML.
5) Створення таблиць, відступи, вирівнювання.
6) Використання форм в HTML-документі.
7) Поняття каскадних таблиць стилів CSS, їх призначення.
8) Типи стилів. Переваги стилів.
9) Способи додавання стилів на сторінку.
10) Базовий синтаксис CSS.
11) Класи у CSS.
12) Ідентифікатори у CSS.
13) Селектори і їх види.
14) Можливості JavaScript, де застосовується.
15) Робота JavaScript з DOM сторінки.
16) JavaScript команди і коментарі.
17) Змінні в JavaScript і операції над ними.
18) Арифметичні оператори в JavaScript.
19) Логічні оператори та умовні конструкції в JavaScript.
20) Вікна оповіщення і підтвердження в JavaScript.
21) JavaScript функції.
22) Локальні і глобальні змінні.
23) Цикли JavaScript.
24) Події та їх обробка.
25) Перевірка форм в JavaScript.
26) Спеціальні символи в JavaScript.
27) Методи об'єктів в JavaScript.
28) Масиви в JavaScript.

## РЕКОМЕНДОВАНІ ДЖЕРЕЛА


1. HTML Living Standard [Електронний ресурс] – Режим доступу: https://html.spec.whatwg.org/multipage.

2. Resources for developers, by developers. MDN Web Docs. Basic sections of a document [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/ru/docs/Learn/HTML/Introduction_to_HTML/Document_and_website_structure.

3 Ресурс https://html5css.ru/html/html_images.php

4. Web-програмування. Частина 1 (frontend) : навч. посіб. / В. В. Босько, Л. В. Константинова, К. М. Марченко, О. С. Улічев ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький : ЦНТУ, 2022. - 208 с.

5 Client-side form validation [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation.

6. Robin Nixon. Learning PHP, MySQL & JavaScript. 6th Ed. 2021, 825p.

7. Dave Crane, Eric Pascarello, Darren James. Ajax in Action. Manning; 1st edition (November 3, 2005) 680 pages.

8. Український веб-довідник [Електронний ресурс] – Режим доступу: https://css.in.ua/html/events.

9. Бородкіна І.Л., Бородкін Р. О. Web-технології та Web-дизайн : застосування мови HTML для створення електронних ресурсів. Видавництво: Ліра До, 2020, 212с.

10. PHP Documentation Group [Електронний ресурс] – Режим доступу: https://www.php.net/manual/en/intro-whatis.php.

11. Перевірка (валідація) полів в формі перед відправкою jQuery validator [Електронний ресурс] – Режим доступу: https://созданиесайта.net/news-new/proverka-validatsiya-poley-v-forme-pered-otpravkoyjquery.html .

12. jQuery Validation Plugin [Електронний ресурс] – Режим доступу: https://jqueryvalidation.org/documentation/ .

13. Web-програмування. Лабораторний практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 125 «Кібербезпека» та 113 «Прикладна математика» / А. Ю. Шелестов, Н. М. Куссуль; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 1047 Кбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 61 с.

**ДОДАТОК А**

Приклад сторінок веб-сайтів

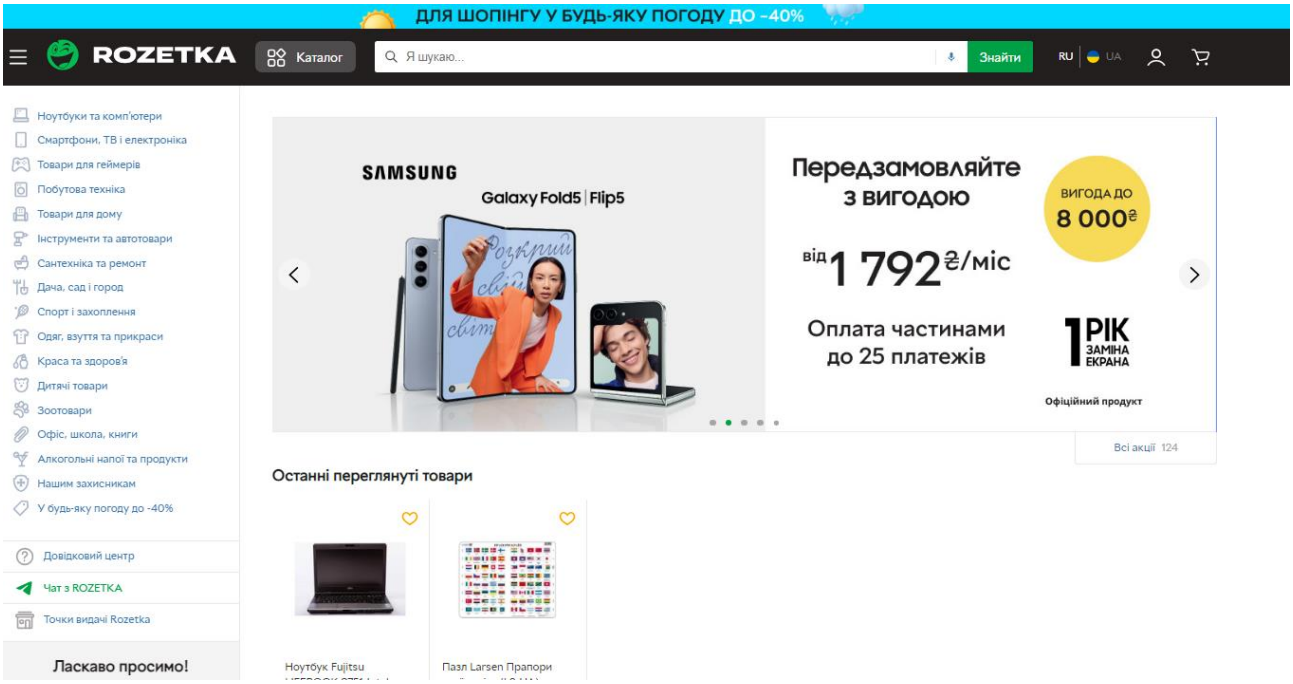



Рис. 1А- Скрін титульної сторінки магазину «Розетка»

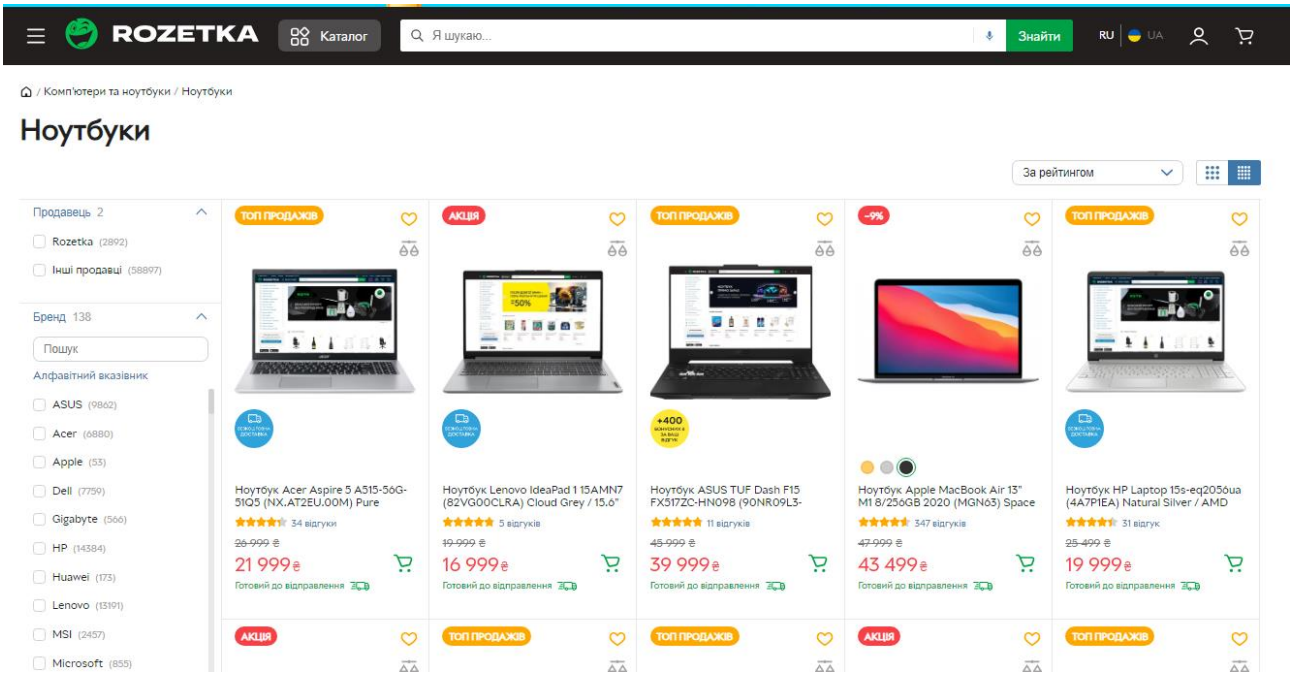



Рис. 2А– Скрін сторінки «Ноутбуки» магазину «Розетка»

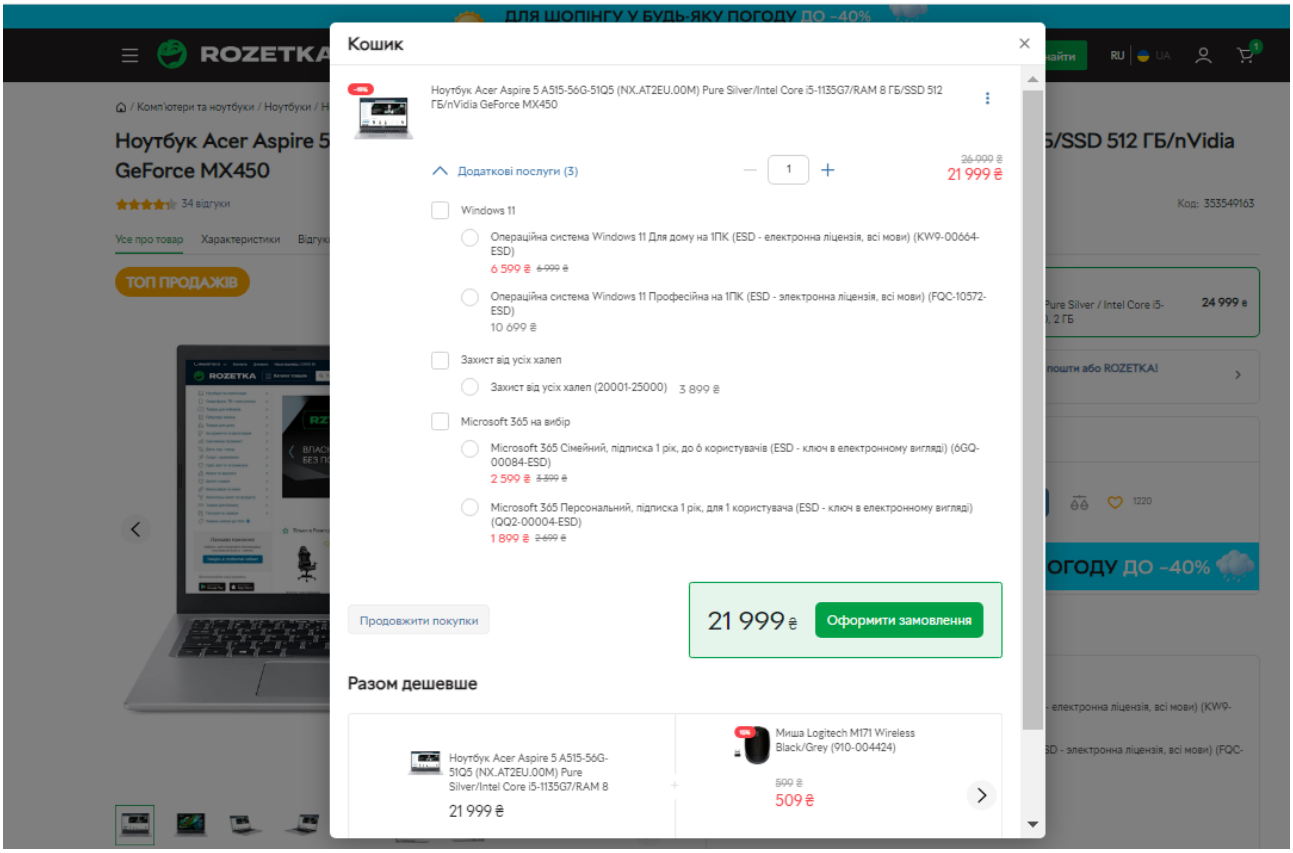



Рис. 3А– Скрін корзини магазину магазину «Розетка»

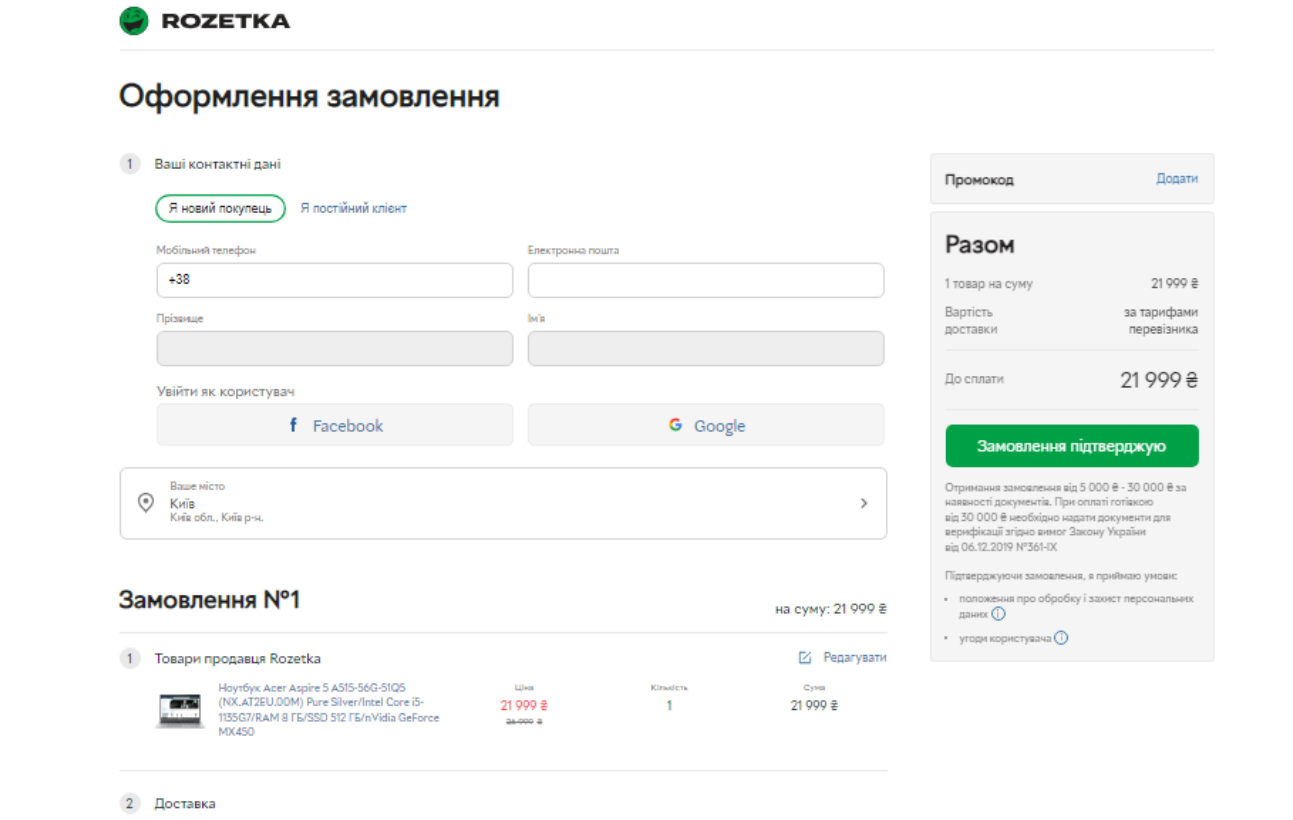



Рис. 4А –Скрін сторінки форми для оформлення замовлення магазину «Розетка»

**ДОДАТОК Б**
**Приклад оформлення звіту з лабораторної роботи**

Міністерство освіти і науки України
Національний технічний університет
«Харківський політехнічний інститут»
Кафедра стратегічного управління

ЗВІТ
ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ №__
з дисципліни «Основи Веб-технологій»

Тема: «_____»

Виконав(ла): студент гр. КН -___
_____ПІБ студента(тки)_____
Перевірив(ла): викладач
_____ПІБ викладача_____

Харків 202_

Навчальне видання

**Методичні вказівки**
для лабораторних робіт
«Основи Веб-технологій»
з дисципліни «Основи Веб-технологій»

для студентів
122 спеціальності – комп’ютерні науки

Укладачі:
ЛИСЕНКО Антон Олександрович
ШУБА Ірина Володимирівна

Відповідальний за випуск (завідувач кафедри) Марина ГРИНЧЕНКО
Роботу рекомендував до друку (експерт РВР) Ігор ГАМАЮН
Комп’ютерна верстка _____
В авторській редакції

План 2023 р., поз. 267

Підп. до друку (дата підпису проректора)_____.
Гарнітура Times New Roman.

Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.
61002, Харків, вул. Кирпичова, 2