

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КИЇВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ
ІМЕНІ ТАРАСА ШЕВЧЕНКА**

Кафедра програмних систем і технологій

ІРИНА ЮРЧУК, ПАВЛО ШАБАТІН, ОЛЕКСАНДР ЛУКАШОВ

РОЗРОБКА ФРОНТЕНДУ ПРОГРАМНИХ СИСТЕМ

Лабораторний практикум для здобувачів спеціальності
F2 (121) «Інженерія програмного забезпечення»
освітнього рівня «Бакалавр»

Київ 2026

УДК 004.738.5 (075.8)

Рецензенти:

доктор філософії за спеціальністю 122 «Комп'ютерні науки», доцент кафедри прикладної математики Державного університету «Київський авіаційний інститут» ВАЛЕРІЙ ЗІВАКІН

к.т.н., доцент кафедри кібербезпеки та захисту інформації факультету інформаційних технологій ОЛЕКСАНДР ТОРОШАНКО

Рекомендовано до публікації кафедрою програмних систем і технологій протокол №10 від «19» січня 2026 року.

Схвалено Вченою радою факультету інформаційних технологій Київського національного університету імені Тараса Шевченка, протокол № 10 від «2» березня 2026 р.

ІРИНА ЮРЧУК, ПАВЛО ШАБАТІН, ОЛЕКСАНДР ЛУКАШОВ

Розробка фронтенду програмних систем [Електронний ресурс]: лабораторний практикум / Ірина Юрчук, Павло Шабатін, Олександр Лукашов. – К.: КНУ імені Тараса Шевченка, 2026. – 105с.

Дане видання містить короткі теоретичні відомості, методичні вказівки та варіанти для виконання лабораторних робіт. Складено відповідно до робочої програми освітнього компоненту «Розробник фронтенду програмних систем».

Для здобувачів спеціальності F2 (121) «Інженерія програмного забезпечення» першого (бакалаврського) рівня вищої освіти.

Публікується в авторській редакції.

Електронна версія цього видання опублікована на сайті кафедри програмних систем і технологій за адресою: <https://pst.knu.ua/library>.

ЗАГАЛЬНІ МЕТОДИЧНІ ВКАЗІВКИ

Освітній компонент «Розробник фронтенду програмних систем» входить до плану підготовки здобувачів першого (бакалаврського) рівня вищої освіти спеціальності F2 (121) «Інженерія програмного забезпечення» та є вибіркоvim у підготовці фахівців даного напрямку. Завдання вивчення освітнього компоненту полягає у формуванні у здобувачів знань про основні підходи, методи та програмні технології розробки фронтенду програмних систем. Освітній компонент спрямований на досягнення компетентностей щодо здатності обирати технології розробки фронтенду програмних систем, виходячи із практичної задачі.

Основна увага приділяється темам, що стосуються розмітки засобами HTML, стилізації за допомогою CSS, розробці клієнтських скриптів мовою JS та засвоєнню базових навичок у роботі із сучасними бібліотеками та фреймворками REACT, VUE.JS, NEXT.JS та ANGULAR.

Кожна лабораторна робота виконується здобувачем під керівництвом викладача та включає наступні етапи:

- вивчення відповідного лекційного матеріалу та теоретичних відомостей до лабораторних робіт;
- виконання завдання на комп'ютері з використанням відповідних програмних засобів та аналіз отриманих результатів;
- оформлення звіту до виконаної роботи, який містить наступні складові: титульний аркуш (назви ЗВО, факультету, кафедри, освітньої компоненти, номер лабораторної роботи, тему роботи, варіант, ім'я та прізвище здобувача, номер групи та підгрупи, рік); мету роботи; сформулюванні завдання та, відповідно до них, скріншоти підтвердження реалізованих елементів фронтенду; посилання на реалізований код (git, jsfiddle, тощо) та висновки, у яких сформульовано набуті в ході виконання роботи вміння та навички.

Для виконання лабораторних робіт, здобувач обирає варіант завдання згідно списку його підгрупи та погоджує його з лектором. Номер варіанту співпадає з порядковим номером здобувача у списку. Якщо у підгрупі кількість здобувачів більша десяти (не більше 18 за нормативними документами ЗВО), то номер варіанту дорівнює цифрі в розряді одиниць порядкового номеру у списку підгрупи.

Оцінювання знань та навиків, яких набув здобувач у ході виконання лабораторної роботи, проводиться викладачем під час захисту роботи.

Зміст

<i>Лабораторна робота 1. Розмітка елементу інтерфейсу засобами HTML</i>	5
<i>Лабораторна робота 2. Стилiзація елементу інтерфейсу засобами CSS</i>	15
<i>Лабораторна робота 3. Основи, функціональне та об'єктно-орієнтоване програмування мовою JavaScript.....</i>	32
<i>Лабораторна робота 4. Реалізація клієнтських скриптів за допомогою JavaScript</i>	47
<i>Лабораторна робота 5. Розробка односторінкових застосунків (SPA) засобами бібліотеки REACT.....</i>	53
<i>Лабораторна робота 6. Розробка користувацьких інтерфейсів з використанням фреймворків VUE.JS та NEXT.JS</i>	56
<i>Лабораторна робота 7. Проектування масштабованих веб-застосунків на платформі ANGULAR</i>	62
<i>Лабораторна робота 8. Адаптивність, кросбраузерність та метрики ефективності фронтенду.....</i>	66
<i>Додатки.....</i>	74
<i>Список літератури</i>	105

Лабораторна робота 1

РОЗМІТКА ЕЛЕМЕНТУ ІНТЕРФЕЙСУ ЗАСОБАМИ HTML

Мета – вивчити основні теги та їх атрибути для розмітки елементу інтерфейсу мовою HTML, набути навичок та вмінь у практичній реалізації фронтенду програмних систем.

Кількість годин – 6

Теоретичні відомості

1. *Вступ до HTML:*

HTML (HyperText Markup Language) — це стандартизована мова розмітки документів для перегляду вебсторінок у браузері, за допомогою якої визначається його структура та вміст. Елемент є одним із основних компонентів розмітки в HTML та має дві основні властивості: атрибути та вміст (контент). Основою HTML є **тег**, який інтерпретується програмним забезпеченням (браузерами) та використовуються для створення структури HTML документа, форматування, створення елементів, тощо. Тег складається із спеціальних символів: "<", ">" та "/". Його основне завдання полягає у позначенні початку і кінця елементів.

Виділяють наступні етапи створення HTML-елемента:

- *створення початкового тега*: напочатку використовується символ "<", потім вказується назва (список символів без пробілу), далі через пробіл атрибути і завершується символом ">";
- *створення контенту елемента*: елемент зазвичай містить контент — дані, які додають після початкового тегу. Якщо його не додавати, то елемент буде порожнім;
- *створення кінцевого тега*: починається зі символів "</", далі додається та сама назва, яку має початковий тег, і завершується символом ">".

Розглянемо приклад створення текстового абзацу за допомогою тегів <p> та </p>, які визначають блоковий елемент, що починається з нового рядка:

<p> У 1991 році британським ученим Тімом Бернерсом-Лі, співробітником Європейського інституту фізики частинок (CERN) у Женеві, було розроблено систему передачі гіпертекстової інформації через інтернет. </p>

Зауважимо, що в одиночних тегів немає контенту і кінцевого тега, тому спочатку пишуть символ "<", далі назву, атрибути та символ ">". Приклади одиночних тегів: ,
.

HTML- елементи до п'ятої версії мови ділилися тільки на блокові елементи (block-level elements) та строкові елементи (inline-level elements). У HTML5 з'явилося поняття контентної моделі.

2. *Контентні моделі. Синтаксис. Огляд основних тегів:*

Flow content — контент-потік, до якого належать більшість елементів, які використовуються в основній частині документів або веб-додатків. Наприклад: <article>, <aside>, тощо.

Interactive content - інтерактивний контент, який передбачає взаємодію з користувачем. Наприклад: <label>, <select>, <textarea>, <video> (якщо є атрибут controls).

Metadata content — контент метаданих. До нього належить контент, що встановлює зображення або поведінку іншої частини контенту або визначає взаємозв'язок HTML-документа з документами (наприклад, з таблицями в стилі CSS), а також відповідає за зовнішній вигляд або поведінку контенту на сторінці. Наприклад: <link>, <meta> та ін.

Phrasing content — текстовий контент. Він представлений елементами для розмітки тексту всередині абзаців. Наприклад: , <bdi>, <bdo>,
, <button> тощо.

Embedded content — вбудований контент, який імпортується в документ з інших джерел, іншої мови розмітки або іншого простору імен. Наприклад: <audio>, <iframe>, та ін.

Heading content — контент, що містить заголовки. Визначає заголовки розділів. Наприклад: <h1>, <h2>, <h3>.

Sectioning content — контент, який являє собою розділи сторінки. Визначається ділянкою заголовків та колонтитулів. Наприклад: <article>, <aside>, <nav>, <section>.

Наведемо Таблицю 1, де надано огляд основних тегів, їх семантики та властивостей.

Таблиця 1. Основні теги та їх опис

Ім'я тегу	Опис
<html>...</html>	кореневий елемент HTML-документу
<head>...</head>	контейнер для колекції метаданих HTML-документу, які дають інформацію браузеру про заголовок сторінки (title), кодування символів, підключені стилі та інше
<link>	встановлює зв'язок даного документа із зовнішнім, наприклад, з файлом зі стилями або шрифтами та завжди повинен знаходитись усередині <head>...</head>
<title>...</title>	заголовок документа, який не відображається безпосередньо на веб-сторінці, а на вкладці у браузері
<meta>	містить дані (метатеги), які потрібні для зберігання інформації, необхідної для браузерів та пошукових систем

Ім'я тегу	Опис
<script>...</script>	використовується для вбудовування клієнтського скрипта (JavaScript). Або містить оператори скриптів, або вказує на зовнішній файл скрипта через атрибут src
<style>...</style>	містить інформацію про стиль (CSS) для документа
<body>...</body>	необхідний для зберігання вмісту веб-сторінки, який відображається у вікні браузера
<header>...</header>	призначений для визначення заголовного блоку або "шапки" документа чи розділу
<nav>...</nav>	розділ сторінки, який посилається на інші сторінки або їх частини
<main>...</main>	призначений для основного контенту, що належить до головної теми документа
<aside>...</aside>	являє собою розділ сторінки, що складається з контенту, який опосередковано пов'язаний з контентом
<article>...</article>	визначає незалежний, самодостатній контент
<footer>...</footer>	призначений для визначення нижньої частини, "підвалу" документа або розділу, де зазвичай розміщують інформацію про автора документа, посилання на умови використання тексту, інформацію про авторські права, контактні дані
<a>...	міститься контент, який відображатиметься на веб-сторінці і є текстом посилання, яке за замовчуванням відображається у браузері з <u>підкресленням</u> , колір шрифту — синій
	додає зображення на веб-сторінку
<picture>...</picture>	надає веб-розробникам більша гнучкість у визначенні ресурсів зображень. Призначений для художнього оформлення в адаптивному дизайні
<video>...</video>	використовується для вбудовування відеоконтенту в документ, такого як відеокліп або інші відеопотоки. Якщо він містить один або кілька тегів <source> з різними джерелами відео. Браузер вибере перше підтримуване джерело
<audio>...</audio>	використовується для вбудовування звукового контенту в документ (музика чи інші аудіопотоки). Може містити один або декілька тегів <source> з різними джерелами аудіо. Браузер вибере перше підтримуване джерело
<iframe>... </iframe>	визначає вбудований фрейм, який використовується для вбудовування іншого документа в поточний HTML-документ.

Ім'я тегу	Опис
...	визначає текстовий абзац. Є блоковим елементом і починається з нового рядка. Абзаци тексту, що йдуть один за одним, поділяються між собою відступом.
	Якщо тега, що закриває, немає, вважається, що кінець абзацу збігається з початком наступного блокового елемента
<figure>...</figure>	визначає автономний контент, такий як ілюстрації, діаграми, фотографії, списки коду тощо. Його вміст пов'язаний з основним потоком, але його положення не залежить від основного потоку
...	використовується як контейнер для елементів HTML, будь-який тип контенту можна помістити в його всередину та не є семантичним тегом. За замовчуванням браузері завжди розміщують розрив рядка перед і після
<button>...</button>	визначає кнопку, на яку можна натискати. Необхідно завжди вказувати атрибут type, щоб повідомити браузерам, який це тип кнопки

3. Атрибути. Глобальні атрибути:

HTML-атрибути повідомляють браузеру, в який спосіб повинен відображатися той чи інший елемент. Атрибути задають елементам додаткові властивості та розширюють можливості. Якщо атрибут має значення, воно береться в лапки. *Назви та значення атрибутів не чутливі до регістру, проте рекомендується писати їх у нижньому регістрі.* Атрибути пишуться усередині тега. Якщо атрибутів декілька, вони перераховуються через пробіл.

Атрибут має бути єдиним в елемента і не повинен повторюватись, але варто дотримуватись одного стилю для зручності читання коду.

Порядок атрибутів елемента **не має значення** і на результат відображення елемента **не впливає**.

У логічних атрибутів немає значень, сама наявність чи відсутність атрибута вже змінює поведінку елемента.

Синтаксис: <тег атрибут > ...</тег>.

Нижче наведено приклад, який реалізує верстку параграфу з емодзі, що є прихованим:

```
<p hidden>Емодзі &#127804; &#128073; &#128120;</p>
```

Універсальні (глобальні) атрибути виділені в окрему групу, тому що застосовуються практично до всіх елементів HTML. Нижче наведено таблицю з найпоширенішими з них та їх значення.

Таблиця 2. Глобальні атрибути та їх опис

Назва атрибуту	Опис
accesskey	визначає комбінацію клавіш для активації/фокусування елемента
class	визначає одну або кілька назв класів для елемента (посилається на клас у таблиці стилів)
contenteditable	визначає чи можна редагувати вміст елемента
data-*	використовується для зберігання користувацьких приватних даних для сторінки або програми
dir	визначає напрямок тексту для вмісту в елементі
draggable	визначає чи можна перетягувати елемент
enterkeyhint	визначає текст клавіші Enter на віртуальній клавіатурі
hidden	вказує, що елемент ще не є або більше не є релевантним
id	визначає унікальний ідентифікатор для елемента
inert	вказує, що браузер повинен ігнорувати цей розділ
inputmode	визначає режим віртуальної клавіатури
lang	визначає мову вмісту елемента
spellcheck	визначає, чи потрібно перевіряти орфографію та граматику елемента
style	визначає вбудований стиль CSS для елемента
tabindex	визначає порядок табуляції елемента
title	визначає додаткову інформацію про елемент
translate	визначає, чи слід перекладати вміст елемента чи ні

Нижче наведено приклад реалізації переліку емодзі, який є перетягуваним та орієнтованим з права на ліво.

```
<p dir="rtl" draggable="true">Емодзі &#127804; &#128073; &#128120;</p>
```

Розглянемо приклад, який реалізує гіперпосилання (на зовнішній ресурс) та внутрішнє посилання (створює перехід на інший розділ поточної веб-сторінки):

```
<a href="https://pst.knu.ua/" target="_blank">Відвідайте сайт рідної кафедри!</a>
<a href="#section2">Відвідайте Розділ 2</a>
```

Спільним для них є атрибут href. Проте, у першому випадку його значення – це URL-адреса, а у другому – якір (ім'я із символом #).

Зауважимо, що атрибут target визначає спосіб, яким представлено вміст після переходу за посиланням і набуває таких значень: `_self` – завантажує документ у поточній вкладці (є значенням за замовчуванням); `_blank` – завантажує документ у новій вкладці; `_parent` – завантажує документ у батьківській вкладці; `_top` – в HTML5 завантажує у вікні найвищого рівня (якщо батька немає, поводить як `_self`).

Деколи у процесі розробки виникає необхідність створювати стилістику (дизайн та оформлення) елементів за допомогою атрибутів. Синтаксис такого підходу наступний:

```
<тег style="властивість:значення;">
```

Розглянемо приклад реалізації абзацу тексту червоного кольору, який вирівняний по центру:

```
<p style="color:red; text-align:center;"> У 1991 році британським ученим Тімом  
Бернерсом-Лі, співробітником Європейського інституту фізики частинок (CERN) у  
Женеві, було розроблено систему передачі гіпертекстової інформації через інтернет.  
</p>
```

Детально стилістичні аспекти розробки фронтенду будуть розглянуті у наступній лабораторній роботі.

4. *HTML-структура. Список. Таблиця. Форма:*

У кожному HTML-документі теги формують дерево HTML-елементів. Перший рядок у кожному веб-документі повинен містити оголошення `<!DOCTYPE html>`, яке інформує веб-браузер про тип і версію HTML, яка використовується під час створення веб-документа.

Елемент `<html>` є кореневим елементом HTML-документу. Елемент `html` містить обов'язково два елементи. Першим елементом є `head`, а іншим — `body`. Елемент `body` завжди йде після `head`.

HTML-списки дозволяють веб-розробникам структурувати набір пов'язаних елементів. Є три типи списків: порядковані, впорядковані та списки означень. Наприклад, для реалізації навігації на сторінці можна використати неупорядкований список.

Неупорядкований список починається з тегу `<ul>`. Кожен елемент списку починається з тегу `<li>`. Елементи списку за замовчуванням будуть позначені маленькими чорними кружечками (для зміни такого маркування необхідно застосувати стилі).

Упорядкований список починається з тегу `<ol>`. Кожен елемент списку починається з тегу `<li>`. За замовчуванням нумерація буде у вигляді арабських цифр.

Нумерація за допомогою арабських цифр — це не єдиний вид маркерів, які є у нумерованого списку. Для зміни маркера можна використовувати атрибут `type`.

HTML також підтримує списки означень, тобто термінів з описом кожного терміна. Тег `<dl>` визначає список означень, тег `<dt>` визначає термін (назву), а тег `<dd>` описує кожен термін.

HTML-таблиці - це елементи, за допомогою яких дані впорядковуються та виводяться на екран. Кожна таблиця має рядки і стовпці, на їх перетині утворюються комірки. У комірках таблиці можуть бути будь-які HTML-елементи,

наприклад: заголовки, списки, текст, зображення, елементи форм і навіть інші таблиці. Детально теги та їх значення наведено у Таблиці 3.

Таблиця 3. Основні теги для формування таблиці та їх опис

Назва тегу	Опис
<td> ... </td>	визначає вміст комірки таблиці
<th> ... </th>	визначає вміст комірки-заголовка таблиці. По замовчуванню вміст відображається напівжирним
<tr> ... </tr>	визначає рядок таблиці
<thead>...</thead>	визначає верхній колонтитул таблиці
<tbody>...</tbody>	визначає основну частину таблиці
<tfoot>...</tfoot>	визначає нижній колонтитул таблиці
<caption>...</caption>	додає підпис таблиці. У разі використання, його необхідно ставити відразу після тегу, що відкриває <table>

Для об'єднання комірок таблиці необхідно використовувати наступні атрибути: `colspan` та `rowspan`. Перший задає кількість комірок у рядку для об'єднання по горизонталі, а другий – по вертикалі. Можливі значення від 1 до 999. Для однієї і тієї ж комірки можуть використовуватись одночасно.

Наведемо приклад верстки таблиці, яка складається з трьох стовпців та п'яти рядків (кількість тегів <tr>), комірка з текстом «Місяць» - це дві об'єднані по вертикалі комірки першого стовпця, аналогічно, комірки з текстами «Кількість» та «22», мають об'єднання по вертикалі:

```
<table>
<caption>Кількість занять</caption>
<thead>
<tr>
<th rowspan="2">Місяць</th>
<th colspan="3">Кількість</th>
</tr>
<tr>
<th>лекції</th>
<th>практики</th>
</tr>
</thead>
<tbody>
<tr>
<td>Вересень</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Жовтень</td>
<td>6</td>
```

```

<td>7</td>
</tr>
</tbody>
<tfoot>
<tr>
<th>Разом</th>
<td colspan="2">22</td>
</tr>
</tfoot>
</table>

```

У сучасних веб-додатках форми використовуються для отримання зворотного зв'язку, збору персональної інформації, способу оплати та доставки товарів, з'ясування переваг користувачів, вибору товарів та послуг, авторизації на сайтах та в багатьох інших випадках.

При додаванні форми на сторінку важливо пам'ятати, що користувач найчастіше неохоче ділиться своїми даними, тому потрібно враховувати кілька простих правил:

- доступність — використовуйте елементи форми, які допоможуть користувачеві легко ввести дані;
- збирайте інформацію у зручному для зберігання та використання вигляді;
- збирайте лише потрібну інформацію;
- підказуйте користувачам, які дані від них очікуються та у якому форматі.

Основні теги, що застосовуються для реалізації форми та її складових, наведено у Таблиці 4.

Таблиця 4. Основні теги для реалізації форми та їх опис

Назва тегу	Опис
<form>...</form>	додає форму на сторінку, яка є контейнером, всередині якого можуть розміщуватись різні її елементи. Кількість форм на одній сторінці не обмежена стандартом
<input>	це основний елемент форми. Є полем типизованих даних, зазвичай з елементом управління форми, що дозволяє користувачеві редагувати дані. Залежно від значення атрибута type елемент набуває вигляду кнопки, поля для вводу, перемикача, прапорця, тощо
<textarea>...</textarea>	використовується для створення багаторядкової області, в яку можна вводити текст. Атрибут cols визначає ширину поля в символах, атрибут rows задає висоту поля в рядках тексту

Назва тегу	Опис
<label>...</label >	додає текст із поясненням для того, щоб користувач розумів, яку інформацію потрібно вводити в елементи форми.
<fieldset>... </fieldset>	використовується для візуального та логічного поєднання елементів форми. Візуально згруповані елементи обводяться рамкою та отримують внутрішні відступи. Рамка розтягується на всю ширину батьківського блоку
<select >...</select>	елемент управління для вибору набору параметрів. Список опцій елемента select складається з усіх дочірніх елементів option елемента select. Ці елементи визначені заздалегідь і користувач не може додавати значення
<button>...</button>	додає кнопку, яка може містити контент всередині (текст, зображення та ін.)

Існує два способи зв'язування елемента та мітки. У деяких випадках, наприклад, у верстці під мобільні пристрої, розміщувати елемент label просто немає де. Тоді використовується атрибут placeholder елемента input, який виводить текст всередині поля форми. Такий текст зникає при отриманні фокусу або коли користувач починає вводити інформацію.

Для реалізації форм важливо знати можливі атрибути та їх значення для елемента input. У Таблиці 5 наведено значення атрибуту type та їх опис.

Таблиця 5. Значення атрибуту type елементу input

Значення атрибуту	Опис
button	кнопка, на яку можна натискати (здебільшого використовується з JavaScript для активації скрипта)
checkbox	прапорець
color	засіб вибору кольору
date	елемент керування датою (рік, місяць, день (без часу))
datetime-local	елемент керування датою та часом (рік, місяць, день, час (без часового поясу))
email	поле для адреси електронної пошти
file	поле вибору файлу та кнопку "Огляд" (для завантаження файлів)
hidden	приховане поле введення
image	зображення як кнопку надсилення
month	елемент керування місяцем та роком (без часового поясу)
number	поле для введення числа

Значення атрибуту	Опис
password	поле пароля
radio	радіокнопка
range	елемент керування діапазоном (наприклад, повзунок)
reset	кнопка скидання
search	текстове поле для введення рядка пошуку
submit	кнопка надсилання
tel	поле для введення номера телефону
text	однорядкове текстове поле. Дане значення є за замовчуванням
time	елемент керування для введення часу (без часового поясу)
url	поле для введення URL-адреси
week	елемент керування тижнем та роком (без часового поясу)

Нижче наведено реалізацію вибору користувачем «ранку», «обіду» та «вечора». Причому опція «ранок» вибирається за замовчуванням, а опція «вечір» є недоступною.

```
<label for="lesson-time">Оберіть зручний час:</label>
<select name=" lesson-time" id=" lesson-time">
  <option value="mornining" selected > ранок </option>
  <option value="afternoon" > обід </option>
  <option value="evening" disabled> вечір</option>
</select>
```

Завдання

1. Вивчити теги, їх атрибути та можливі значення, що пов'язані з семантичними елементами, посиланнями, текстами, зображеннями, списками, таблицями та формами для розмітки сторінки мовою HTML.

2. Реалізувати ті елементи сторінок з поєднанням досвіду у предметній області, яка відповідає варіанту (див. *Додаток 1*), вимогам *Додатку 2*, а також з дотриманням наступних вимог до «Головної сторінки», де обов'язковими є:

- використання тегів <header>, <nav>, <main>, <aside>, <article>, <footer>;
- наявність двох посилань на зовнішні ресурси та двох на складові документу;
- наявність у таблиці назви, верхнього, нижнього колонтитулів та основної частини, а також не менше 6 рядків та 5 стовпців, з трьома або більше комірками, які є об'єднаними як по вертикалі, так і по горизонталі;
- наявність речення, що стилізоване тегами , <i>, , , <u>.

3. Забезпечити відповідність контенту (текст, зображення, аудіо, відео, тощо) кожної зі сторінок тій, предметній області, яка відповідає обраному варіанту.

Контрольні запитання

1. У чому різниця між рядковими і блоковими елементами? Наведіть приклади рядкових та блокових елементів.
2. Чи є тег `div` семантичним? Наведіть приклади не семантичних тегів у HTML5.
3. Як у контент елементу додати символ, якого немає на клавіатурі?
4. У чому полягає основна різниця між додаванням кнопки за допомогою елементу `button` та універсальним `input` з атрибутом `type="button"`?

ЛІТЕРАТУРА: [1–5, 9].

Лабораторна робота 2

СТИЛІЗАЦІЯ ЕЛЕМЕНТУ ІНТЕРФЕЙСУ ЗАСОБАМИ CSS

Мета – вивчити основні селектори, їх властивості та значення для стилізації елементу інтерфейсу засобами CSS, набути навиків та вмінь у практичній реалізації фронтенду програмних систем.

Кількість годин – 6

Теоретичні відомості

1. *Види таблиць стилів. Пріоритетність:*

Зовнішня таблиця стилів (External)— це текстовий файл із розширенням .css (наприклад, styles.css) у якому містяться css стилі (без html розмітки).

Цей файл може бути створений у будь-якому текстовому редакторі чи редакторі коду. За допомогою зовнішньої таблиці стилів можна створювати або редагувати зовнішній вигляд всього сайту, змінюючи тільки один файл. Підключити зовнішню таблицю стилів до веб-сторінки можна за допомогою тегу `<link>`, розташованого всередині розділу `<head></head>`. Правило `@import` для зовнішніх таблиць стилів дозволяє імпортувати зовнішні таблиці стилів, з'єднуючи в одному місці, в одному документі.

Ключове слово `@import` вказує URI-адресу таблиці стилів, яку потрібно підключити. Після URI, через кому можна вказати типи пристроїв, які використовуватимуть зазначені таблиці: `print` — для друкувального пристрою; `screen` - екран монітора; `all` - всі типи, (це значення використовується за замовчанням) та інші.

Внутрішня таблиця стилів (Internal) зазвичай використовується, якщо одна або кілька окремих сторінок HTML мають унікальний стиль. Вбудовується в розділ `<head></head>` HTML-документа та визначається всередині тега `<style></style>`. Внутрішні стилі мають пріоритет над зовнішніми.

Вбудовані стилі можуть використовуватися для застосування унікального стилю для одного елемента. Щоб використовувати вбудований стиль, потрібно застосувати CSS-код безпосередньо всередині тега елемента за допомогою

атрибуту style. Атрибут style може містити одну або кілька властивостей CSS, перерахованих через крапку з комою. При різних значеннях однієї й тієї ж властивості використовується правило з найвищим пріоритетом.

Оголошення !important перевизначає вбудовані стилі та його застосування не особливо вітається багатьма розробниками. В основному його прийнято використовувати, якщо конфлікт стилів не вдається вирішити іншими засобами. Це правило порушує стандартні правила успадкування таблиць стилів. Робота з такою стильовою таблицею (пошук помилок, модернізація) буде більш трудомісткою. Проте, за допомогою цього підходу можна без проблем перекрити стилі сторонніх бібліотек або пріоритизувати властивості, які повинні бути застосовані на веб сторінці.

2. CSS Синтаксис. Селектор:

Селектор — це частина CSS-правила, яка повідомляє браузеру, до якого елемента (або елементів) інтерфейсу (веб-сторінки) буде застосовано стиль.

Оголошення CSS завжди закінчується крапкою з комою, а блоки оголошення беруть у фігурні дужки. Таблиця стилів складається з набору правил, які визначають значення властивостей селекторів. Часто елементи у таблиці стилів мають спільні властивості, тому їх варто поєднувати. Однією з особливостей CSS є можливість використовувати скорочені властивості та значення. Більшість властивостей та значень мають скорочені альтернативи.

При побудові CSS необхідно дотримуватись певного синтаксису:

селектор {властивість_1:значення; ... властивість_n:значення;}

Наведемо приклад стилізації усіх заголовків h1 за допомогою зміни його кольору шрифту на зелений та розміру до 18 px:

```
h1 {color: green; font-size: 18px;}
```

У Таблиці 6 наведено основні селектори, їх описи та приклади запису.

Таблиця 6. Перелік селекторів, їх описи та приклади

Назва селектору	Опис	Приклад
універсальний селектор	метасимвол, що замінює будь-який доступний елемент	* { margin: 0;}
селектор елемента	елемент, до якого буде застосовуватися правило	p {...}
селектор ідентифікатора	використовується для визначення правил стилю одного унікального елемента	#main-header {...}

Назва селектору	Опис	Приклад
селектор класу	встановлюють стилі для одного чи кількох елементів з однаковим ім'ям класу. Ці елементи можуть знаходитись у різних місцях розмітки	.main-heading { ... }
селектор групування	вибирає всі елементи з однаковими визначеннями стилів	h1, h2, p {...}
комбінатор "пробіл"	вибирає елементи, які знаходяться всередині зазначеного елемента незалежно від рівня вкладеності	/*вибираємо всі h2 всередині .main */ .main h2 {...}
дочірній селектор	вибирає дочірні елементи, які йдуть відразу за батьківським	/*вибираємо всі h2 першого рівня вкладеності всередині header */ header > h2 {...}
селектор сусідніх сестринських елементів	вибирає елементи, що знаходяться безпосередньо після зазначеного елемента на тому самому рівні вкладеності	/*вибираємо всі елементи <p>, які розміщені одразу після елемента <h2>*/ h2+p {...}
загальний селектор сестринських елементів	вибирає всі елементи, які знаходяться на цьому ж рівні вкладеності, що і вказаний, і розташовані після зазначеного елемента з тим самим батьком	/*виберемо всі елементи <p>, які є спорідненими для елементів <h2>*/ h2~p {...}
[атрибут]	вибирає всі елементи, що містять вказаний атрибут	[type] {...}
селектор[атрибут]	вибирає всі елементи даного типу, що містять вказаний атрибут	p[class] {...}
[атрибут="значення"]	вибирає всі елементи, що містять зазначений атрибут з конкретним значенням	[class="item"] {...}
[атрибут~="значення"]	вибирає всі елементи, що частково містять дане значення	/*виберемо всі елементи, які містять item у значенні атрибуту class */ [class~="item"] {...}

Назва селектору	Опис	Приклад
[атрибут "значення"]	вибирає всі елементи, список значень атрибуту яких починається із зазначеного слова	/*виберемо всі елементи, значення атрибуту class яких починається з слова item */ [class "item"] {...}
[атрибут^="значення"]	вибирає всі елементи, значення атрибуту яких починається із зазначеного значення	/*виберемо всі елементи, значення атрибуту class яких починається з it */ [class^="it"] {...}
[атрибут*="значення"]	вибирає всі елементи, значення атрибуту яких містить будь-де зазначене слово	/*виберемо всі елементи, значення атрибуту class містить it */ [class*="it"] {...}

Стилі CSS зазвичай застосовуються до видимих у вихідному коді елементів веб-сторінки. Але іноді виникає необхідність створювати стиль для певного стану елемента (наприклад, встановити зовнішній вигляд посилання при наведеному курсорі), а також для елемента, який чітко не позначений в структурі сторінки (наприклад, перший символ в абзаці).

Ці завдання можна вирішити за допомогою псевдокласів та псевдоелементів. Перелік основних псевдокласів та псевдоелементів з описами до них надано у Таблицях 7 та 8, відповідно. Повний список наданий у [5].

Таблиця 7. Перелік основних псевдокласів та їх опис

Псевдоклас	Опис
:hover	задає стиль елемента, коли на нього наводиться курсор
:link	задає стиль посиланням, за якими користувач ще не перейшов
:active	задає стиль посиланням, за якими користувач ще не перейшов
:not()	дозволяє вибрати та стилізувати ті елементи, які не містять селектор, який вказаний у дужках
:first-child	задає стиль першого елемента групи сестринських елементів (мають одного батька)
:last-child	задає стиль останнього елемента групи сестринських елементів
:visited	задає стиль вже відвіданим посиланням
:focus	задає стиль елемента у момент фокусування на ньому

Псевдоклас	Опис
:lang()	використовується для вибору елементів з атрибутом lang із заданим значенням в дужках
:nth-child()	вибирає будь-який елемент, який є n-дочірнім елементом свого батьківського елемента
:nth-last-child()	вибирає будь-який елемент, який є n-дочірнім елементом свого батьківського елемента, рахуючи з кінця
:only-child	вибирає будь-який елемент, який є єдиним дочірнім елементом свого батьківського елемента
:only-of-type	вибирає будь-який елемент, який є єдиним елементом свого батьківського елемента
:root	вибирає кореневий елемент документа
:scope	вибирає елементи, які є опорною точкою або областю видимості для зіставлення селекторів
:state()	вибирає користувацькі елементи, що мають заданий користувацький стан

Таблиця 8. Перелік основних псевдоелементів та їх опис

Псевдоелемент	Опис
::after	використовується разом із властивістю content і дозволяє вивести необхідні дані після вмісту елемента.
::before	використовується разом із властивістю content і дозволяє вивести необхідні дані перед вмістом елемента
::first-line	використовується для зміни стилю першого рядка тексту елемента
::first-letter	використовується для зміни стилю першого символу у тексті елемента
::selection	дозволяє встановити колір і фон для тексту, виділеного користувачем
::marker	вибирає маркери елементів списку
::placeholder	стилізує текст-заповнювач елементів
::spelling-error	стилізує текст, який браузер позначив як неправильно написаний
::view-transition	представляє корінь накладання переходів перегляду, який містить усі переходи перегляду на сторінці
::view-transition-group	представляє одну групу знімків переходу перегляду
::view-transition-image-pair	представляє контейнер для "старого" та "нового" станів перегляду переходу перегляду - до та після переходу

Псевдоелемент	Опис
::view-transition-new	представляє "новий" стан перегляду переходу
::view-transition-old	представляє "старий" стан перегляду переходу

3. Колір. Рамка. Фон. Шрифти. Одиниці вимірювання:

Значення властивості `color` може бути представлено одним із способів подання кольору:

- **ключове слово** (колір задається текстовим значенням, встановити прозорість для нього можна за допомогою `opacity`);
- **числові значення** у форматі RGB у функціональній нотації (`rgb()`) у дужках якого слідує розділений комами список з трьох числових (або цілочисельних або відсоткових) значень. Цілочисленне значення може бути від 0 до 255);
- **шістнадцяткове значення** у форматі RGB (починається зі знака «#», за яким відразу йдуть три або шість шістнадцяткових символів);
- **у форматі відтінок, насиченість, яскравість** (`hsla()`), де у дужках відтінок представлений у вигляді кута колірного, який вимірюється у градусах, насиченість і яскравість представлені у відсотках, а альфа канал представлений в діапазоні від 0 до 1 і визначає прозорість.

Використовуючи можливості CSS, можна встановлювати різноманітні рамки для різних елементів. Розглянута вище властивість `color` впливає на колір рамки, встановлений за замовчуванням. Це означає, що якщо для рамки явно не вказаний якийсь колір, то він буде такий самий, як і колір тексту.

Універсальна (скорочена) властивість `border` дає можливість одночасно встановити товщину, стиль та колір рамки навколо елемента. Для того, щоб встановити рамку лише з деяких сторін елемента, можна використовувати властивості: `border-top`, `border-bottom`, `border-left`, `border-right`.

Для рамки визначено наступні властивості:

- **`border width`** — ширина, яка може бути задана як визначений розмір (в пікселях, `pt`, `cm`, `em` і т.д.), так і за допомогою одного із трьох попередньо визначених значень (`thin`, `medium`, `thick`);
- **`border color`** — колір рамки, який задають назвою (`name`), HEX (наприклад, `#ff0000`), RGB або HSL. Якщо ця властивість не задана, то рамка наслідує колір елемента;
- **`border style`**: може мати від одного до чотирьох значень (для верхньої межі, правої межі, нижньої межі та лівої межі) і допускається крапкова (`dotted`), пунктирна (`dashed`), суцільна (`solid`), подвійна (`double`), тримірна рифлена (`groove`), тримірна ребриста (`ridge`), зовнішня тримірна (`outset`), прихована (`hidden`) межі та її відсутність (`none`);

- `border-radius` дозволяє встановити закруглення для 4-х кутів рамки. Приймає значення у відсотках або числових одиницях. Відсотки розраховуються від ширини блоку. Зазначимо, що властивість `border-radius` є універсальною властивістю для властивостей: `border-top-left-radius`; `border-top-right-radius`; `border-bottom-right-radius`; `border-bottom-left-radius`. Зазначені властивості дозволяють окремо задати закруглення конкретного кута.

Кожен блок має свій фоновий шар. Він буває прозорий (за замовчуванням) чи кольоровий. Як фон може бути використане одне або кілька зображень. Для встановлення фону використовуються властивості `background-color` та `background-image`.

Різні шрифти для заголовків, абзаців та інших елементів використовуються для того, щоб можна було задавати певний стиль тексту, передавати емоції та настрої. Не забувайте, що незважаючи на різноманітність шрифтів, основний текст на веб-сторінці має добре читатися. Не рекомендовано використовувати більше двох шрифтів на одній веб-сторінці. Якщо необхідно отримати контрастний варіант, то краще зробити це за допомогою різної товщини, розміру, зображення або ж використовувати різні кольори шрифтів.

Існують наступні властивості для задання стилістики шрифтів елементу:

- `family-name` — ім'я сімейства шрифтів, наприклад, Helvetica або Verdana;
- `generic-family` — представляє загальний вибір шрифту і поводить себе як псевдонім для одного або кількох локально встановлених шрифтів, що належать до зазначеної категорії універсальних шрифтів;

- `font-weight` вказує вагу гліфів у шрифті, ступінь їх чорноти або товщину обведення;

- `font-style` дозволяє вибирати курсивні чи похилі накреслення шрифту. Може приймати значення `normal`, `italic`, `oblique`, наприклад: `font-style: oblique 45deg`;

- `font-size` вказує бажану висоту гліфів шрифту. Значення можуть бути задані як абсолютно (`xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`, `xxx-large`), так і відносно (`larger`, `smaller`). Також дозволяється використовувати будь-які допустимі одиниці в CSS: `em`, `rem`, `px`, `%`. За 100% береться розмір шрифту батьківського елемента.

Одиниці виміру поділяються на дві групи: абсолютні та відносні.

Одиниці абсолютного виміру є фіксованими, і довжина, виражена в будь-якій із них, буде відображатися саме як фіксований розмір. Наприклад, сантиметри (cm), міліметри (mm), дюйми (in), точки (pt).

Абсолютні одиниці виміру не рекомендуються для використання виведення зображення на екрані, тому що розміри екранів різних пристроїв можуть сильно відрізнятися. Однак їх можна використовувати, якщо відомий вихідний носій, наприклад, для друкованого макета (сторінки для друку).

Одиниці відносного виміру визначають величину відносно іншої властивості виміру. Наприклад, відносно розміру шрифту елемента (em), відносно ширини “0” (ch), відносно розміру шрифту кореневого елемента (rem), відносно 1% ширини області перегляду (vw), відносно 1% висоти області перегляду (vh), відносно 1% меншого розміру області перегляду (vmin), відносно 1% більшого розміру області перегляду (vmax) та відносно батьківського елемента (%).

Відносні одиниці виміру краще масштабуються між різними середовищами рендерингу (виведення на екран).

4. Блокова модель. Позиціонування елементів:

За відображення елементів відповідає властивість `display`. Всі елементи мають значення цієї властивості за замовчуванням, яку можна змінювати. Найчастіше використовують такі значення властивості `display`: `block`, `inline`, `inline-block` та `none`, а також `flex`, `grid` та інші.

Якщо елемент має значення `display: none`, то він буде прихований і сторінка відобразиться так, ніби елемента не існує. Будь-які елементи, вкладені в цей елемент, також будуть приховані.

Кожен елемент на будь-якій веб-сторінці відповідає блоковій моделі, тому розуміння як вона працює та її використання дуже важливе для стилізації веб-документів.

Основні властивості блокової моделі:

- `width` — встановлює ширину елемента, яка за замовчуванням ґрунтується на значенні `display`. У блокових елементів ширина за умовчанням дорівнює 100% батьківського блоку і займає весь доступний горизонтальний простір;
- `height` — встановлює висоту елемента, яка за замовчанням визначається його вмістом. Елемент розширюватиметься і стискатиметься по вертикалі за необхідності, щоб умістити його вміст;
- `padding` — задає внутрішній відступ елемента, який є відстанню між основним вмістом та його межею (рамкою). Якщо елементу, до якого застосовується `padding` встановити фон, то він пошириться також і на поля елемента. Внутрішній відступ не може набувати від’ємних значень;
- `margin` — задає зовнішній відступ елемента, який є відстанню елемента від зовнішньої межі рамки до сусідніх елементів, тим самим розділяючи елементи на сторінці. Зовнішні відступи завжди залишаються прозорими і через них видно фон батьківського елемента. Може набувати від’ємних значень;
- `border` — задає межу або рамку елемента за допомогою трьох значень: ширини, стилю та кольору.

Для розрахунку фактичного розміру блоку елемента потрібно враховувати поля, межі та відступи для всіх чотирьох сторін блоку.

Позиціонування — це положення елемента в декартовій системі координат. Розрізняють п'ять видів позиціонування: нормальне, абсолютне, фіксоване,

відносно та липке. Значення властивості `position` визначає за якими правилами позиціонуватиметься елемент (детально у Таблиці 9). Коли ви використовуєте різні комбінації властивостей `position`, `left`, `top`, `right` і `bottom`, то існує можливість накладати елементи один на одний, виводити в точці з визначеними координатами, фіксувати елементи в зазначеній позиції, визначати положення одного елемента відносно іншого та інше. Керуючи позиціонуванням, можна динамічно змінювати положення елементів без перезавантаження сторінки, створюючи анімацію та різні ефекти.

Таблиця 9. Види позиціонування та їх опис

position:	Опис
static	елемент у нормальному потоці без будь-яких додаткових ефектів. Зазвичай це значення не пишуть. Її додають лише для того, щоб перевизначити інші значення
relative	елемент у нормальному потоці веб-сторінки. Встановлення властивостей <code>top</code> , <code>right</code> , <code>bottom</code> та <code>left</code> призведе до зміщення елемента від його звичайного положення. Інший контент не буде налаштовано, щоб поміститися в будь-який проміжок, залишений елементом
fixed	елемент позиціонується відносно області перегляду, що означає, що він завжди залишається на тому самому місці, навіть якщо сторінку прокручують. Властивості <code>top</code> , <code>right</code> , <code>bottom</code> та <code>left</code> використовуються для встановлення кінцевого розташування елемента. Він не залишає проміжку на сторінці там, де він зазвичай розташовується
absolute	елемент позиціонується відносно найближчого позиціонованого предка (з позицією, відмінною від <code>static</code>). Однак, якщо елемент з абсолютним позиціонуванням не має позиціонованих предків, він використовує тіло документа та переміщується разом із прокручуванням сторінки. Елементи з абсолютним позиціонуванням видаляються зі звичайного потоку документа та можуть перекривати інші елементи
sticky	елемент балансує між відносною та фіксованою позицією, залежно від позиції прокручування. Липкий елемент позиціонується відносно, доки не буде досягнуто певної позиції прокручування, а потім він "застрягає" в цьому місці як <code>position:fixed</code>

Задавати порядок накладання елементів дозволяє властивість `z-index`, яка приймає ціле число. За замовчуванням `z-index` дорівнює `auto`. Ця властивість

працює для елементів, що знаходяться в одному контексті накладання. Контекст накладання - це елементи, які мають спільного батька чи кореневий елемент і разом переміщуються осі Z.

5. Модуль CSS Flex Layout (Flexible Box):

Цей модуль призначений для створення гнучких макетів. Він дає можливість досить просто та гнучко розміщувати елементи у контейнері, розподіляти доступний простір між ними та вирівнювати їх, навіть якщо вони не мають конкретних розмірів. CSS Flexbox дозволяє створити адаптивний дизайн набагато простіше, ніж з використанням властивостей float [5] та position. Адаптивний дизайн дозволяє сторінці підлаштовуватися під екрани різних розмірів для зручного для користувача відображення контенту. Flexbox можна використовувати як для CSS розмітки цілої сторінки, так і її окремих блоків.

Для того, щоб елемент HTML працював як flex-контейнер необхідно властивості display надати значення **flex** або **inline-flex**. Після чого використаний елемент стає **flex контейнером**, а всі його безпосередні дочірні елементи – **flex-елементами**. Спочатку (за замовчуванням) flex-елементи в контейнері займатимуть всю його висоту.

У модулі Flexbox є властивості двох типів:

- **властивості контейнера** (батьківський елемент називають flex контейнер);
- **властивості дочірніх елементів**.

Опис властивостей, їх можливі значення як контейнера, так і дочірніх елементів наведено у Таблицях 10 та 11, відповідно.

Таблиця 10. Перелік властивостей flex-контейнера та їх значень

Властивість	Опис	Можливі значення
flex-direction	встановлює напрямок відображення гнучких елементів	row (default) column row-reverse column-reverse
flex-wrap	визначає, чи повинні гнучкі елементи переноситися чи ні	nowrap (default) wrap wrap-reverse
flex-flow	скорочена властивість для flex-direction та flex-wrap	
justify-content	вирівнює гнучкі елементи, коли вони не використовують весь доступний простір на головній осі (горизонтально)	center flex-start (default) flex-end space-around space-between space-evenly

Властивість	Опис	Можливі значення
align-items	вирівнює гнучкі елементи, коли вони не використовують весь доступний простір на поперечній осі (вертикально)	center flex-start flex-end stretch baseline normal (default)
align-content	вирівнює лінії гнучкості, коли є зайвий простір на поперечній осі, та гнучкі елементи переносяться	center stretch (default) flex-start flex-end space-around space-between space-evenly

Таблиця 11. Перелік властивостей flex-елементів та їх значень

Властивість	Опис	Можливі значення
order	визначає порядок відображення елементів flex усередині контейнера flex	число (0 за замовчуванням)
flex-grow	визначає, наскільки елемент flex зростатиме відносно решти елементів flex	
flex-shrink	визначає, наскільки елемент flex стискатиметься відносно решти елементів flex	
flex-basis	визначає початкову довжину елемента flex	значення у деяких одиницях вимірювання (px, em, тощо)
flex	скорочена властивість для flex-grow, flex-shrink та flex-basis	
align-self	визначає вирівнювання елемента flex усередині контейнера flex. Змінює вирівнювання за замовчуванням, встановлене властивістю align-items контейнера	center flex-start flex-end stretch baseline normal (default)

Зауважимо, що в Таблицях 10 та 11 є порожні комірки у стовпцях можливих значень, проте уважний читач легко сформує їх як поєднання парами значень із відповідних описів властивостей.

6. Модуль Grid Layout:

На відміну від Flexible Box Layout, орієнтованого на одну вісь, Grid Layout оптимізовано для двовимірних макетів: тих, у яких потрібно вирівнювання вмісту в обох вимірах.

Головна особливість у тому, що веб-сторінка поділяється на рядки та стовпці, при цьому в отриманій таким чином сітці можна розміщувати елементи. Управління їх розмірами і розташуванням виконується з допомогою спеціальних властивостей модуля. Grid Layout складається з батьківського елемента з одним або декількома дочірніми елементами.

Для того, щоб елемент HTML працював як контейнер сітки, потрібно встановити для властивості display значення **grid** або **inline-grid**.

Контейнери сітки складаються з елементів сітки, розміщених у стовпцях і рядках.

Властивості модуля Grid Layout розділяють на два класи: властивості контейнеру та елементів. У Таблицях 12 та 13 наведено властивості, їх опис та можливі значення grid-контейнеру та grid-елементів, відповідно.

Таблиця 12. Перелік властивостей grid-контейнеру та їх значень

Властивість	Опис	Можливі значення
align-content	Вертикально вирівнює елементи сітки всередині контейнера	stretch (default value) center space-between space-around space-evenly initial inherit
justify-content	вирівнює елементи сітки по горизонталі всередині контейнера	center space-between space-around space-evenly initial inherit
place-content	скорочена властивість для властивостей align-content та justify-content	
align-items	визначає вирівнювання за замовчуванням для елементів всередині flexbox або grid-контейнера	normal (default) stretch center start end baseline initial

Властивість	Опис	Можливі значення
column-gap	визначає проміжок між стовпцями	length normal (default value - 1em) initial inherit
row-gap	визначає проміжок між рядками сітки	
gap	скорочена властивість для властивостей row-gap та column-gap	
grid-auto-columns	визначає розмір стовпця за замовчуванням	auto (default value) max-content min-content minmax() fit-content() length % initial inherit
grid-auto-flow	визначає, як автоматично розміщені елементи вставляються в сітку	row (default value) column dense row dense column dense
grid-auto-rows	визначає розмір рядка за замовчуванням	auto (default value) fit-content() max-content min-content minmax() length % initial inherit
grid-template-areas	визначає, як відображати стовпці та рядки, використовуючи іменовані елементи сітки	none (default value) itemnames
grid-template-columns	визначає розмір стовпців та кількість стовпців у сітці	none (default value) auto max-content min-content minmax() length % fit-content() repeat() subgrid initial inherit

Властивість	Опис	Можливі значення
grid-template-rows	визначає розмір рядків у сітці	none auto fit-content() max-content min-content minmax() length % repeat() subgrid initial inherit
grid	скорочена властивість для властивостей grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns та grid-auto-flow	
grid-template	скорочена властивість для властивостей grid-template-rows, grid-template-columns та grid-areas	

Таблиця 13. Перелік властивостей grid-елементів та їх значень

Властивість	Опис	Можливі значення
align-self	Вирівнює вміст певного елемента сітки вздовж осі стовпця	auto (default value) stretch center baseline initial inherit
justify-self	вирівнює вміст певного елемента сітки вздовж осі рядка	auto (default value) normal stretch start left center end right overflow-alignment baseline alignment initial inherit
place-self	скорочена властивість для властивостей align-self та justify-self	

Властивість	Опис	Можливі значення
grid-row-end	вказує, де закінчується елемент сітки	auto (default value) span n row-line
grid-row-start	вказує, де починається елемент сітки	auto (default value) row-line
grid-row	скорочена властивість для властивостей grid-row-start та grid-row-end	
grid-column-end	вказує, де закінчується елемент сітки	auto (default value) span n column-line
grid-column-start	вказує, де починається елемент сітки	
grid-column	скорочена властивість для властивостей grid-column-start та grid-column-end	
grid-area	скорочена властивість для властивостей grid-row-start, grid-column-start, grid-row-end та grid-column-end	

Зауважимо, що в Таблицях 12 та 13, для скорочених властивостей значення не вказані, проте, вони формуються як поєднання тих, що відповідають описам.

7. Прийоми урізноманітнення:

7.1 Гرادієнти. За допомогою градієнтів можна відобразити плавні переходи між двома і більше кольорами. Градієнти найчастіше використовуються як фонові зображення. У CSS3 існує два типи градієнтів: лінійні та радіальні.

Для встановлення лінійного градієнта або радіального градієнту необхідно використовувати функції linear-gradient() або radial-gradient() у властивості background або background-image.

При використанні лінійного градієнта можна задавати напрямок або лінію градієнта. Якщо напрямок не задано, то використовується значення за замовчуванням — зверху-вниз. Кольори градієнта за умовчанням розподіляються рівномірно у напрямі, перпендикулярному лінії градієнта.

Для створення градієнта необхідно встановити хоча б два кольори (початковий та кінцевий), між якими буде плавний перехід. Крім цього, ви можете встановити початкову точку та напрямок або кут для градієнта. Напрямок градієнта можна задати двома способами: використовуючи кут нахилу, встановлений у градусах deg, або ключові слова, що відповідають куту градієнта (to top (0deg), to right (90deg), to bottom (180deg), to left (270deg)). Наприклад, якщо потрібно, щоб градієнт переходив від лівої верхньої до правої нижньої частини елемента, використовують ключові слова to right bottom.

Для реалізації радіального градієнту можемо визначати його форму, використовуючи ключові слова `circle` або `ellipse`. За замовчуванням радіальний градієнт набуває форми `ellipse`. Керувати розміром еліпсоподібного градієнта можна, задавши значення `%`, `em` або `px`.

7.2 Переходи (transitions). Вони дають можливість змінити один стиль іншим через певний проміжок часу із заданою швидкістю. Так, наприклад, можемо налаштувати плавну зміну кольору посилання при наведенні курсору. Для реалізації переходу обов'язковими є дві характеристики: `transition-property` і `transition-duration`. Розглянемо їх:

- `transition-property` — задає назву CSS-властивості до якої буде застосовано ефект переходу. Значення за замовчуванням - `"all"`. У цьому випадку перехід буде застосовуватися при зміні будь-якої властивості, яка підтримує перехід. До таких властивостей відносяться CSS-властивості, для яких можна визначити перехід від одного значення до іншого: кольори, розміри тощо;

- `transition-duration` — задає час переходу. Приймає одне або кілька значень за секунди або мілісекунди;

- `transition-timing-function` — задає функцію, яка визначає з якою швидкістю проходить перехід у різні моменти часу. Наприклад, можна встановити швидкий перехід спочатку і повільний в кінці.

- `transition-delay` — встановлює затримку перед початком переходу. Задається в секундах або мілісекундах;

- `transition` — скорочена властивість для `transition-property`, `transition-duration`, `transition-timing-function` та `transition-delay`.

Властивості `transition-property` та `transition-duration` є обов'язковими для завдання переходу.

7.3. 2D та 3D перетворення. За допомогою перетворень можемо повертати, зрушувати, нахилити та масштабувати вибрані елементи, не впливаючи на інші елементи на веб-сторінці. Для цього необхідно використовувати CSS-властивість `transform`.

Перелік можливих 2D-перетворень елемента наступний:

- `translate()` — зсув елемента з його поточної позиції (відповідно до параметрів, заданих для осей X та Y);

- `rotate()` — поворот елемента за годинниковою стрілкою або проти годинникової стрілки відповідно до заданого градуса;

- `scaleX()` — збільшує або зменшує ширину елемента;

- `scaleY()` — збільшує або зменшує висоту елемента;

- `scale()` — збільшує або зменшує розмір елемента (відповідно до заданих параметрів ширини та висоти);

- `skewX()` — зміщує елемент вздовж осі X на заданий кут;

- `skewY()` — зміщує елемент вздовж осі Y на заданий кут;

- `skew()` — зміщує елемент вздовж осей X та Y на задані кути;
- `matrix()` — об'єднує всі функції 2D-перетворення в одну.

Перелік 2D-перетворень елемента детально висвітлено в [5].

7.4. Анімація. Для її створення необхідно використовувати коротку властивість `animation`, яка має властивості `animation-name`, `animation-duration`, `animation-timing-function`, `animation-delay`, `animation-iteration-count`, `animation-direction`, `animation-fill-mode`, `animation-play-state`, або їх поокремо для елемента, який анімується.

Зовнішній вигляд анімації встановлюється за допомогою правила `@keyframes`, всередині якого вказують стилі, у проміжних кроках анімації. Щоб анімація працювала, її необхідно прив'язати до елемента.

Наведемо приклад анімації, що полягає у "тремтячому" ефекті при наведенні курсора на зображення галереї:

```
.gallery img:hover {
  animation-name: shake;
  animation-duration: 3s;
}
```

Детально про значення вказаних вище властивостей для анімації в [5].

Завдання

1. Вивчити селектори, їх властивості та значення, що пов'язані зі стилізацією сторінки мовою CSS;

2. Реалізувати зовнішню та внутрішню таблиці стилів для стилізації сторінок, які розмічені засобами HTML у Лабораторній роботі №1. Стилiзацію виконуємо відповідно до правил одного з усталених стилів веб-дизайну (скевоморфізм, ар-деко, метро, hi-tech, мальований, полігональний, мінімалізм, органічний, гарна типографіка, класичний);

3. Реалізувати селектори (не менше шість різних, як-то селектор елемента, `id`, класу, `[атрибут]`, `[атрибут*='значення']`, тощо), комбінатори (не менше двох різних, як-то `'пробіл'`, `~`, `+`, `>`), два різні шрифти, кольори трьома різними способами задання, як-то `name`, HEX, RGB, HSL, фон, анімацію, 2D трансформацію, тощо, для стилізації сторінок відповідно до вимог, що надані в індивідуальному завданні згідно варіанту (див. *Додаток 3*);

4. Для забезпечення стилістики сторінки «Головна сторінка», необхідно дотримуватись наступних вимог:

- використовувати властивості `display`, `position`, `float` та `z-index`, де властивість `position` має приймати не менше трьох різних можливих значень;

- в одного з елементів сторінки наявний перехід між двома різними стилями (не менше трьох властивостей). Наприклад, при наведенні курсора на

елемент з'являється інший колір фону, тінь від рамки, змінюється шрифт, рамка змінює колір, тощо;

- в одного з елементів сторінки, який містить найбільше текстового контенту наявна смуга прокручування (властивість `overflow`), тип та умови появи якої, здобувач формує самостійно;

5. Стилзація сторінки «Зворотній зв'язок» забезпечується модулем `Flex` із заданням властивостей як `flex-контейнеру` (`flex-flow`, `justify-content`, `align-items`, `align-content` і т.д.), так і `flex-елементів` (`order`, `flex-grow`, `flex-shrink`, `align-self` і т.д.);

6. Стилзація всіх інших сторінок забезпечується модулем `Grid` для забезпечення адаптивності, читабельності та зручності використання.

Контрольні запитання

1. Чи успадковуються властивості, які стилізують фон елементу?
 2. Наведіть приклади властивостей, які можуть набувати від'ємних значень?
 3. Що таке веб-безпечний шрифт? Наведіть приклад веб-безпечного шрифту.
 4. Чому для одного й того ж елементу застосовують як властивість `family-name` так і властивість `generic-family`, задаючи шрифти?
 5. Чи є `Flexbox` одним із способів, який забезпечує кросбраузерність фронтенду?
 6. Які властивості є спільними для `flex` та `grid-контейнерів`?
 7. Які переваги застосування `Grid` для стилзації елементу?
- ЛІТЕРАТУРА:** [1–4, 6, 9].

Лабораторна робота 3

ОСНОВИ, ФУНКЦІОНАЛЬНЕ ТА ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ МОВОЮ JAVASCRIPT

Мета – вивчити типи даних, основні оператори, умовні конструкції, цикли, типи функцій мови програмування JS, набути навиків та вмінь у створенні програмного коду.

Кількість годин – 4

Теоретичні відомості

JavaScript (JS) — це мова програмування, яка з'явилася для додавання динамічності веб-сторінкам. Однак сучасний JS використовується набагато ширше. JavaScript було створено *Брендоном Айком* в 1995 в Netscape Communications. JS є реалізацією стандарту мови програмування *ECMAScript*. JS є скриптовою мовою, що обробляється інтерпретатором окремими інструкціями (командами). Кожна команда JS зазвичай пишеться в окремому рядку і в більшості випадків завершується крапкою з комою.

JavaScript - реєстрозалежна мова програмування, тобто JS розрізняє великі і малі літери. Код JavaScript повинен визначатися за допомогою кодування Unicode.

У кожній мові програмування існують набори домовленостей, що регламентують особливості оформлення коду, порядок використання синтаксичних конструкцій, що допускають неоднозначності, тощо. Це дозволяє зробити код більш зрозумілим. Ці домовленості визначають стиль написання коду. Мова JavaScript не є винятком.

Для оформлення коду мовою JS, використовують наступні правила:

- camelCase для іменування об'єктів, функцій та змінних.
- PascalCase для іменування конструкторів класів.
- підкреслення `_` як префікс для іменування внутрішніх методів та змінних об'єктів

1. Змінні. Константи. Типи даних:

Змінна – це комірка пам'яті (контейнер), яка використовується для зберігання значень даних. Для оголошення змінної використовується ключове слово «let» або «var», основна відмінність між якими полягає у тому, що let створює змінну з обмеженою областю видимості і доступ до неї можна з обмежених фрагментів коду, що підвищує його надійність. Детальніше про відмінності між ними у [14].

Для того, щоб визначити **константу** – треба замість ключового слова let використовувати const. Значення константи не можна змінити. При спробі змінити значення константи отримаєте помилку. Константа має бути обов'язково ініціалізована відразу при її оголошенні.

Типи даних JavaScript можна розділити на дві групи: елементарні (або примітивні) та об'єктні.

Перелік елементарних даних у JS:

- Boolean використовується для зберігання двох логічних значень: «істина» або «хиба». Часто застосовується для умов розгалуження у коді програми;
- Number дозволяє зберігати цілі числа та числа з рухомою крапкою в діапазоні від $-(2^{53} - 1)$ до $(2^{53} - 1)$;
- Infinity, -Infinity та NaN (Not-A-Number);
- BigInt дозволяє працювати з цілими числами довільного розміру. Щоб вказати використання типу BigInt, необхідно в кінці додати літерал «n»;
- Symbol використовується для генерації унікальних значень, які будуть ідентифікаторами змінних інших типів даних (наприклад, об'єктів);
- String використовується для зберігання рядків. Рядки JavaScript повинні бути укладені в одинарні або подвійні лапки;
- Null свідчить, що отримано порожнє значення. Така ситуація може бути коли ми намагаємося отримати об'єкт через API взаємодії з іншими сервісами, але цей об'єкт не існує. Літерал null може також використовуватися для спеціального присвоєння порожнього значення;
- Undefined вказує, що значення змінної не визначено.

Object JavaScript — це набір властивостей та методів. Об'єкти доцільно використовувати для групування даних та роботи з ними. Об'єкти також можуть зберігати методи (функції) для роботи з даними. Отримувати значення властивостей об'єкта можна через крапку або квадратні дужки.

Наведемо приклад для зберігання інформації про користувача (ідентифікатор, ім'я, прізвище та вік) та виведемо у консоль його ідентифікатор, ім'я та прізвище:

```
let id = Symbol("id");
let user = {
  [id]: 1,
  firstName: 'Stepan',
  secondName: 'Kravchuk',
  age: 20
};
console.log(user.firstName);
console.log(user["secondName"]);
console.log(user[id]);
```

2. *Оператори. Динамічна типізація. Неявне перетворення типів:*

Для визначення типу змінної використовується оператор `string typeof` операнд, який повертає тип змінної у вигляді рядка. Зверніть увагу, що для `null` оператор `typeof` поверне тип `"object"`. Ця особливість була пов'язана з тим, що очікувався об'єкт, а отримано значення `null`.

Унарний оператор застосовується до одного операнда. Унарні оператори інкрементів — збільшення змінної на одиницю та декремент — зменшення змінної на одиницю поділяються на префіксні та постфіксні.

Операції інкремент та декремент застосовуються лише до змінних. Ці операції не застосовуються до числових літералів.

Наприклад, оператор унарний мінус `"-"` змінить знак змінної на протилежний.

Бінарний оператор застосовується до двох операндів. Наприклад, оператор мінус може бути як унарним тобто застосовується до одного операнда (робить число негативним), так і бінарним (операція віднімання).

За призначенням оператори поділяються на арифметичні, логічні, оператори порівняння, присвоєння та інші. У Таблиці 14 наведено перелік бінарних операторів, їх опис та приклад коду.

Таблиця 14. Перелік бінарних операторів та їх опис

Оператор	Опис	Приклад коду
+	арифметичний оператор додавання чисел, операція конкатенації, оператор явного перетворення	<pre>console.log (5 + 2); // 7 let result = 'Арифметичні ' + 'оператори'; console.log (result); // "Арифметичні оператори"</pre>

-	арифметичний оператор віднімання і застосовується лише до чисел	console.log (5 - 2); // 3
*	арифметичний оператор, здійснює множення двох операндів і застосовується тільки до чисел	console.log (5 * 2); //10
**	арифметичний оператор піднесення до степеня. Лівий операнд підносимо у степінь правого	console.log (5 ** 2); //25
/	арифметичний оператор, дає результат частки від ділення лівого операнда на правий і застосовується тільки до чисел	console.log(5 / 2); //2.5
%	арифметичний оператор остача від ділення (ділення за модулем) повертає все, що залишилося від цілочисельного ділення лівого операнда на правий і застосовується тільки до чисел	console.log(5 % 2); //1
=	оператор присвоєння. Спочатку обчислюється значення виразу у правій частині (після знака "="), потім це значення присвоюється змінній, яка розміщується у лівій частині	
+=	x+=u рівносильно x=x+u	
-=	x-=u рівносильно x=x-u	
=	x=u рівносильно x=x*u	
/=	x/=u рівносильно x=x/u	
%=	x%=u рівносильно x=x%u	

У JS, як і інших мовах програмування, оператори порівняння повертають булевий тип даних. Тобто результатом завжди буде true чи false. Перелік операторів порівняння надано у Таблиці 15. При порівнянні завжди відбувається перетворення даних до числа. Крім порівняння рядків та суворої рівності (нерівності).

Таблиця 15. Перелік операторів порівняння

Оператор	Приклад	Повертає значення true, якщо
>	a>b	a більше b
>=	a>=b	a більше або дорівнює b
<	a<b	a менше b
<=	a<=b	a менше або дорівнює b
==	a==b	a дорівнює b
===	a===b	a дорівнює b (без приведення типів)
!==	a!==b	a не дорівнює b чи їх типи відрізняються

При порівнянні у JavaScript деякі типи даних не перетворюються до числового типу. Так, значення `undefined` незрівнянно з іншими значеннями:

```
console.log( undefined > 0 ); // false
console.log( undefined < 0 ); // false
console.log( undefined == 0 ); // false
```

При застосуванні оператора порівняння `"=="` до `null` або `undefined`, чисельне перетворення не відбувається, оскільки `null` може дорівнювати лише `null` або `undefined`, і нічому іншому. `NaN` не дорівнює нічому, навіть самому собі. `NaN !== NaN`

У JS існує три логічні оператори: `||` (АБО), `&&` (І) та `!` (НЕ). На відміну від строго типізованих мов програмування, дані оператори можуть застосовуватися до значень будь-яких типів і результат теж може бути будь-яким типом.

Перетворення типів — це процес перетворення значення з одного типу на інший. Будь-який тип даних, будь то примітивний тип або об'єкт, може бути перетворений на інший тип.

Для явного перетворення до булевого значення потрібно застосувати функцію `Boolean()` або подвійне заперечення `!!`.

Неявне перетворення відбувається у логічному контексті `if (val) { ... }` або за умови застосування логічних операторів (`||` та `&&`).

Для явного перетворення до числа потрібно застосувати функцію `Number()`.

Неявне перетворення дещо заплутаніше, оскільки воно викликається у більшій кількості випадків, коли застосовуються:

- оператори порівняння (`>`, `<`, `<=`, `>=`);
- бінарні оператори (`|` та `^`);
- арифметичні оператори (`-`, `+`, `*`, `/`, `%`). Зверніть увагу, що бінарний оператор `+` не викликає чисельного перетворення, якщо один із операндів є рядком;
- унарний оператор `+`;
- оператор нестрогої рівності `==` (включаючи `!=`). Зверніть увагу, що цей оператор не викликає чисельного перетворення, якщо обидва операнди є рядками.

Для явного приведення значення рядка необхідно застосувати до нього функцію `String()`. Неявне перетворення буде викликане бінарним оператором `+`, коли один із операндів є рядком.

3. Умовні конструкції. Цикли:

Набір згрупованих команд називається **блоком коду**. Блоки використовуються для обмеження області видимості змінних та функцій, а також дозволяють звертатися до блоку операторів, як до єдиної інструкції. Блоки можуть бути вкладені один в одного або можуть бути порожніми.

У JS блок коду позначається фігурними дужками `{}`. Блок зазвичай використовується з конструкціями `if`, `for`, `while`, `function` та ін.

Починаючи з ES6, блоки коду обмежують область видимості змінних.

Важливими є умовні конструкції, наведемо їх перелік та синтаксис у Таблиці 16.

Таблиця 16. Перелік умовних конструкторів та їх синтаксис

Назва	Синтаксис	Приклад коду
if – оператор розгалуження	if (умова) { //виконуються оператори, якщо умова істинна }	let weather = 'дощ'; if (weather==='дощ'){ console.log('Взяти парасольку'); }
if-else	if (умова) { // виконуються оператори, якщо умова є істинною } else { //виконуються оператори, якщо умова хибна }	if (a%2===0) { console.log('Число парне'); } else { console.log('Число непарне'); }
тернарний (потрійний) оператор	умова ? вираз1 : вираз2	a%2==0 ? console.log('Число парне') : console.log('Число непарне');
switch	switch (змінна) { case значення1: //оператори, якщо змінна дорівнює значення1 //[break;] case значення2: //оператори, якщо змінна дорівнює значення2 // [break;] ... case значенняN: //оператори, якщо змінна дорівнює значенняN //[break;] default: //оператори, якщо не знайдено відповідного case //[break;] }	let distance = 10; let transport = "літак"; switch(transport){ case 'велосипед' : console.log('Час у дорозі - \${distance/10}'); break; case 'авто' : console.log('Час у дорозі - \${distance/80}'); break; case 'потяг' : console.log('Час у дорозі - \${distance/100}'); break; case 'літак' : console.log('Час у дорозі - \${distance/500}'); break; default: console.log('Про такий транспорт не знаю');} // Час у дорозі - 0.02

В оператор «if–else» допускається вкладати інші оператори «if» або «if–else». Такі ланцюжки використовуються для комбінування else та if, щоб перевірити різні взаємовиключні варіанти.

При створенні ланцюжка операторів if–else–if–else діють *наступні правила*:

- оператор if може не мати або мати лише один else, який стоїть після будь-якого else–if;
- оператор if може не мати або мати багато else–if і вони повинні стояти до його else;
- після того, як будь-який else–if виконаний (тобто результат умови true), то жодний з else–if або else не перевірятимуться.

Синтаксис ланцюжка операторів if–else–if–else наступний:

```
if (умова 1) { //оператори } else if ( умова 2) { //оператори } else if ( умова 3) {  
//оператори } else { //оператори }
```

В JS в операторі if як умову можна передавати будь-який вираз або літеральну константу.

У JS існують наступні оператори циклу:

- for або цикл із лічильником найчастіше використовується тоді, коли заздалегідь відомо, скільки разів потрібно виконати блок коду. Для створення такого циклу використовується оператор for, який складається з трьох блоків у круглих дужках, розділених “;” та тіла циклу, укладеного у фігурні дужки. Синтаксис:

```
for (/*блок ініціалізації*/; /*блок умови виходу*/; /*блок змінної цикла*/;)  
    { /*оператори*/ };
```

- while або цикл з передумовою доцільно в тих випадках, коли потрібно повторювати частину дій, а кількість повторень невідома і залежить від певної умови, наприклад, від даних, які вводить користувач. Синтаксис:

```
while (Умова) { /*оператори*/ };
```

- do-while або цикл з постумовою доцільно використовувати у випадках що й з передумовою, проте його тіло завжди виконається щонайменше один раз. Синтаксис:

```
do { /*оператори*/ } while (умова);
```

Оператори переходу — це оператори, які дозволяють здійснити перехід з точки, де оператор викликається, в іншу частину програми. Оператори break і continue. Основна відмінність break від continue полягає в тому, що break перериває виконання всієї конструкції, а continue лише поточної ітерації.

4. Масиви. Функції. Область видимості:

Масиви призначені для роботи з даними. Для створення масиву застосовується вираз new Array(). Також існує більш короткий спосіб оголошення масиву: let myArray = []. У цьому випадку створюється порожній масив. Внесемо дані в нього:

```
let people = ["Mykola", "Yuriy", "Andriy"];
```

Для звернення до окремих елементів масиву використовуються індекси. Відлік починається з нуля, тобто перший елемент має індекс 0, а останній – 2:

```
console.log(people[0]); // Mykola
```

На відміну від інших мов програмування, в JS масиви не строго типізовані, тобто один масив може зберігати дані різних типів:

```
let obj = ["Andriy ", 44, 10.79, true];
```

Масиви можуть бути одновимірними та багатовимірними. Щоб дізнатись довжину масиву, використовують властивість `length`. Розглянемо приклад двовимірного масиву:

```
var people = [  
  ["Mykola ", 33, false],  
  ["Andriy", 44, true],  
  ["Yuzik", 55, false]  
];  
console.log(people[0]); // ["Mykola ", 33, false]  
console.log(people[1]); // ["Andriy ", 44, true]  
console.log(people[0][1]); // 33
```

У Таблиці 17 наведено основні методи для роботи з масивами та їх короткий опис. Повний перелік у [14].

Таблиця 17. Методи для роботи з масивами та їх опис

Метод	Опис
<code>push()</code>	додає елемент в кінець масиву;
<code>pop()</code>	видаляє останній елемент масиву;
<code>shift()</code>	повертає і видаляє перший елемент масиву;
<code>concat()</code>	повертає новий масив, утворений об'єднанням довільної кількості масивів та окремих значень, які перераховані в методі як аргументи;
<code>join()</code>	об'єднує всі елементи масиву в один рядок;
<code>unshift()</code>	додає новий елемент в початок масиву;
<code>splice()</code>	видаляє всі елементи масиву, починаючи із заданого індексу;
<code>sort()</code>	сортують масив;
<code>every()</code>	перевіряє, чи всі елементи відповідають заданій умові;
<code>reverse()</code>	змінює порядок розташування елементів масиву на обернений;
<code>indexOf()</code>	повертає індекс першого входження шуканого елемента в масиві; в протилежному випадку повертає -1;

Метод	Опис
indexOf()	повертає індекс останнього входження шуканого елемента в масиві; в протилежному випадку повертає -1;
filter()	приймає функцію умови й повертає масив тих елементів, які відповідають вказаній умові;
some()	перевіряє, чи відповідає хоча б один елемент умові, і повертає true; якщо елементів, що відповідають умові, в масиві немає, то метод повертає false;
forEach()	екземпляри виконують надану функцію один раз для кожного елемента масиву
map()	екземпляри створюють новий масив, заповнений результатами виклику наданої функції для кожного елемента у викликаючому масиві

Функція в JS — це тип об'єктів, який за допомогою засобів мови дозволяє формалізувати поведінку та обробку даних. Функції мають такі особливості:

- функція — це підпрограма, призначена для виконання конкретного завдання;
- функція виконується під час виклику;
- у функцію можуть бути передані значення, які використовуються всередині неї;
- функція завжди повертає значення. Якщо значення, що повертається, не вказано, функція поверне undefined.

Функція JS визначається за допомогою ключового слова `function`, після якого йде ім'я та дужки `()`. Назви функцій можуть містити літери, цифри, підкреслення та знаки долара (за тими самими правилами, що й змінні). У дужках можуть бути назви параметрів, розділені комами. У середині функції аргументи (параметри) поведуться як локальні змінні. Код, який виконується функцією, розміщується у `{}`.

Для створення функції, її необхідно оголосити. Оголошення функцій бувають таких видів:

- function declaration: `function name (parameters) {};`
- function expression: `let name = function (parameters) {};`
- стрілкова функція, детальніше у [14]: `() => expression`

Стрілковий функціональний вираз — це компактна альтернатива традиційному функціональному виразу з деякими семантичними відмінностями та навмисними обмеженнями у використанні:

- не мають власних прив'язок до `this`, аргументів або `super`, і не повинні використовуватися як методи;
- не можна використовувати як конструктори. Їх виклик з `new` викликає помилку `TypeError`. Вони також не мають доступу до ключового слова `new.target`;
- не можуть бути створені як функції-генератори.

Зауважимо також, що *кількість параметрів* під час опису функції та виклику може відрізнятись. Якщо під час виклику функції передано менше параметрів, ніж є в описі, непередані параметри дорівнюватимуть `undefined`.

Якщо змінну оголошено всередині функції, до неї не можна отримати доступ зовні. Виходячи з цього, якщо потрібна змінна тільки для функції, то її потрібно оголошувати в області видимості функції (`scope`).

Функція матиме доступ до всіх змінних та функцій, оголошених у зовнішньому scope.

Функція, оголошена у глобальному `scope`, має доступ до всіх змінних у глобальному `scope`. Функція, оголошена всередині іншої функції, ще має доступ до всіх змінних її батьківської функції та інших змінних, до яких ця батьківська функція має доступ.

Зазначимо, що для `Function declaration` функція може бути викликана до опису. Цей механізм отримав назву `hoisting` (піднімання). Його суть полягає в тому, що перед виконанням коду інтерпретатором відбувається певна оптимізація та опис функції піднімається перед її викликом. Якщо бути більш точними, він розміщується на початку області видимості. Функція, описана як `Function expression`, не піднімається.

У деяких випадках функцію необхідно викликати одразу. Для такого завдання використовується наступний синтаксис:

```
(function () { код })();
```

5. Об'єктно-орієнтований `js`-код:

Використовується у випадку, коли потрібно описати сутність «людина», яка має свої характеристики: ім'я, вік, стать і так далі. Тоді «людина» буде виступати як складна комплексна структура, що складається з окремих властивостей: вік, зріст, ім'я, прізвище і т.д.

Для роботи з подібними структурами в `JS` використовуються об'єкти. Кожен об'єкт може зберігати властивості, які описують його стан, і методи, які описують його поведінку. Створення нового об'єкта: `let person = new Object();` або `let person = {}`.

Після створення об'єкта можна визначити в ньому властивості. Щоб визначити властивість, потрібно після імені об'єкта через крапку вказати ім'я властивості і присвоїти їй значення. Також для присвоєння значення властивості використовується символ двокрапки, а після визначення значення ставиться кома (а не крапка з комою).

Методи об'єкта визначають його поведінку, тобто дії, які він виконує або які виконуються з ним. Фактично методи являють собою функції.

Наведемо приклад рядків коду для виводу ім'я і віку людини:

```
let person = {};  
person.name = "Mykola";  
person.age = 16;  
person.display = function() {  
  console.log(person.name);  
  console.log(person.age);  
};
```

```
};  
person.display();// виклик методу
```

Щоб звернутися до властивостей або методів об'єкта всередині цього об'єкта, використовується ключове слово **this**, яке означає посилання на поточний об'єкт.

Існує також альтернативний спосіб визначення властивостей і методів за допомогою синтаксису масивів:

```
let person = {  
  person ["name"] = " Mykola ";  
  person ["age"] = 26;  
  person ["display"] = function() {  
    console.log(person.name);  
    console.log(person.age);  
  };  
  person ["display"](); //виклик методу  
  person.display(); //виклик методу спрацює так само
```

Можна видаляти властивості і методи за допомогою оператора delete. Як і у випадку з їх визначенням, можна видаляти властивості двома способами:

- delete об'єкт.властивість;
- delete об'єкт["властивість"].

Після видалення властивість буде не визначена, тому при спробі звернення до неї, повертається значення undefined.

Оскільки об'єкти відносяться до типу object, а значить, мають всі його методи і властивості, то, відповідно, також можуть використовувати метод `hasOwnProperty()`, який визначений в типі object. Метод повертає true у випадку, якщо об'єкт має не успадковану властивість із зазначеним ім'ям та false, якщо об'єкт не має такої властивості або ця властивість успадкована від свого об'єкта прототипу.

Крім створення нових об'єктів, JavaScript надає можливість створювати нові типи об'єктів за допомогою конструкторів. *Конструктор* – це функція, яка призначена для визначення та задання властивостей та методів. Для звертання до властивостей і методів використовується ключове слово **this**. Оператор `instanceof` () дозволяє перевірити, за допомогою якого конструктора створено об'єкт. Якщо об'єкт створений за допомогою певного конструктора, то оператор повертає true.

Інкапсуляція є одним із ключових понять об'єктно-орієнтованого програмування і полягає в прихованні стану об'єкта від прямого доступу ззовні. За замовчуванням, всі властивості об'єктів публічні, тобто загальнодоступні, і до них можна звернутися з будь-якого місця програми. Для оголошення локальних змінних в конструкторах використовують знак підкреслення у назві.

JS підтримує успадкування. Це дозволяє при створенні нових типів об'єктів за необхідності успадкувати їх функціонал від уже існуючих.

Існує спосіб визначення об'єктів за допомогою класів. Клас являє собою опис об'єкта, його стану і поведінки, а об'єкт є конкретним екземпляром класу.

Для визначення класу використовується ключове слово `class`:

```
class Person { }.
```

Після слова `class` вказується назва класу (в даному випадку, клас називається `Person`), потім у фігурних дужках визначається тіло класу. Можна визначити й анонімний клас і присвоїти його змінній:

```
let Person = class {}.
```

Після цього можна створити об'єкти класу за допомогою конструктора:

```
let andriy = new Person();  
let ivan = new Person();
```

Для створення об'єкта за допомогою конструктора використовується ключове слово `new`, після якого виклик конструктора, фактично, виклик функції по імені класу. За замовчуванням, класи мають один конструктор без параметрів. Викликаючи такий конструктор, в нього не передається жодних параметрів. Також можна визначити в класі свої конструктори, властивості і методи. Конструктор визначається за допомогою методу з іменем `constructor`. По суті, це звичайний метод, який може мати параметри. Основна мета конструктора – ініціалізувати об'єкт початковими даними.

Одні класи можуть успадковуватися від інших. Спадкування дозволяє скоротити обсяг коду в класах-нащадках. Для успадкування одного класу від іншого у визначенні класу застосовується ключове слово **`extends`**, після якого вказується назва базового класу.

Наведемо приклад коду, де клас `Employee` успадковується від класу `Person`:

```
class Person {  
  constructor(n, a) {  
    this.name = n;  
    this.age = a;  
  }  
  print() {  
    console.log(this.name, this.age);  
  }  
}  
class Employee extends Person {  
  constructor(n, a, c) {  
    super(n, a);  
    this.company = c;  
  }  
  print() {  
    super.print();  
    console.log("Працює у", this.company);  
  }  
  work() {  
    console.log(this.name, " є працівитим");  
  }  
}
```

```

}
let andriy = new Person("Andriy", 34);
let ivan = new Employee("Ivan", 36, "Google");
andriy.print(); // Andriy 34
ivan.print(); // Ivan 36 Працює у Google
ivan.work(); // Ivan є працівником

```

Статичні методи викликаються для всього класу в цілому, а не для окремого об'єкту. Для їх визначення застосовується оператор `static`.

6. Об'єкти *Date*, *Math*, *Number* та *String*:

Об'єкт **Date** дозволяє працювати з датами і часом в JS. снують різні способи створення об'єкта *Date*.

Перший спосіб полягає у використанні порожнього конструктора без параметрів:

```
let currentDate = new Date();
```

У цьому випадку, об'єкт буде містити поточну дату комп'ютера.

Другий спосіб полягає у передачі в конструктор *Date* кількості мілісекунд, які пройшли з початку епохи Unix, тобто з 1 січня 1970 року 00:00:00 GMT:

```
let myDate = new Date(1359270000000);
```

Третій спосіб полягає в передачі в конструктор *Date* дня, місяця і року дати:

```
let myDate1 = new Date("27 March 2008");
```

// або так

```
let myDate2 = new Date("3/27/2008");
```

Якщо використовується повна назва місяця, то вона пишеться англійською; якщо використовується скорочений варіант, то формат "місяць/день/рік".

Четвертий спосіб полягає в передачі в конструктор *Date* всіх параметрів дати і часу:

```
// Tue Dec 25 2012 18:30:20 GMT + 0300 (RTZ 2 (зима))
```

```
let myDate = new Date(2012, 11, 25, 18, 30, 20, 10);
```

У цьому випадку, в зазначеному порядку використовуються наступні параметри:

`new Date (рік, місяць, число, час, хвилини, секунди, мілісекунди).`

При цьому потрібно враховувати, що відлік місяців починається з нуля, тобто січень – 0, а грудень – 11.

Для отримання різних складових дати використовуються методи:

- `getDate()` повертає день місяця;
- `getDay()` повертає день тижня (відлік починається з нуля – неділя, останній день – 6 – субота);
- `getMonth()` повертає номер місяця (відлік починається з нуля, тобто місяць з номером 0 – січень);
- `getFullYear()` повертає рік;

- `toString()` повертає повну дату у вигляді рядка;
- `getHours()` повертає годину (від 0 до 23);
- `getMinutes()` повертає хвилини (від 0 до 59);
- `getSeconds()` повертає секунди (від 0 до 59);
- `getMilliseconds()` повертає мілісекунди (від 0 до 999);
- `getTimeString()` повертає повний час у вигляді рядка.

Для задання дати використовують такі методи об'єкта `Date`:

- `setDate()` задає день дати;
- `setMonth()` задає місяць (відлік починається з нуля, тобто місяць з номером 0 - січень);
- `setFullYear()` задає рік;
- `setHours()` задає годину;
- `setMinutes()` задає кількість хвилин;
- `setSeconds()` задає кількість секунд;
- `setMilliseconds()` задає кількість мілісекунд.

Об'єкт **Math** надає набір математичних функцій, які можна використовувати при обчисленнях. Розглянемо основні з них:

- `abs()` повертає абсолютне значення числа;
- `min()` і `max()` повертають, відповідно, мінімальне та максимальне значення з набору чисел (ці функції не обов'язково повинні приймати два числа, в них можна передавати більшу кількість чисел);
- `ceil()` округлює число до найближчого більшого цілого числа (фактично, повертає цілу частину числа +1);
- `floor()` округлює число до найближчого меншого цілого числа (фактично, повертає цілу частину числа);
- `round()` заокруглює число до найближчого меншого цілого числа, якщо десятих менше 5; якщо ж дорівнює або більше 5, то округлення до найближчого більшого цілого числа;
- `random()` повертає випадкове число з плаваючою крапкою з діапазону від 0 до 1;
- `pow()` повертає число, піднесене до степеня;
- `sqrt()` повертає квадратний корінь числа;
- `log()` повертає натуральний (за основою $e=2,718$) логарифм числа;
- тригонометричні функції `sin()`, `cos()`, `tan()` повертають, відповідно, синус, косинус, тангенс кута, що задається параметром у радіанах;
- обернені тригонометричні функції: `asin()` обчислює арксинус числа, `acos()` - арккосинус, а `atan()` - арктангенс.

Крім методів, об'єкт `Math`, також визначає набір вбудованих констант, які можна використовувати в різних обчисленнях, зокрема:

- `Math.PI` (число π): 3.141592653589793;
- `Math.E` (число e або число Ейлера): 2.718281828459045

Об'єкт **Number** представляє число. Щоб створити число, потрібно передати в конструктор `Number` число або рядок, що містить число.

Об'єкт `Number` надає ряд властивостей і методів. Деякі його властивості:

- `Number.MAX_VALUE`: найбільше можливе число, приблизно $1.79E + 308$, числа, які більше цього значення, розглядаються як `Infinity`;
- `Number.MIN_VALUE`: найменше можливе додатне число, приблизно $5e-324$;
- `Number.NaN`: спеціальне значення, яке вказує, що об'єкт не є числом;
- `Number.NEGATIVE_INFINITY`: значення, яке позначає від'ємну нескінченність (`-Infinity`) і яке виникає при переповненні, наприклад, при додаванні двох від'ємних чисел, які за модулем рівні `number.MAX_VALUE`;
- `Number.POSITIVE_INFINITY`: додатня нескінченність (`Infinity`), також, як і від'ємна, виникає при переповненні, тільки тепер в додатному напрямку.

Деякі основні методи об'єкту `Number` :

- `isNaN()` визначає, чи є об'єкт числом, якщо об'єкт не є числом, то повертає значення `true`;
- `parseFloat()` перетворює рядок в число з плаваючою крапкою;
- `toFixed()` залишає в числі з плаваючою точкою певну кількість знаків у дробовій частині.

Для роботи з рядками призначений об'єкт **String** і конструктор `String`:

```
let name = new String("George");
```

Об'єкт `String` має великий набір властивостей і методів, за допомогою яких можна працювати з рядками. Зокрема, властивість `length` повертає довжину рядка.

Метод `repeat()` дозволяє створити рядок шляхом багаторазового повторення іншого рядка, приймає як аргумент кількість повторів, причому повтори не розділяються жодними символами (`console.log(hello.repeat(2));`)

Для пошуку в рядку деякого підрядка використовуються методи `indexOf()` (індекс першого входження підрядка) і `lastIndexOf()` (індекс останнього входження підрядка). Ці методи мають два параметри:

- підрядок, який потрібно знайти;
- необов'язковий параметр, який вказує, з якого символу виконувати пошук підрядка в рядку.

Обидва ці методи повертають індекс символу, з якого в рядок починається підрядок. Якщо підрядок не знайдено, то повертається число «-1».

Ще один метод `includes()` повертає `true`, якщо рядок містить певний підрядок.

Для того, щоб отримати (скопювати) з рядка підрядок, застосовуються методи `substr()` і `substring()`.

Метод `substring()` дозволяє отримати (скопювати) символи з рядка (підрядок) між двома заданими індексами, які передаються як параметр методу, або від певного індексу до кінця рядка, якщо другий параметр не вказано.

Метод `substr()` також дозволяє отримати (скопювати) з рядка певну кількість символів (підрядок), починаючи із заданого індексу. Цей метод приймає як перший параметр початковий індекс підрядка, а як другий – довжину підрядка, який потрібно отримати. Якщо другий параметр не вказано, то копіюється решта рядка.

Для зміни регістру призначені методи `toLowerCase()` (перетворення в нижній регістр) і `toUpperCase()` (перетворення у верхній регістр).

Щоб отримати певний символ в рядку за індексом, застосовують методи `charAt()` і `charCodeAt()`. Обидва методи як параметр приймають індекс символу: але як результат метод `charAt()` повертає сам символ, а метод `charCodeAt()` – числовий код цього символу.

Метод `concat()` об'єднує два рядки. Метод `replace()` замінює перше входження одного підрядка на інший. Метод `split()` розбиває рядок на масив підрядків за певним роздільником. Як роздільник використовується параметр, що передається в метод.

Завдання

1. Вивчити типи даних, основні оператори, умовні конструкції, цикли, типи функцій мови програмування JavaScript та основи об'єктно-орієнтованого підходу в ній;
2. Реалізувати програмно завдання, які наведені у Додатку 4, згідно варіанту здобувача;
3. Провести тестування кожного із програмно реалізованих завдань.

Контрольні запитання

1. Чи однаково створені змінні типу `Symbol` є рівні між собою?
2. Чи може `undefined` використовуватись для присвоєння порожнього значення?
3. Що повертає `Boolean(String)`, `Number(Undefined)` та `String (Null)`?
4. Чи працює піднімання (`hoisting`) для змінних оголошених через `let` і `const`? А для `var`?
5. Чи вірно, що всі змінні які використовуються у функції, мають бути оголошені як локальні?
6. За допомогою чого оголошуються об'єкти/масиви для того, щоб запобігати будь-якій випадковій зміні типу?

ЛІТЕРАТУРА: [1–2, 11-14].

Лабораторна робота 4

РЕАЛІЗАЦІЯ КЛІЄНТСЬКИХ СКРИПТІВ ЗА ДОПОМОГОЮ JS

Мета – вивчити основи мови програмування JavaScript та DOM для практичної реалізації клієнтських скриптів та динамічних ефектів фронтенду програмних систем.

Кількість годин – 6

Теоретичні відомості

1. JavaScript та програмування Web-документів:

JavaScript має декілька важливих аспектів, які характеризують її як спеціалізовану мову для програмування Web-документів:

Аспект DOM. JavaScript надає програмний інтерфейс доступу до усіх складових Web-сторінки: вузлів, елементів, атрибутів тощо. Цей інтерфейс

реалізовано у вигляді вбудованих об'єктів таких, як **Document**, **Event**, **HTML Element** та об'єктів, специфічних для тегів XHTML: **Form**, **Input**, **Body**, **Button**, **Frame**, **Image**, **Anchor** тощо. Таким чином, реалізація DOM дозволяє працювати з вмістом Web-документа з одного боку як з XML-документом, з іншого – як з HTML-документом, враховуючи специфіку тегів та їх атрибутів.

Аспект браузера. Оскільки Web-документ відображається та оброблюється безпосередньо браузером, мова JavaScript має вбудовані об'єкти для роботи зі специфічними для браузера компонентами, зокрема **Window** (вікно документа в цілому), **Navigator** (інформація про поточний браузер), **Screen** (інформація про екран користувача), **History** (дані про вже відвідані сторінки) та **Location** (інформація про поточну адресу URL сторінки).

Ці об'єкти дозволяють більш ефективно пристосувати відображення сторінки для конкретного браузера користувача

Основними є наступні завдання JS у розробці фронтенду програмних систем:

- надання веб-сторінкам можливості взаємодії з користувачем (інтерактивності) шляхом розробки сценаріїв;
- динамічні зміни структури та зовнішнього вигляду веб-сторінок;
- валідації та обробка даних користувача;
- обмін даними з серверними програмами та сервісами;
- створення візуальних ефектів та web анімації;
- розробка мобільних додатків, стаціонарних Desktop-додатків, односторінкових веб-додатків.

Для підключення JS скриптів на веб-сторінку використовується тег `<script>`. Цей тег можна використовувати у двох різних варіантах підключення:

- підключення зовнішнього файлу зі скриптом JS;
- безпосереднє визначення скрипта в середині тега `<script>`

Якщо браузер бачить тег `<script>`, то згідно зі стандартом він має зобов'язання спочатку виконати його, а лише потім показувати частину сторінки, яка залишилася. Тобто всі теги, що йдуть за тегом `<script>` не будуть виконані доти, доки `<script>` не виконається. Така поведінка браузера називається **синхронною**. Для того, щоб не зупиняти відображення сторінки у разі знаходження тега `<script>` можна використовувати атрибут `async` або `defer`.

У разі знаходження `<script async src="">` браузер не зупиняє відображення сторінки, скрипт підвантажується асинхронно і буде виконаний, як тільки завантажиться.

При знаходженні `<script defer src="">` браузер працює асинхронно, подібно до атрибуту `async`, але з двома відмінностями: зберігається порядок виконання скриптів; скрипт з атрибутом `defer` виконається, коли вся веб-сторінка буде оброблена браузером.

JS надає ряд стандартних механізмів взаємодії з користувачем. Сюди відносяться наступні функції створення вікон

- `alert()` - виводить повідомлення,

- `confirm()` - Приймає два аргумента: `result = prompt(title, [default]);` відобразить модальне вікно з текстом, полем для вводу тексту і кнопками ОК/відміна.

- `prompt()` - відображає модальне вікно з текстом питання `question` і двома кнопками: ОК і Відміна.

Модальним називається вікно, яке блокує роботу з веб-сторінкою, доки його не закриють.

Всі ці методи є модальними: зупиняють виконання скриптів і не дозволяють користувачу взаємодіяти з рештою сторінки до тих пір, поки вікно не буде закрито.

На всі вказані методи поширюється два обмеження:

- розміщення вікон визначається браузером. Зазвичай вікна знаходяться в центрі;

- візуальне відображення вікон залежить від браузера і ми не можемо змінити їх вигляд.

2. *Об'єктна модель документа (DOM):*

XHTML DOM – це стандартизований підхід до пошуку та маніпулювання вузлами та елементами Web-документа.

DOM дозволяє виконувати такі задачі, як пошук вузлів за визначеним критерієм, додавати, вилучати та замінювати вузли Web-документа, звертатись до атрибутів та змінювати вміст елементів XHTML, маніпулювати стилями CSS, а також підключати оброблювачі подій. Практично, в браузерах XHTML DOM реалізується у вигляді об'єктно-орієнтованого програмного інтерфейсу (API), де об'єктами є вузли Web-документа, властивостями – їх атрибути, а методи визначають логіку обробки даних вузлів.

HTML DOM - це стандарт того, як отримувати, змінювати, додавати або видаляти HTML елементи.

Вузли у XHTML DOM розділені на наступні типи:

1. Весь документ – це вузол-документ;
2. Кожний елемент XHTML – вузол-елемент;
3. Текстовий вміст елемента XHTML – це текстовий вузол;
4. Атрибут елемента XHTML – це вузол-атрибут;
5. Коментар – це вузол-коментар.

Після завантаження веб-сторінки браузер створює об'єктну модель документа (анг. Document Object Model) цієї сторінки.

Відповідно до типів вузлів надається визначений список властивостей та методів щодо їх обробки. Особливим вузлом є вузол-елемент, стандартні властивості та методи якого варіюються в залежності від його призначення.

Виходячи із деревоподібної моделі, доступними є такі дії, як отримання атрибута (`attributes`) елемента, його батьківського елемента (`parentNode`), списку дочірніх елементів (`childNodes`), попереднього того ж рівня (`previousSibling`), наступного того ж рівня (`nextSibling`) та інші.

У Таблиці 18 наведено перелік методів пошуку елементів HTML та їх опис.

Таблиця 18. Методи пошуку елементів та їх опис

Метод	Опис
getElementById()	якщо елемент буде знайдений, то він буде повернутий у вигляді об'єкта (в змінну myElement). У протилежному випадку, буде містити значення null
getElementsByTagName()	пошук елементів за ім'ям тега
getElementsByName()	пошук всіх HTML елементів за їх атрибутом name
getElementsByClassName()	пошук всіх HTML елементів за їх атрибутом class
querySelectorAll()	пошук всіх HTML елементів, які підходять за заданим CSS селектором (id, імена класів, типи, атрибути, значення атрибутів і т.п.)

Для прямого запису в потік виведення HTML, в JavaScript можна використовувати метод document.write ().

Для маніпуляції елементами документа використовують наступні методи:

1. appendChild – додати елемент у кінець списку;
2. removeChild – вилучити елемент;
3. replaceChild – замінити елемент;
4. getAttribute/setAttribute – отримати/встановити значення атрибута та інші

Найпростіший спосіб змінити вміст HTML елемента - це скористатися властивістю innerHTML. Синтаксис:

document.getElementById(id).innerHTML = новий HTML код

HTML DOM дозволяє змінювати значення атрибутів HTML елементів. Синтаксис:

document.getElementById(id).атрибут = нове значення

Також HTML DOM дозволяє JavaScript змінювати стиль HTML елементів. Синтаксис:

document.getElementById (id).style.властивість = новий стиль

Наведемо приклад коду зміни кольору шрифту для елемента, ідентифікатор якого, має значення p2:

```
<body>
<p id="p2">Hello World!</p>
<script>
document.getElementById("p2").style.color = "blue";
</script>
</body>
```

3. Оброблювачі подій мовою JavaScript:

Більшість дій, які виконує програма мовою JavaScript, - реакції на визначені типи подій від користувача або браузера. Виділяють такі типи подій:

Події DOM. Такі події спрацьовують при натисканні, підведенні/відведенні миші над визначеним елементом, натисканні клавіш клавіатури над елементом, який знаходиться в фокусі, власне при отриманні/втраті фокусу, зміні вмісту елементів форм XHTML, тощо.

Події завантаження. Виникають при закінченні завантаження сторінки, фрейму або зображення, а також при закритті сторінки.

Події вікна та інші події. До цієї групи відносять такі події, як зміна розміру екрану, закінчення завантаження даних з сервера, тощо.

Додати оброблювач подій в JavaScript можна декількома способами:

Використання атрибуту `onПодія` безпосередньо для елемента DOM, наприклад:

```
<a href="#" onclick="alert('Кнопка натиснута')">Гіперпосилання</a>.
```

В такому випадку значення атрибуту містить фрагмент коду JavaScript. Іншими атрибутами подій можуть бути, зокрема: `ondblclick`, `onblur`, `onmousemove` та інші. Такий спосіб є найпростішим, але використовується найчастіше у випадках, коли треба виконати невеликий за обсягом код, наприклад, викликати функцію. Проте його використання призводить до небажаного змішування XHTML та JavaScript коду, що ускладнює подальшу підтримку програм.

Другий спосіб полягає у використанні відповідної властивості об'єкта DOM.

Наприклад реалізуємо модальне вікно при подвійному кліку на елементі, значення атрибуту `id` якого `myEl`:

```
<body>
<p id='myEl'>Параграф</p>
<script>
document.getElementById('myEl').ondblclick =
function() {alert('Подвійне натискання')}
</script>
</body>
```

У Таблиці 19 наведено основні HTML події та їх опис.

Таблиця 19. Основні HTML події та їх опис

Подія	Опис
<code>onkeydown</code>	користувач натискає клавішу
<code>onkeypress</code>	користувач натискає клавішу
<code>onkeyup</code>	користувач відпускає клавішу
<code>onclick</code>	клацання миші на елементі
<code>ondblclick</code>	подвійне клацання миші на елементі
<code>onmousedown</code>	на елементі натиснуто кнопку миші

Подія	Опис
onmousemove	вказівник миші рухається, коли він знаходиться над елементом
onmouseout	вказівник миші виходить за межі елемента
onmouseover	вказівник миші переміщується над елементом
onmouseup	кнопка миші відпускається над елементом
onwheel	колесо миші котиться вгору або вниз над елементом
ondrag	перетягування елемента
ondragend	кінець операції перетягування
ondragenter	перетягування елемента до дійсної точки перетягування
ondragleave	елемент залишає дійсну точку перетягування
ondragover	перетягування елемента над дійсною точкою перетягування
ondragstart	початок операції перетягування
ondrop	перетягування перетягнутого елемента
onscroll	прокручування смуги прокручування елемента

Події onload і onunload спрацьовують, коли користувач заходить на веб-сторінку або залишає її. Подія onload може використовуватися, наприклад, для перевірки типу і версії користувацького браузера і завантаження відповідного варіанту веб-сторінки, ґрунтуючись на цій інформації.

Ще події onload і onunload можуть використовуватися для роботи з файлами куки.

Подія onchange використовується в поєднанні з функціями перевірки полів введення. Нижче показаний приклад, як можна використовувати подію onchange.

Функція upperCase() буде викликатися, коли користувач змінює вміст поля введення:

```
<input type="text" id="fname" onchange="upperCase()">
```

Події onmouseover і onmouseout можуть використовуватися для виклику певних функцій тоді, коли користувач наводить курсор миші на HTML елемент або виводить його з елемента.

Метод addEventListener() приєднує обробник події до певного елемента. При цьому новий обробник події не переписує вже існуючі обробники подій.

Таким чином, можна додавати скільки завгодно обробників подій до одного елемента. При цьому це можуть бути обробники подій одного типу, наприклад, дві події натискання мишкою. Метод addEventListener () дозволяє легко контролювати те, як обробник реагує на, так зване, "спливання" події.

Коли використовується метод addEventListener(), JavaScript відділяється від розмітки HTML, що покращує якість сприйняття при читанні скрипта і дозволяє додавати обробники подій навіть тоді, коли ви не можете контролювати розмітку HTML.

Видалити обробник події може метод removeEventListener(). Синтаксис:

Елемент.addEventListener(подія, функція, useCapture).

Перший параметр -тип події (наприклад, "click" або "mousedown"). Другий параметр - функція, яка буде викликатися при виникненні події. Третій параметр - логічне значення (true / false), що визначає, чи слід відправити подію далі ("спливання") або потрібно закрити цю подію. Цей параметр необов'язковий.

Завдання

1. Вивчити механізми взаємодії з користувачем, методи відбору елементів, DOM та її основні оброблювачі подій.

2. Реалізувати динамічні елементи за допомогою JS відповідно до вимог, які надані у Додатку 5 та згідно варіанту здобувача.

Контрольні запитання

1. Коли браузер створює модель DOM?
2. Який об'єкт використовується для отримання доступу до DOM в коді JavaScript?
3. Якщо код розміщується в <script> в секції <head> сторінки, то чи буде він намагатися змінювати DOM до того, як сторінка буде повністю завантажена?

ЛІТЕРАТУРА: [1–2, 14].

Лабораторна робота 5

РОЗРОБКА ОДНОСТОРІНКОВИХ ЗАСТОСУНКІВ (SPA) ЗАСОБАМИ БІБЛІОТЕКИ REACT

Мета — засвоїти принципи проектування односторінкових застосунків (Single Page Applications), вивчити архітектурні особливості бібліотеки React, набути практичних навичок роботи з JSX, функціональними компонентами та хуками.

Кількість годин – 2

Теоретичні відомості

1. Концепція Single Page Application (SPA):

Односторінковий застосунок (SPA) — це тип веб-застосунку або веб-сайту, що взаємодіє з користувачем шляхом динамічного переписування поточної веб-сторінки новими даними з веб-сервера, замість типового для класичних сайтів завантаження цілих нових сторінок.

React — це декларативна JavaScript-бібліотека з відкритим кодом для створення інтерфейсів користувача. Архітектура React базується на компонентному підході, що забезпечує високу ефективність рендерингу та модульність коду, що є критично важливим для побудови масштабованих SPA-систем.

2. Ключові технології та архітектурні принципи React:

JSX (JavaScript XML) — це розширення синтаксису мови JavaScript, що дозволяє описувати структуру інтерфейсу, візуально нагадуючи HTML. Під час

компіляції (наприклад, за допомогою Babel) вирази JSX трансформуються у виклики функцій JavaScript, які створюють об'єкти React. Приклад синтаксису:

```
const element = <h1>Вітаємо у React!</h1>;
```

Virtual DOM (Віртуальний DOM) — це концепція програмування, в якій "віртуальне" представлення інтерфейсу користувача зберігається в пам'яті як структура даних (дерево об'єктів JavaScript) і синхронізується зі "справжнім" DOM бібліотекою ReactDOM.

Процес синхронізації називається Reconciliation (Узгодження). Коли стан компонента змінюється, React створює нове дерево Virtual DOM. Далі алгоритм порівнює (diffing algorithm) це дерево з попередньою версією, щоб визначити, які саме вузли змінилися. Після виявлення відмінностей React вносить до реального DOM лише необхідний мінімум змін, що значно оптимізує продуктивність застосунку порівняно з повним перемальовуванням сторінки.

3. Компоненти та передача даних:

Компонент у React — це незалежний модуль коду, що відповідає за відображення певної частини інтерфейсу та має власний життєвий цикл. Сучасним стандартом є використання функціональних компонентів. Дані в компоненти передаються через об'єкт props (властивості). Важливо зазначити, що props є незмінними (read-only) для компонента, що їх отримує. Приклад функціонального компонента:

```
// Оголошення компонента, що приймає об'єкт props
function Welcome(props) {
  return <h1>Привіт, {props.name}</h1>;}
```

4. Хуки (Hooks): призначення та класифікація:

Хуки — це механізм, впроваджений у React версії 16.8, який дозволяє використовувати стан та інші можливості React у функціональних компонентах без написання класів. Огляд основних хуків наведено в Таблиці 20.

Таблиця 20. Огляд основних хуків React

Ім'я хука	Опис
useState	Забезпечує збереження внутрішнього стану компонента між рендерами. Повертає кортеж, що складається з поточного значення стану та функції-сетера для його модифікації.
useEffect	Дозволяє виконувати побічні ефекти (side effects) у функціональних компонентах. Використовується для запитів до API, підписок на події, маніпуляцій з DOM тощо. Замінює методи життєвого циклу класових компонентів.
useRef	Повертає мутабельний об'єкт ref, властивість .current якого ініціалізується переданим аргументом. Часто використовується для імперативного доступу до DOM-елементів.

Ім'я хука	Опис
useContext	Надає можливість підписатися на контекст React. Це дозволяє передавати дані глибоко в дерево компонентів без необхідності явно передавати props через кожен рівень вкладеності.

Приклад реалізації компонента з використанням useState:

```
import React, { useState } from 'react';

function Counter() {
  // Ініціалізація змінної стану 'count' зі значенням 0.
  // 'setCount' — це функція для оновлення цього значення.
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Поточний рахунок: {count}</p>
      /* При кліку викликається функція setCount.
      React планує оновлення інтерфейсу,
      викликає компонент Counter знову
      з новим значенням count та оновлює DOM.
      */
      <button onClick={() => setCount(count + 1)}>
        Збільшити
      </button>
    </div>
  );
}
```

5. *Маршрутизація на стороні клієнта:*

Для реалізації навігації в SPA використовується бібліотека React Router. Вона забезпечує зіставлення URL-адреси браузера з конкретними компонентами React, дозволяючи змінювати вміст сторінки без звернення до сервера за новим HTML-документом. Приклад конфігурації маршрутів:

```
// BrowserRouter — оболонка, що використовує HTML5 History API
// Routes — контейнер для всіх можливих маршрутів
// Route — визначення відповідності шляху (path) та
// компонента (element)
<BrowserRouter>
  <Routes>
    <Route path="/" element={<HomePage />} />
    <Route path="/about" element={<AboutPage />} />
  </Routes>
</BrowserRouter>
```

Завдання

1. Вивчити принципи проєктування односторінкових застосунків (SPA) та архітектурні особливості бібліотеки React;
2. Програмно реалізувати односторінковий застосунок за допомогою React, який відповідає тематиці робіт 1, 2 та 4 і містить не менше 75% їх контенту;
3. Провести тестування отриманого SPA.

Контрольні запитання

1. Вкажіть, що найбільшою перевагою React?
2. Чи підходить React для реалізації як великомасштабної програмної системи, так і маломасштабної?
3. Що є React кросбраузерним рішенням для фронтенду?
4. Який тип зв'язування даних використовує React?

ЛІТЕРАТУРА: [14, 15].

Лабораторна робота 6

РОЗРОБКА КОРИСТУВАЦЬКИХ ІНТЕРФЕЙСІВ З ВИКОРИСТАННЯМ ФРЕЙМВОРКІВ VUE.JS ТА NEXT.JS

Мета – ознайомитися з архітектурою прогресивного фреймворку Vue.js, засвоїти структуру однофайлових компонентів (SFC), вивчити концепцію реактивності даних та опанувати роботу з директивами. А також дослідити технології попереднього рендерингу веб-сторінок, засвоїти принципи маршрутизації на основі файлової системи, вивчити особливості роботи з серверними компонентами (React Server Components) та реалізувати отримання даних на стороні сервера на основі фреймворку Next.js.

Кількість годин – 2

Теоретичні відомості

1. *Концепція прогресивного фреймворку:*

Vue.js – це JavaScript-фреймворк для створення користувацьких інтерфейсів. Його архітектура визначається як "прогресивна", що передбачає можливість поетапного впровадження. Ядро бібліотеки зосереджене виключно на шарі відображення (View), що дозволяє інтегрувати Vue в існуючі сторінки для вирішення локальних задач. Водночас, екосистема Vue містить інструменти для побудови складних односторінкових застосунків (SPA), де фреймворк керує всім інтерфейсом.

2. *Структура однофайлового компонента (Single File Component):*

У розробці на Vue.js загальноприйнятим стандартом є використання файлів з розширенням .vue. Такий підхід називається Single File Component (SFC) і дозволяє інкапсулювати структуру, логіку та стилі одного елемента інтерфейсу в одному файлі. Компонент складається з трьох основних блоків:

- <template>: Містить розмітку HTML, що визначає структуру компонента;
- <script>: Містить код JavaScript, що експортує об'єкт з логікою компонента;

•<style>: Містить CSS-стилі, які можуть бути локальними для цього компонента.

Приклад структури файлу .vue:

```
<template>
  <div class="container">
    <h1>Заголовок компонента</h1>
  </div>
</template>

<script>
export default {
  name: 'MyComponent',
  // Тут розміщується логіка компонента
}
</script>

<style scoped>
/* Атрибут scoped обмежує дію стилів лише цим компонентом */
.container {
  padding: 20px;
}
</style>
```

3. Система реактивності даних:

Система реактивності та методи компонента. Реактивність у Vue.js дозволяє автоматично оновлювати DOM-дерево при зміні стану JavaScript-об'єктів. Логіка компонента будується на взаємодії двох основних властивостей:

1. data() – функція, що повертає об'єкт із початковим станом (змінними). Саме ці властивості стають реактивними: як тільки їх значення змінюється, Vue оновлює шаблон.
2. methods – об'єкт, у якому описуються функції (методи) для обробки подій користувача або виконання бізнес-логіки.

Важливо про контекст this: Щоб звернутися до реактивних змінних (оголошених у data) зсередини функцій (оголошених у methods), необхідно використовувати префікс this. Наприклад, якщо у data є змінна count, то всередині методу до неї слід звертатися як this.count. Без використання this JavaScript буде шукати локальну змінну всередині функції, а не властивість компонента.

Приклад оголошення реактивного стану:

```
<script>
export default {
  // 1. Оголошення реактивних даних
  data() {
    return {
      userName: 'Студент',
      isLoggedIn: true,
      score: 0
    }
  }
}
```

```

    }
  },

  // 2. Оголошення методів для роботи з даними
  methods: {
    resetScore() {
      // Використовуємо this для доступу до score
      this.score = 0;
    }
  }
}
</script>

```

4. Директиви шаблонів:

Директиви – це спеціальні атрибути в шаблоні Vue, які починаються з префікса `v-` та застосовують спеціальну реактивну поведінку до відрендереного DOM-елемента. Значенням директиви зазвичай виступає один вираз JavaScript, результат виконання якого безпосередньо визначає зміни в об'єктній моделі документа. Окрім основного виразу, повний синтаксис директиви може бути розширений за рахунок використання аргументів та модифікаторів. Аргумент вказується безпосередньо після назви директиви через двокрапку і конкретизує параметр елемента, на який здійснюється вплив (наприклад, у записі `v-bind:href` аргументом виступає `href`). У свою чергу, модифікатори являють собою спеціальні суфікси, що позначаються крапкою і визначають специфічний алгоритм обробки директиви, як-от `.prevent` у конструкції `v-on:submit.prevent`, що скасовує стандартну поведінку браузера при відправці форми.

Таблиця 21. Огляд основних директив Vue.js

Назва директиви	Опис функціоналу
<code>v-bind</code>	Забезпечує динамічне зв'язування атрибутів HTML (наприклад, <code>src</code> , <code>href</code> , <code>class</code>) зі значенням у моделі даних.
<code>v-on</code>	Встановлює слухач подій DOM (наприклад, клік миші, натискання клавіші) та викликає методи компонента при їх настанні.
<code>v-if</code> / <code>v-else</code>	Здійснює умовний рендеринг: елемент фізично додається або видаляється з DOM-дерева залежно від істинності умови.
<code>v-for</code>	Використовується для рендерингу списку елементів на основі масиву або об'єкта даних.
<code>v-model</code>	Створює двостороннє зв'язування даних (two-way binding) між елементом форми (<code>input</code> , <code>textarea</code> , <code>select</code>) та змінною стану.

Приклад використання директиви `v-on` для обробки подій:

У наведеному нижче прикладі демонструється взаємодія Logic (JS) та Template (HTML). При натисканні кнопки спрацьовує директива v-on, яка викликає метод increment з секції methods. Всередині методу використовується звернення this.count++, що змінює реактивну змінну. Завдяки системі реактивності, зміна числового значення автоматично оновлює текст у тегу <button>.

```
<script>
export default {
  data() {
    return {
      count: 0
    }
  },
  methods: {
    increment() {
      // "this" надає доступ до властивостей з data()
      this.count++
    }
  }
}
</script>

<template>
<button v-on:click="increment">
  Кількість кліків: {{ count }}
</button>
</template>
```

5. Парадигма гібридного рендерингу. Next.js:

Next.js – це мета-фреймворк, побудований поверх бібліотеки React, який розширює її можливості за рахунок перенесення частини обчислень з клієнта (браузера) на сервер. Ключовою архітектурною відмінністю Next.js є можливість попереднього рендерингу (Pre-rendering). Це означає, що HTML-код сторінки генерується заздалегідь або на сервері під час запиту, а не виключно засобами JavaScript у браузері користувача. Такий підхід вирішує проблеми SEO-оптимізації та покращує показники продуктивності, забезпечуючи пошуковим роботам готовий HTML-код, що покращує індексацію. Висока швидкість досягається за рахунок автоматичного розділення коду (code splitting), оптимізації зображень, шрифтів та попереднього завантаження сторінок (prefetching), що значно підвищує показники Core Web Vitals.

6. Стратегії генерації контенту:

Фреймворк дозволяє розробнику обирати оптимальну стратегію рендерингу для кожної окремої сторінки залежно від вимог до актуальності даних. Основні стратегії рендерингу наведено в Таблиці 22, яка розташована нижче.

Таблиця 22. Класифікація стратегій рендерингу в Next.js

Стратегія	Опис процесу генерації
Server-Side Rendering (SSR)	Генерація HTML відбувається на сервері при кожному вхідному HTTP-запиті. Це забезпечує відображення актуальних даних, але збільшує час відповіді сервера (Time to First Byte).
Static Site Generation (SSG)	HTML-сторінки генеруються один раз під час збірки проєкту (build time). Отримані файли кешуються та роздаються через CDN. Цей метод є найшвидшим, але підходить лише для статичного контенту.
Incremental Static Regeneration (ISR)	Гібридний підхід, що дозволяє оновлювати статичні сторінки (SSG) у фоновому режимі після розгортання застосунку, без необхідності повної перезбірки проєкту.
Client-Side Rendering (CSR)	Стандартний для SPA підхід, коли ініціалізація інтерфейсу та завантаження даних відбуваються у браузері після завантаження JavaScript-банду.

7. Маршрутизація на основі файлової системи (App Router):

У Next.js (версія 13+) використовується маршрутизатор App Router. Маршрути визначаються ієрархією папок всередині директорії app. Файл page.js відповідає за відображення унікального контенту маршруту, а файл layout.js визначає спільний інтерфейс (Header, Footer, Sidebar) для групи маршрутів.

Кореневий макет (app/layout.js) є обов'язковим елементом архітектури і повинен містити базову структуру HTML-документа.

Приклад реалізації глобального макету:

```
// app/layout.js
import './globals.css';
```

```
export const metadata = {
  title: 'Навчальний портал',
  description: 'Лабораторна робота з Next.js',
};
```

```
// Функціональний компонент приймає prop 'children' – контент поточної сторінки
export default function RootLayout({ children }) {
  return (
    <html lang="uk">
    <body>
    <header>
    <nav>
    <Link href="/">Головна</Link>
```

```

</nav>
</header>

{/* Рендеринг вкладки контенту (page.js) */}
<main>{children}</main>

<footer>
<p>© 2024 Університетський портал</p>
</footer>
</body>
</html>
);
}

```

8. *Серверні компоненти та отримання даних:*

Архітектура React Server Components (RSC) дозволяє компонентам рендеритися виключно на сервері. Це зменшує розмір JavaScript-коду, що надсилається клієнту, оскільки бібліотеки, використані на сервері, не потрапляють у клієнтський бандл. Отримання даних у серверних компонентах виконується за допомогою асинхронних функцій та стандартного Web API fetch. Оскільки код виконується на сервері, розробник має прямий доступ до баз даних або внутрішніх мікросервісів без необхідності створення проміжного API-шару.

Для реалізації динамічної маршрутизації – створення сторінок, які залежать від змінних параметрів (наприклад, ID користувача) – використовується назва папки у квадратних дужках. Наприклад, шлях `app/users/[id]/page.js` дозволить обробляти запити типу `/users/1` або `/users/123`, а значення цього параметра буде доступне всередині компонента через об'єкт `params`.

Водночас варто пам'ятати про обмеження серверних компонентів: вони не підтримують клієнтську інтерактивність (обробники подій на кшталт `onClick`, хуки `useState` чи `useEffect`). Якщо компоненту необхідна така функціональність, у першому рядку файлу слід додати директиву `'use client'`, що явно перетворить його на клієнтський компонент.

Приклад серверного компонента з асинхронним отриманням даних:

У наведеному прикладі компонент `UserProfile` є асинхронним. Виконання коду зупиняється на рядку з `await fetch`, поки сервер отримує дані. Браузер отримає вже готовий HTML з підставленим ім'ям та поштою.

```

export default async function UserProfile({ params }) {
  // Виконання запиту до зовнішнього API на стороні сервера
  // next: { revalidate: 60 } вмикає ISR (оновлення кешу кожні 60 с)
  const response = await fetch(`https://api.example.com/users/${params.id}`, {
    next: { revalidate: 60 }
  });

  // Обробка відповіді
  if (!response.ok) {
    throw new Error('Не вдалося завантажити дані користувача');
  }
}

```

```
const user = await response.json();

// Рендеринг HTML з отриманими даними
return (
  <div className="profile-container">
    <h1>Профіль користувача: {user.name}</h1>
    <p>Email: {user.email}</p>
    <p>Статус: {user.isActive ? 'Активний' : 'Заблокований'}</p>
  </div>
);
}
```

Завдання

1. Засвоїти архітектуру прогресивного фреймворку Vue.js, структуру однофайлових компонентів (SFC) та концепцію реактивності даних;
2. Вивчити технології попереднього рендерингу веб-сторінок, принципи маршрутизації на основі файлової системи та особливості роботи з серверними компонентами (React Server Components) на основі Next.js;
3. Програмно реалізувати два односторінкових застосунки (SPA) за допомогою фреймворків Vue.js та Next.js, які відповідають тематиці робіт 1, 2 та 4 і містять не менше 75% їх контенту;
4. Провести тестування отриманих SPA.

Контрольні запитання

1. Чи дозволяє Vue.js розробникам масштабуватися від простого фреймворку до повного зовнішнього рішення?
2. Який шаблон використовує Vue.js для створення інтерфейсів користувача на основі моделей даних, через реактивне зв'язування даних?
3. Що таке директиви у Vue.js?
4. Який механізм покращує швидкодію завантаження сторінок, особливо для SEO та випадків, коли потрібно отримати дані на сервері перед відображенням сторінки?
5. Чи підтримує Next.js масштабування?
6. Якщо веб-додаток включає багато сторінок, компонентів та станів, чи доцільним є використання Next.js для його розробки?

ЛІТЕРАТУРА: [14, 16-17].

Лабораторна робота 7

ПРОЄКТУВАННЯ МАСШТАБОВАНИХ ВЕБ-ЗАСТОСУНКІВ НА ПЛАТФОРМІ ANGULAR

Мета — засвоїти принципи побудови застосунків корпоративного рівня засобами платформи Angular, вивчити особливості використання мови TypeScript та декораторів, опанувати модульну архітектуру та патерн «Впровадження залежностей» (Dependency Injection).

Теоретичні відомості

1. *Архітектура платформи Angular:*

Angular – це компонентний фреймворк для розробки динамічних веб-застосунків, який використовує мову TypeScript. Платформа надає комплексний набір інструментів, інтегрованих в єдину екосистему, що включає засоби для маршрутизації, обробки форм, взаємодії з сервером через HTTP-клієнт та тестування. Архітектура застосунку базується на модульності (NgModules), що дозволяє логічно групувати компоненти, директиви та канали, забезпечуючи структурованість коду та можливість "лінивого" завантаження (lazy loading). На відміну від бібліотек (як React), Angular є повноцінним фреймворком, що диктує чітку структуру проєкту.

2. *Структура компонента та Декоратори:*

Основним будівельним блоком інтерфейсу в Angular є компонент. Технічно компонент являє собою клас TypeScript, до якого застосовано декоратор `@Component`.

Декоратори – це спеціальна синтаксична конструкція TypeScript, яка дозволяє додавати метадані до класів, методів або властивостей. У випадку компонента декоратор повідомляє компілятору Angular, з яким HTML-шаблоном та стилями пов'язаний даний клас, а також визначає його селектор (тег) для використання в розмітці.

Приклад структури компонента:

```
import { Component } from '@angular/core';

// Декоратор визначає метадані компонента
@Component({
  selector: 'app-user-card', // Кастомний HTML-тег
  template: `
    <div class="card">
      <h2>{{ title }}</h2>
      <button (click)="activateUser()">Активувати</button>
    </div>
  `,
  styles: ['.card { border: 1px solid #ccc; padding: 1rem; }']
})
export class UserCardComponent {
  // Властивості класу (State)
  title: string = 'Профіль користувача';
  isActive: boolean = false;

  // Метод класу (Logic)
  activateUser(): void {
    this.isActive = true;
    console.log('Користувача активовано');
  }
}
```

3. Механізм *Dependency Injection (DI)* та *Сервіси*:

Для відокремлення бізнес-логіки (обробка даних, звернення до API) від логіки відображення (UI) в Angular використовується патерн *Dependency Injection* (Впровадження залежностей). Логіка виноситься у спеціальні класи – *Сервіси*, які позначаються декоратором `@Injectable`. Система DI в Angular відповідає за створення екземплярів сервісів та передачу їх у компоненти, які цього потребують. Це відбувається через конструктор класу компонента. Такий підхід забезпечує слабку зв'язаність коду (*loose coupling*) та полегшує тестування.

Приклад впровадження сервісу:

```
import { Injectable, Component } from '@angular/core';

// 1. Оголошення сервісу
@Injectable({
  providedIn: 'root', // Сервіс доступний у всьому застосунку (Singleton)
})
export class LoggerService {
  log(message: string) {
    console.log(`Log entry: ${message}`);
  }
}

// 2. Використання сервісу в компоненті
@Component({
  selector: 'app-log-button',
  template: `<button (click)="doLog()">Записати лог</button>`
})
export class LogComponent {
  // Ін'єкція сервісу через конструктор
  constructor(private logger: LoggerService) {}

  doLog() {
    this.logger.log('Кнопку натиснуто');
  }
}
```

4. *Прив'язка даних (Data Binding)*:

Механізм прив'язки даних в Angular координує взаємодію між шаблоном (HTML) та компонентом (TypeScript). Він дозволяє відображати значення властивостей класу в інтерфейсі та реагувати на дії користувача.

Таблиця 23. Типи прив'язки даних

Тип прив'язки	Синтаксис	Опис механізму
Interpolation (Інтерполяція)	{{ value }}	Одностороння прив'язка від джерела до представлення. Використовується для вбудовування рядкових значень у текст HTML.

Тип прив'язки	Синтаксис	Опис механізму
Property Binding (Прив'язка властивостей)	[property]="value"	Встановлює значення властивості DOM-елемента або директиви на основі виразу класу.
Event Binding (Прив'язка подій)	(event)="handler()"	Дозволяє компоненту слухати події DOM (кліки, натискання клавіш) та викликати відповідні методи.
Two-way Binding (Двостороння прив'язка)	[(ngModel)]="value"	Забезпечує синхронізацію даних: зміни в UI оновлюють клас, а зміни в класі оновлюють UI (часто використовується у формах).

Примітка: Для використання [(ngModel)] необхідно імпортувати модуль FormsModule з пакету @angular/forms у модуль застосунку (AppModule).

5. Структурні директиви:

Структурні директиви відповідають за маніпуляції з макетом DOM-дерева шляхом додавання, видалення або заміни елементів. Синтаксичною особливістю структурних директив є наявність префікса * перед ім'ям директиви. Цей символ є "синтаксичним цукром", який вказує Angular на необхідність розгорнути директиву в елемент <ng-template>. Приклад використання *ngIf та *ngFor:

```
@Component({
  selector: 'app-list',
  template: `
    <ul *ngIf="users.length > 0">
      <li *ngFor="let user of users">
        {{ user.name }}
      </li>
    </ul>
    <p *ngIf="users.length === 0">Список порожній</p>
  `
})

export class ListComponent {
  users = [{ name: 'Alice' }, { name: 'Bob' }];
}
```

Завдання

1. Вивчити принципи побудови застосунків корпоративного рівня засобами платформи Angular та особливості використання мови TypeScript та декораторів;
2. Програмно реалізувати односторінковий застосунок за допомогою фреймворку Angular, який відповідає тематиці робіт 1, 2 та 4 і містить не менше 75% їх контенту;
3. Провести тестування отриманого SPA.

Контрольні запитання

1. Який тип зв'язування даних для представлення шару моделі підтримує Angular?
 2. Який тип архітектури підтримує Angular?
 3. Чи використовує Angular концепцію «області видимості»?
 4. Вкажіть, які основні особливості мови TypeScript?
 5. Вкажіть, які вимоги до фронтенду програмної системи, та команди, що її розробляє, необхідні для того, щоб використання Angular було виправданим?
- ЛІТЕРАТУРА: [1, 14, 18].**

Лабораторна робота 8

АДАПТИВНІСТЬ, КРОСБРАУЗЕРНІСТЬ ТА МЕТРИКИ ЕФЕКТИВНОСТІ ФРОНТЕНДУ

Мета – набути практичних навичок у реалізації адаптивності фронтенду, методів та підходів, які забезпечують його кросбраузерність, а також в оцінці ефективності розробленого продукту.

Кількість годин – 2

Теоретичні відомості

Розробка фронтенду програмних систем тісно пов'язана з його тестуванням. Саме цей етап життєвого циклу продукту забезпечує його адаптивність та кросбраузерність, а також передбачає його рефакторинг.

1. Адаптивність фронтенду:

Адаптивність фронтенду – це його здатність автоматично підлаштовуватись під різні розміри екранів та пристроїв, на яких він переглядається. Відповідність розробленого продукту вимогам адаптивності включає оптимізацію дизайну, швидкості завантаження, розміщення елементів і загального користувацького досвіду.

Існує два основні підходи до реалізації адаптивності:

- responsive web design (RWD) або чутливий дизайн;
- adaptive web design (AWD) або адаптивний дизайн.

Основна відмінність чутливого й адаптивного дизайну — у реакції на зміну розміру екрана. Чутливий дизайн допомагає динамічно підлаштовувати контент під розміри та роздільну здатність екрана для одного й того ж макета, адаптивний — підбрати найбільш відповідний із декількох макетів і відобразити його користувачеві.

На сьогодні, частіше використовується чутливий дизайн, оскільки створення та підтримання єдиного універсального макета займає менше часу й коштує дешевше.

Адаптивний дизайн виправданий у складних проєктах, заточених під певні пристрої: наприклад, під час розроблення сайту онлайн-банкінгу, більшість користувачів якого відвідують сайт з останніх моделей iPhone.

1.1. Responsive web design (RWD) або чутливий дизайн:

Чутливий вебдизайн базується на трьох принципах:

- гнучкий макет на основі сітки (flexible, grid-based layout);
- гнучкі зображення (flexible images);
- медіазапит (CSS3 media queries).

Гнучкість макета базується на використанні відносних одиниць вимірювання замість фіксованих піксельних значень. Проблема гнучких зображень вирішується за допомогою правила `img (width: 100%; max-width: 100%;)` для всіх картинок на сайті. Це правило гарантує, що зображення ніколи не будуть ширше, ніж їхні контейнери і ніколи не перевищать своїх справжніх розмірів на великих екранах. У 2022 році в CSS з'являються наступні одиниці:

- `lvh` (Large Viewport Height) — ця одиниця відображає повну висоту екрана, коли інтерфейс браузера не впливає на доступну висоту;

- `svh` (Small Viewport Height) — на відміну від `lvh`, ця одиниця вимірює мінімальну можливу висоту екрану, коли інтерфейс браузера займає більше простору. Наприклад, при першому завантаженні сторінки на мобільному телефоні, коли видимий адресний рядок ще не прихований. Вона корисна для тих випадків, коли необхідно адаптувати дизайн до ситуації, де доступної висоти менше.

- `dvh` (Dynamic Viewport Height) — це динамічна одиниця, яка враховує зміни в інтерфейсі браузера в режимі реального часу. Вона свого роду динамічне поєднання двох попередніх. Наприклад, коли користувач взаємодіє з екраном, і адресний рядок зникає або з'являється, `dvh` автоматично підлаштовує висоту елементів під ці зміни.

Гнучкі зображення необхідні, щоб підібрати оптимальний варіант для конкретного пристрою залежно від розміру, орієнтації та типу дисплея.

Для цього використовуються атрибути `srcset` і `sizes`: перший дає можливість вказати шляхи до всіх варіацій зображення та розміри в пікселях, другий — задавати критерії вибору того чи іншого зображення через медіазапит.

За допомогою атрибутів можна підказати браузеру, яку версію картинки завантажувати: для вузького екрана — полегшену версію для смартфонів, для широкого телевізора — картинку в потрібному розмірі.

Медіазапити змінюють стилі на підставі характеристик пристрою, пов'язаних з відображенням контенту, включаючи тип, ширину, висоту, орієнтацію й характеристики екрану. Це правила CSS, які дозволяють керувати стилями елементів залежно від значень технічних параметрів пристроїв. Браузер застосовує таблицю стилів тільки в разі, якщо активувався саме цей тип пристрою.

Загалом медіа-запит складається з ключового слова, що описує тип пристрою (необов'язковий параметр) і вираз, котрий перевіряє характеристики цього пристрою. З усіх характеристик найчастіше перевіряється ширина пристрою `width`. Медіа-запит є логічним виразом, який повертає істину або хибність.

Запит записують у кінці таблиці стилів (у стилів, які записані пізніше, вища пріоритетність).

Основні типи носіїв у медіа-запитах: all – використовується за замовчуванням; handheld – смартфони та аналогічні пристрої; print – принтери та інші друкарські пристрої; projection – проектор; screen – екран монітора; speech – мовні синтезатори; tv – телевізори.

Список популярних медіа-запитів для стандартного набору розширення екранів:

```
@media only screen and (max-width : 1920px) { /* CSS правила */ }
@media only screen and (max-width : 1680px) {}
@media only screen and (max-width : 1366px) {}
@media only screen and (max-width : 1280px) {}
@media only screen and (max-width : 1024px) {}
@media only screen and (max-width : 800px) {}
@media only screen and (max-width : 768px) {}
@media only screen and (max-width : 600px) {}
@media only screen and (max-width : 533px) {}
@media only screen and (max-width : 360px) {}
@media only screen and (max-width : 320px) {}
@media only screen and (max-width : 240px) {}
@media only screen and (max-width : 176px) {}
```

Наприклад, якщо ви хочете створити макет із трьома колонками для великих екранів та макет з однією колонкою для малих екранів (наприклад, телефонів), ви можете змінити напрямок flex-напрямку з рядка на колонку в певній точці зупинки (600 пікселів у прикладі нижче):

```
.flex-container {
  display: flex;
  flex-direction: row;
}

.flex-item {
  padding: 10px;
  font-size: 30px;
  text-align: center;
  width: 100%;
}

/* Make a one column-layout instead of three-column layout */
@media (max-width: 600px) {
  .flex-container {
    flex-direction: column;
  }
}
```

1.2. *Adaptive web design (AWD) або адаптивний дизайн:*

Його основою є декілька фіксованих макетів найпопулярніших розмірів екрана. Усі варіанти фронтенду розгортаються за одним і тим самим URL.

Зазвичай розробляють шість макетів відповідно до найпопулярнішої ширини екрана: 320 px, 480 px, 760 px, 960 px, 1200 px, 1600 px.

Головне в адаптивній верстці — це прив'язка до конкретних пристроїв з певною роздільністю екрану.

Стилі перемикаються від одного брейкпоінта (контрольні точки, сталий термін у програмуванні, у випадку верстки означає перемикання з одних умов на інші) до іншого. Тобто насправді існують фіксовані макети для кожного девайсу.

Деякі інструменти для адаптивної верстки:

- "Media Query Inspector" дає змогу розробникам перевіряти та регулювати медіа-запити в реальному часі, забезпечує точне налаштування відображення контенту на пристроях будь-якого типу;

"Responsive Design Mode" в інструментах розробника браузера також стає незамінним. Він надає прев'ю для різних розмірів екранів, що дає змогу миттєво оцінювати, який вигляд сайт має на мобільних гаджетах і планшетах.

Метрики ефективності фронтенду охоплюють багато аспектів розробки. Зокрема, продуктивність (швидкість завантаження та взаємодії), якість коду (читабельність, підтримка, баги) та UX/UI (зручність користувача, конверсії), включаючи швидкість (Time to Interactive), використання ресурсів (пам'ять, CPU), кількість помилок у консолі та показники Google Core Web Vitals, що разом допомагають покращити досвід користувача та роботу програмної системи.

2. *Кросбраузерність:*

Кросбраузерність — це правильна верстка фронтенду, за допомогою якої він однаково відображається в різних браузерах та їх версіях. Це пов'язано зокрема з відсутністю єдиного стандарту при розробці браузерів. Тому, кожний переглядач має свої відмінності в інтерпретації гіпертексту, що пов'язано з різною реалізацією рушіїв зображення та створенням пропрієтарних тегів для розширення можливостей стандартного HTML. Так, один текст може бути відтворено по-різному в Internet Explorer та Safari.

Забезпечення кросбраузерності відбувається за допомогою HTML і CSS, а також за допомогою різноманітних хаків, а в окремих випадках — JavaScript. Вона досягається декількома способами:

- HTML використовує лише ті елементи, які відображаються у всіх браузерах однаково. Потрібно віддавати перевагу універсальним елементам — тим, які однаково відображаються у більшості браузерів;
- хаками – набором спеціальних правил та селекторів, який розуміє певний браузер;
- реалізацією на основі готових фреймворків та бібліотек.

У кожного браузера є свої вбудовані, експериментальні, або нестандартні властивості, для коректної роботи яких, використовують вендорні префікси. Такі властивості змінюють поведінку елемента тільки в певному браузері та ігноруються іншими платформами.

Перелік вендорних префіксів для різних браузерів:

- -webkit- для Chrome, Safari, Opera, Microsoft Edge;

- -moz- для Firefox;
- -o- для ранніх версій Opera;
- -ms- для ранніх версій Microsoft Edge (до 18 версії) та Internet Explorer.

Наприклад, елемент `border-radius` у MozillaFirefox представлений властивістю `-moz-border-radius`, а в Chrome і Safari— `-webkit-border-radius`. Наведемо приклад уніфікації властивості елемента мати тінь від рамки:

```
.element_with_border_shadow {
  -webkit-box-shadow: 0 0 4px rgba (0, 0, 0, 0.5);
  -moz-box-shadow: 0 0 4px rgba (0, 0, 0, 0.5);
  -ms-box-shadow: 0 0 4px rgba (0, 0, 0, 0.5);
  box-shadow: 0 0 4px rgba (0, 0, 0, 0.5);
}
```

Як відомо, кожен браузер за замовчуванням має певний набір базових стилів, які він застосовує до сторінки. У різних браузерах ці правила трохи відрізняються. Щоб їх усунути та зробити за замовчуванням відображення сторінки у всіх браузерах однаковим, використовують спеціальні CSS-файли: `reset.css` або `normalize.css`. Файл `reset.css` містить перелік усіх можливих HTML-тегів і скидає їх значення в нуль, тобто прибирає всі можливі відступи, робить шрифт однаковим у всіх тегах. Таким чином, всі заголовки й абзаци відображаються простим текстом, одним розміром і без відступів. Як наслідок відбувається скидання стилів за замовчуванням у всіх браузерах. Спочатку на сторінці користувачу слід підключити файл `reset.css`, а потім власний файл зі стилями `style.css`. У будь-якому браузері вся розмітка ґрунтуватиметься на тих стилях, які потрібно поставити в `style.css`.

`Normalize.css` — продукт глибокого дослідження відмінностей між початковими стилями браузера. Дослідження провів Ніколас Галахер, взявши за мету: зберігати корисні налаштування; нормалізувати стилі для більшості HTML-елементів; коригувати помилки й основні невідповідності браузера; удосконалювати юзабіліті непомітними поліпшеннями; пояснювати код, використовуючи коментарі та детальну документацію.

Наразі `normalize.css` використовується в Twitter Bootstrap, HTML5 Boilerplate, GOV.UK, Rdio, CSS Tricks і в багатьох інших фреймворках, інструментах, тощо.

3. *Метрики продуктивності (Performance Metrics):*

LCP (Largest Contentful Paint) — це показник швидкості завантаження, що вимірює час від початку завантаження сторінки до моменту, коли браузер повністю відобразить найбільший видимий елемент контенту (наприклад, велике зображення, відео чи текстовий блок), а ідеально – цей час має бути меншим за 2.5 секунди, щоб користувач не сприймав сторінку як повільну, що покращує досвід та позиції в пошуку Google.

FID (First Input Delay) / INP (Interaction to Next Paint): швидкість реакції на першу взаємодію користувача (клік, тап). є показником, який оцінює загальну інтерактивність сторінок сайтів на основі часу реакції на взаємодію користувача.

INP є результатом спостереження за всіма кваліфікованими взаємодіями, що відбуваються протягом візиту користувача на сторінку.

Метрика Interaction to Next Paint призначена для оцінки швидкості відгуку на дії користувача. Тож вона передбачає вимірювання затримки введення: від моменту взаємодії користувача зі сторінкою до появи візуальної відповіді. INP реєструє затримку під час взаємодії з вебсторінкою, навіть коли відбувається її перезавантаження. На думку розробників Google, швидкість відгуку має вирішальне значення, адже людина має бачити, що сайт реагує на взаємодію з нею.

CLS (Cumulative Layout Shift): стабільність верстки (відсутність "стрибків" елементів). Вона вимірює неочікувані проблеми візуальної стабільності на вебсторінках, де контент перескакує під час завантаження сторінки, дратуючи користувачів і потенційно спричиняючи випадкові кліки.

Час до інтерактивності (Time to Interactive, TTI) — це метрика вебпродуктивності, що визначає момент, коли веб-сторінка стає повністю функціональною та здатною реагувати на дії користувача (кліки, введення тексту). Це відбувається після відображення корисного контенту (FCP) та реєстрації обробників подій для видимих елементів, при цьому браузер не виконує довгих завдань і може миттєво відповідати на взаємодії, що покращує досвід користувача та SEO.

Час завантаження JavaScript: як швидко виконується JS. Ідеально, коли він не перевищує 1-2 секунди для швидкого досвіду користувача. Для вимірювання використовують інструменти розробника в браузері, API Performance, а оптимізація включає відкладене завантаження, мініфікацію та асинхронні запити.

Розмір бандла (Bundle Size): загальний розмір усіх файлів (HTML, CSS, JavaScript, зображень), зібраних разом для веб-сторінки чи застосунку, і його зменшення критично важливе для швидкості завантаження, продуктивності та позитивного користувацького досвіду, оскільки більші бандли сповільнюють сайт, вимагають більше трафіку та ресурсів. Цей розмір включає код компонентів, стилі та залежності, і оптимізується за допомогою технік трейшкінгу (treeshaking), лінивої загрузки (lazy loading), оптимізації імпорту та видалення невикористовуваного коду.

Використання пам'яті та CPU: навантаження на пристрій користувача.

3.1. Метрики якості коду та розробки:

Щільність дефектів (Defect Density): ключова метрика якості ПЗ, що показує кількість знайдених багів (дефектів) на певну одиницю продукту, найчастіше — на 1000 рядків коду (KLOC) або на функцію/модуль, яка допомагає оцінити стабільність коду та ефективність тестування, вказуючи на проблеми в певних частинах системи чи загальну якість розробки.

Цикломатична складність (Cyclomatic Complexity): вимірює складність логіки функцій чи методів, показуючи кількість незалежних шляхів виконання.

Час виправлення багів (Bug Fix Time): скільки часу займає усунення дефекту.

Розмір коду (LOC, Lines of Code): відноситься до загальної метрики, що використовується для вимірювання розміру класу, функції або загальної кількості рядків усієї реалізації. Вона кількісно визначає кількість рядків коду в програмі або програмному забезпеченні.

Метрики зв'язності (Coupling) та згуртованості (Cohesion): вимірюють, наскільки залежні класи між собою та наскільки функціонально об'єднані елементи класу.

3.2. Метрики UX та бізнесу:

Коефіцієнт конверсії (Conversion Rate): показник ефективності в маркетингу, що показує відсоток користувачів, які виконали цільову дію (купівлю, реєстрацію, підписку, заповнення форми) з-поміж загальної кількості відвідувачів або взаємодій. Розраховується за формулою: $(\text{Кількість цільових дій} / \text{Загальна кількість відвідувачів}) * 100\%$. Цей метрика допомагає оцінити ефективність сайту та рекламних кампаній, дозволяючи збільшувати прибуток без збільшення витрат на трафік.

Час на сайті / глибина перегляду: як довго користувач залишається на сайті та скільки сторінок переглядає.

Показник відмов (Bounce Rate): відсоткове співвідношення кількості відвідувачів, що покинули сайт прямо зі сторінки входу або переглянули не більше однієї сторінки сайту.

Задоволеність користувачів (NPS, CSAT): опитування для оцінки загального досвіду. NPS (Net Promoter Score) оцінює лояльність та готовність рекомендувати бренд (від -100 до +100), та CSAT (Customer Satisfaction Score), яка вимірює задоволеність конкретною взаємодією чи продуктом (зазвичай за шкалою 1-5 або 1-10, виражається у відсотках). CSAT фокусується на поточних транзакціях, даючи швидкий зворотний зв'язок, тоді як NPS вимірює довгострокову відданість і є прогностичною метрикою лояльності.

Завдання

1. Вивчити основні прийоми реалізації гнучкого та адаптивного дизайну фронтенду програмних систем, способи досягнення кросбраузерності фронтенду та ознайомитись з функціоналом інструментів для вимірювання ефективності фронтенду: Google PageSpeed Insights, Lighthouse, Google PageSpeed Insights, Lighthouse, Chrome DevTools, WebPageTest, Sentry, New Relic, Datadog;

2. Провести перевірку адаптивності реалізованого у лабораторних роботах 1-4 фронтенду для стандартного набору розширень екранів пристроїв, забезпечивши зручний користувацький досвід у навігації, перегляді меню та каталогів, а також заповненні форми зворотнього зв'язку, за допомогою інструментів розробника. Наприклад, можна скористатись F12, що у більшості браузерів відкриває front-end development/debugging. За необхідності внести зміни у код: замінити абсолютні одиниці виміру на відносні (див. Лабораторна робота 2); всі зображення зробити гнучкими; забезпечити наявність медіа-запитів для різних типів розширень екранів; реалізувати брейкпоінт-макети;

3. Провести перевірку кросбраузерності реалізованого у лабораторних роботах 1-4 фронтенду, сформувавши таблицю відповідності між використаними тегами та їх підтримкою браузерами Chrome, Edge/Internet Explorer, Firefox, Safari та Opera, використовуючи довідник з підтримки тегів різними браузерами. За необхідності внести зміни у код за допомогою вендорних префіксів у реалізації стилістики тих елементів, властивості яких відрізняються у Chrome, Edge/Internet Explorer, Firefox, Safari та Opera [5];

4. Провести перевірку адаптивності та кросбраузерності продуктів, реалізованих у лабораторних роботах 5-7, та сформувати таблицю, що відображає отримані результати;

5. Провести заміри основних метрик ефективності програмних продуктів розроблених у роботах 1-7 та оформити відповідно до них таблиці, які відображають отримані результати. Провести аналіз причин наявності низьких показників ефективності.

Контрольні запитання

1. Чи є надмірне використання стилів кроком на зустріч адаптивності?
2. Чи попередній перегляд вигляду фронтенду на емуляторах повністю забезпечує адаптивність?
3. Чим відрізняється адаптивність від доступності?
4. Перелічіть проблеми, які виникають через відмінності у браузерах?
5. Яка основна причина наявності відмінностей між різними браузерами?
6. Яка роль HTML5 у забезпеченні кросбраузерності фронтенду?
7. Як досягнути кросбраузерності фронтенду, використовуючи правила CSS?
8. Які фреймворки спрощують розробку кросбраузерного фронтенду програмної системи?
9. Проведіть огляд основних класів метрик ефективності фронтенду.
10. Які прийоми запобігання поганих метричних показників?
11. Які інструменти для вимірювання метрик ефективності фронтенду?
12. Яка роль фремворків та бібліотек для розробки фронтенду у покращенні метрик його ефективності?

ЛІТЕРАТУРА: [7-10, 19, 20].

Додаток 1. Варіанти завдань для виконання лабораторних робіт 1-8.

№ варіанту	Тема
1	Інтернет-магазин (одяг, меблі, телефони, оргтехніка тощо)
2	Сайт Tech start-up
3	Новинний портал
4	Веб-портал для навчання дітей (ігри, розваги, тощо)
5	Інформаційний сайт районної адміністрації міста або області
6	Промо-сайт корпорації
7	Тематична он-лайн енциклопедія (вибір предметної області за здобувачем)
8	Веб-ресурс з надання он-лайн консультацій лікаря, психолога, дизайнера, тощо
9	Сервісний сайт із пошиття одягу, ремонту авто, ремонту оргтехніки, виготовлення виробів, тощо
10	Веб-портал для вивчення іноземних мов, мов програмування, тощо

Додаток 2. Перелік вимог згідно варіанту до лабораторної роботи 1

№ варіанту	Вимоги
1	1. Користувач має можливість відвідати головну сторінку, де надана наступна інформація: • Назва інтернет-магазину та його логотип. • Розділ з описанням мети створення інтернет-магазину та список (перелік) декількох завдань (не менше 3-х нумерованих малими римськими) для досягнення мети. • Загальну інформацію про тип товару, що реалізується, у вигляді таблиці. • Навігаційне меню з кнопками «Каталог», «Відгуки про товар» та «Зворотній зв'язок» • Адреса, контактні телефони магазину. • Кнопка «Перейти у власний кабінет» • Кнопка «Зареєструватись» 2. Користувач має можливість відвідати сторінку «Каталог», де надана наступна інформація: • Назва інтернет-магазину та його логотип. • Список товарів (до 10 шт). • Інформація про кожен товар: ілюстративне фото, назва товару, країна походження, перелік 5 основних характеристик. • Кнопка для переходу на Головну сторінку. 3. Користувач має можливість відвідати сторінку «Зворотній зв'язок», де надана можливість ввести наступну інформацію: 1. Прізвище та ім'я. 2. E-mail. 3. Вік. 4. Освіту (вибрати одну з переліку: повна, неповна, вища, професійна) 5. Користувач має можливість вибрати з випадального списку «Мета зворотнього зв'язку» - співпраця, скарга на порушення права власності, пропозиція, наявність помилки. 6. Користувач має можливість ввести текст у поле «Детально» яке має обмеження на висоту, ширину і кількість символів. 7. Кнопки з відправленням користувачем наданої у п.1-6 інформації 8. Поле зі згодою на обробку інформації. 9. Поля 1, 2, 3 та 4 згруповані, обведені рамкою з легендою «Персональна інформація».

№ варіанту	Вимоги
2	1. Користувач має можливість відвідати головну сторінку, де надана наступна інформація: • Назва Tech start-up та його логотип. • Розділ з описанням мети створення Tech start-up та список (перелік) декількох завдань (не менше 3-х нумерованих) для досягнення мети. • Загальну інформацію про засновників у вигляді таблиці, де є інформація про них. • Навігаційне меню з кнопками «Проекти», «Відгуки про нас» та «Зворотній зв'язок» • Адреса, контактні телефони. • Кнопка «Перейти у власний кабінет» • Кнопка «Зареєструватись» 2. Користувач має можливість відвідати сторінку «Проекти», де надана наступна інформація: • Назва Tech start-up та його логотип. • Список проєктів (до 10 шт). • Інформація про кожен проєкт: ілюстративне фото, назва, країна, перелік 5 основних характеристик. • Кнопка для переходу на Головну сторінку. 3. Користувач має можливість відвідати сторінку «Зворотній зв'язок», де надана можливість ввести наступну інформацію: 1. Прізвище та ім'я. 2. E-mail. 3. Вік. 4. Освіту (вибрати одну з переліку: повна, неповна, вища, професійна) 5. Інтереси (вибрати декілька з переліку: науковець, інвестор, промоутер, держслужбовець) 6. Користувач має можливість вибрати з випадючого списку «Мета зворотнього зв'язку» - співпраця, скарга на порушення права власності, пропозиція, наявність помилки. 7. Користувач має можливість ввести текст у поле «Детально» яке має обмеження на висоту, ширину і кількість символів. 8. Кнопки з відправленням користувачем наданої у п.1-7 інформації. 9. Поле зі згодою на обробку інформації. 10. Поля 1, 2, 3, 4 та 5 згруповані, обведені рамкою з легендою «Персональна інформація».

№ варіанту	Вимоги
3	1. Користувач має можливість відвідати Головну сторінку, де надана наступна інформація: • Назва новинного порталу та його логотип. • Розділ з описанням мети створення порталу та список (перелік) декількох (не менше 3-х нумерованих великими літерами латинського алфавіту) завдань для досягнення мети. • Загальну інформацію про працівників (редакторів, маркетологів, тощо) та джерела інформації у вигляді таблиці. • Інформацію про головного редактора: фото, біографія, освіта. • Навігаційне меню з кнопками «Гарячі новини», «Новини культури», «Новини економіки» та «Зворотній зв'язок» • Адреса, контактні телефони. 2. Користувач має можливість відвідати сторінку «Новини культури», де надана наступна інформація: • Назва порталу та його логотип. • Список новин (до 10 шт). • Інформація про кожну новину: ілюстративне фото, коротке повідомлення, автор. • Кнопка для переходу на «Головну сторінку». 3. Користувач має можливість відвідати сторінку «Зворотній зв'язок», де надана можливість ввести наступну інформацію: 1. Прізвище та ім'я. 2. E-mail. 3. Вік. 4. Освіту (вибрати одну з переліку: повна, неповна, вища, професійна) 5. Користувач має можливість вибрати з випадаючого списку «мета зворотнього зв'язку» - співпраця, скарга на порушення права власності, пропозиція, наявність помилки. 6. Користувач має можливість ввести текст у поле «Детально» яке має обмеження на висоту, ширину і кількість символів. 7. Кнопки з відправленням користувачем наданої у п.1-6 інформації 8. Поле зі згодою на обробку інформації. 9. Поля 1, 2, 3 та 4 згруповані, обведені рамкою з легендою «Персональна інформація».

№ варіанту	Вимоги
4	1. Користувач має можливість відвідати головну сторінку, де надана наступна інформація: • Назва веб-порталу для навчання дітей та його логотип. • Розділ з описанням мети створення веб-порталу та список (перелік) декількох (не менше 3-х нумерованих) завдань для досягнення мети. • Загальну інформацію про те чому навчають, список тем для вивчення у вигляді таблиці. • Навігаційне меню з кнопками «Навчальні матеріали» та «Зворотній зв'язок» • Адреса, контактні телефони. • Кнопка «Перейти у власний кабінет» • Кнопка «Зареєструватись» 2. Користувач має можливість відвідати сторінку «Наші курси», де надана наступна інформація: • Назва веб-порталу та його логотип. • Список курсів (до 10 шт). • Інформація про кожен курс: ілюстративне фото, назва курсу, назви уроків, тип завдань. • Кнопка для переходу на Головну сторінку. 3. Користувач має можливість відвідати сторінку «Зворотній зв'язок», де надана можливість ввести наступну інформацію: • Прізвище та ім'я батьків. • E-mail батьків. • Вік дитини • Стать дитини. • Користувач має можливість вибрати з випадального списку «Мета зворотнього зв'язку» - співпраця, скарга на неякісний контент, пропозиція, наявність помилки. • Користувач має можливість ввести текст у поле «Детально», яке має обмеження на висоту, ширину і кількість символів. • Кнопки з відправленням користувачем наданої у п.1-6 інформації • Поля 1, 2, 3 та 4 згруповані, обведені рамкою з легендою «Персональна інформація». • Поле зі згодою на обробку інформації.

№ варіанту	Вимоги
5	1. Користувач має можливість відвідати Головну сторінку, де надана наступна інформація: • Назва державної установи та її логотип. • Розділ з описанням мети створення державної установи, список (перелік) декількох (не менше 3-х нумерованих малими літерами латинського алфавіту) завдань для досягнення мети та місце у держустрій. • Загальну інформацію про керівників установи та інформацію про них у вигляді таблиці, де є прізвище та ініціали, досвід та освіта. • Навігаційне меню з кнопками «Напрями діяльності» та «Зворотній зв'язок» • Адреса, контактні телефони. • Кнопка «Перейти у власний кабінет» • Кнопка «Зареєструватись». 2. Користувач має можливість відвідати сторінку «Напрями діяльності», де надана наступна інформація: • Назва державної установи та її логотип. • Список напрямів (до 10 шт). • Інформація про кожен напрям: ілюстративне фото, коротке повідомлення, зв'язок з державними органами влади. • Кнопка для переходу на «Головну сторінку». 3. Користувач має можливість відвідати сторінку «Зворотній зв'язок», де надана можливість ввести наступну інформацію: • Прізвище та ім'я. • E-mail. • Вік. • Освіту (вибрати одну з переліку: повна, неповна, вища, професійна) • Користувач має можливість вибрати з випадального списку «мета зворотнього зв'язку» - співпраця, скарга на порушення права власності, пропозиція, наявність помилки. • Користувач має можливість ввести текст у поле «Детально» яке має обмеження на висоту, ширину і кількість символів. • Кнопки з відправленням користувачем наданої у п.1-6 інформації • Поле зі згодою на обробку інформації. • Поля 1, 2, 3 та 4 згруповані, обведені рамкою з легендою «Персональна інформація».

№ варіанту	Вимоги
6	1. Користувач має можливість відвідати Головну сторінку, де надана наступна інформація: • Назва корпорації та її логотип. • Розділ з описанням мети створення корпорації та список (перелік) декількох (не менше 3-х нумерованих великими римськими) завдань для досягнення мети. • Загальну інформацію про засновників корпорації та інформацію про них у вигляді таблиці, де є прізвище та ініціали, досвід та освіта. • Навігаційне меню з кнопками «Напрями діяльності» та «Зворотній зв'язок» • Адреса, контактні телефони. 2. Користувач має можливість відвідати сторінку «Напрями діяльності», де надана наступна інформація: • Назва корпорації та її логотип. • Список напрямів (до 10 шт). • Інформація про кожен напрям: ілюстративне фото, коротке повідомлення, зв'язок з державними органами. • Кнопка для переходу на «Головну сторінку». 3. Користувач має можливість відвідати сторінку «Зворотній зв'язок», де надана можливість ввести наступну інформацію: 1. Прізвище та ім'я. 2. E-mail. 3. Вік. 4. Освіту (вибрати одну з переліку: повна, неповна, вища, професійна) 5. Користувач має можливість вибрати з випадаючого списку «мета зворотнього зв'язку» - співпраця, скарга на порушення права власності, пропозиція, наявність помилки. 6. Користувач має можливість ввести текст у поле «Детально» яке має обмеження на висоту, ширину і кількість символів. 7. Кнопки з відправленням користувачем наданої у п.1-6 інформації 8. Поле зі згодою на обробку інформації. 9. Поля 1, 2, 3 та 4 згруповані, обведені рамкою з легендою «Персональна інформація».

№ варіанту	Вимоги
7	1. Користувач має можливість відвідати Головну сторінку, де надана наступна інформація: • Назва тематичної он-лайн енциклопедії та його логотип. • Розділ з описанням мети створення енциклопедії та список (перелік) декількох (не менше 3-х нумерованих) завдань для досягнення мети. • Загальну інформацію про предметну область (тематику) та авторів енциклопедії у вигляді таблиці. • Навігаційне меню з кнопками «Розділ про SOME1», «Розділ про SOME2» та «Зворотній зв'язок» • Адреса, контактні телефони. 2. Користувач має можливість відвідати сторінку «Розділ про SOME1», де надана наступна інформація: • Назва тематичної он-лайн енциклопедії та її логотип. • Список лотів про SOME1 (до 10 шт). • Інформація про кожен дот: ілюстративне фото, коротке повідомлення, автор. • Кнопка для переходу на «Головну сторінку». 3. Користувач має можливість відвідати сторінку «Зворотній зв'язок», де надана можливість ввести наступну інформацію: 1. Прізвище та ім'я. 2. E-mail. 3. Вік. 4. Освіту (вибрати одну з переліку: повна, неповна, вища, професійна) 5. Користувач має можливість вибрати з випадаючого списку «мета зворотнього зв'язку» - співпраця, скарга на порушення права власності, пропозиція, наявність помилки. 6. Користувач має можливість ввести текст у поле «Детально» яке має обмеження на висоту, ширину і кількість символів. 7. Кнопки з відправленням користувачем наданої у п.1-6 інформації 8. Поле зі згодою на обробку інформації. 9. Поля 1, 2, 3 та 4 згруповані, обведені рамкою з легендою «Персональна інформація».

№ варіанту	Вимоги
8	1. Користувач має можливість відвідати Головну сторінку, де надана наступна інформація: • Назва он-лайн консультацій та їх логотип. • Розділ з описанням мети створення, список (перелік) декількох (не менше 3-х нумерованих арабськими цифрами) завдань для досягнення мети та її місце у сфері надання послуг. • Загальну інформацію про історію виникнення, керівників установи та інформацію про них у вигляді таблиці, де є прізвище та ініціали, досвід та освіта. • Навігаційне меню з кнопками «Послуги» та «Зворотній зв'язок» • Адреса, контактні телефони. • Кнопка «Перейти у власний кабінет» • Кнопка «Зареєструватись» 2. Користувач має можливість відвідати сторінку «Послуги», де надана наступна інформація: • Назва он-лайн консультацій та їх логотип. • Список послуг (до 10 шт). • Інформація про кожну послугу: ілюстративне фото, короткий опис послуги, вартість. • Кнопка для переходу на «Головну сторінку». 3. Користувач має можливість відвідати сторінку «Зворотній зв'язок», де надана можливість ввести наступну інформацію: 1. Прізвище та ім'я. 2. E-mail. 3. Вік 4. Користувач має можливість вибрати з випадючого списку «мета зворотнього зв'язку» - співпраця, скарга на якість послуги, пропозиція, наявність помилки на сайті. 5. Користувач має можливість ввести текст у поле «Детально» яке має обмеження на висоту, ширину і кількість символів. 6. Кнопки з відправленням користувачем наданої у п.1-5 інформації 7. Поле зі згодою на обробку інформації. 8. Поля 1, 2 та 3 згруповані, обведені рамкою з легендою «Персональна інформація».

№ варіанту	Вимоги
9	1. Користувач має можливість відвідати головну сторінку, де надана наступна інформація: • Назва сервісного сайту та його логотип. • Розділ з описанням мети створення сайту та список (перелік) декількох (не менше 3-х нумерованих великими римськими) завдань для досягнення мети. • Загальну інформацію про типи сервісів у вигляді таблиці. • Навігаційне меню з кнопками «Каталог сервісів», «Відгуки про сервіси» та «Зворотній зв'язок» • Адреса, контактні телефони. • Кнопка «Перейти у власний кабінет» • Кнопка «Зареєструватись» 2. Користувач має можливість відвідати сторінку «Каталог сервісів», де надана наступна інформація: • Назва сервісного сайту та його логотип. • Список сервісів (до 10 шт). • Інформація про кожен сервіс: ілюстративне фото, назва товару, перелік 5 основних характеристик. • Кнопка для переходу на Головну сторінку. 3. Користувач має можливість відвідати сторінку «Зворотній зв'язок», де надана можливість ввести наступну інформацію: 1. Прізвище та ім'я. 2. E-mail. 3. Вік. 4. Користувач має можливість вибрати з випадючого списку «Мета зворотнього зв'язку» - співпраця, скарга на неякісний пошив, пропозиція, наявність помилки. 5. Користувач має можливість ввести текст у поле «Детально» яке має обмеження на висоту, ширину і кількість символів. 6. Кнопки з відправленням користувачем наданої у п.1-6 інформації 7. Поле зі згодою на обробку інформації. 8. Поля 1, 2 та 3 згруповані, обведені рамкою з легендою «Персональна інформація».

№ варіанту	Вимоги
10	1. Користувач має можливість відвідати головну сторінку, де надана наступна інформація: • Назва веб-порталу та його логотип. • Розділ з описанням мети створення веб-порталу та список (перелік) декількох (не менше 3-х нумерованих великими літерами латинського алфавіту) завдань для досягнення мети. • Загальну інформацію про мову, список тем для вивчення у вигляді таблиці. • Навігаційне меню з кнопками «Навчальні теми» та «Зворотній зв'язок» • Адреса, контактні телефони. • Кнопка «Перейти у власний кабінет» • Кнопка «Зареєструватись» 2. Користувач має можливість відвідати сторінку «Навчальні теми», де надана наступна інформація: • Назва веб-порталу та його логотип. • Список тем (до 10 шт). • Інформація про кожну тему: ілюстративне фото, назва теми, назви уроків, тип завдань. • Кнопка для переходу на Головну сторінку. 3. Користувач має можливість відвідати сторінку «Зворотній зв'язок», де надана можливість ввести наступну інформацію: • Прізвище та ім'я. • E-mail. • Вік. • Чи вже вивчаєте мову *? (вибрати одну з переліку: так, я профі; так, але знаю початки; планую; ні) • Користувач має можливість вибрати з випадального списку «Мета зворотнього зв'язку» - співпраця, скарга на неякісний контент, пропозиція, наявність помилки. • Користувач має можливість ввести текст у поле «Детально», яке має обмеження на висоту, ширину і кількість символів. • Кнопки з відправленням користувачем наданої у п.1-6 інформації • Поле зі згодою на обробку інформації. • Поля 1, 2, 3 та 4 згруповані, обведені рамкою з легендою «Персональна інформація».

Додаток 3. Перелік вимог згідно варіанту до лабораторної роботи 2

№ варіанту	Вимоги
1	1. Головна сторінка: • Блок Назва інтернет-магазину та його логотип абсолютно позиціонований з 0 відступом зверху. • Блок розділу з описанням мети створення інтернет-магазину та список (перелік) декількох завдань для досягнення мети та має більші зовнішні відступи, ніж внутрішні (від умовної рамки блоку) . • При наведенні курсору на блок із Загальною інформацією про тип одягу, що реалізується, у вигляді таблиці, він зміщується вгору на 2% від його величини. • При виділенні користувачем тексту адреси та контактних телефонів шрифт і фон через 2 сек підсилюється на 10% від заданих. • Перед описом мети створення інтернет-магазину вивести контент з привітним емодзі. 2. Сторінку «Каталог»: • Блок Назва інтернет-магазину та його логотип фіксований зверху. • Інформація про кожен товар взята у рамку темно-зеленого кольору, у якій верхня межа є крапковою, решта меж суцільні та тонкі, з наявною тінню в середину рамки. • Текст, що відповідає назві товару, має більшу вагу і висоту гліфів шрифту, ніж текст опис, та є похилим. • При наведенні курсора на будь-яке ілюстративне фото, воно анімується за допомогою зміни до «імітації старовинного зображення». • Вже відвідані посилання на даній сторінці мають темно-зелений колір шрифту.
2	1. Головна сторінка: • Блок Назва Tech start-up та його логотип абсолютно позиціонований з 0 відступом зверху. • Блок Розділу з описанням мети створення Tech start-up та список (перелік) декількох завдань для досягнення мети та немає зовнішніх (сусідні блоки) відступи, але має внутрішні (від умовної рамки блоку). • Під час фокусування на блоці із Загальною інформацією про засновників у вигляді таблиці, де є інформація про них, він зміщується праворуч на 2% від величини блоку. • При виділенні користувачем тексту адреси та контактних телефонів шрифт і фон через 2 сек підсилюється на 10%. • Після опису мети створення Tech start-up вивести контент з hi-tech емодзі. 2. Сторінка «Проекти»: • Блок Назва Tech start-up та його логотип фіксований зверху. • Інформація про кожен проєкт взята у рамку темно-синього кольору, у якій нижня межа є подвійною, решта сторін суцільні середньої товщини. • Текст «назва проєкту» має більшу вагу і висоту гліфів шрифту та є похилим. • При наведенні курсора на будь-яке ілюстративне фото, воно анімується за допомогою зміни до «чорно-білого зображення».

	Посилання за якими користувач ще не перейшов на даній сторінці мають темно-помаранчевий колір шрифту.
3	1. Головна сторінка: - Блок Назва новинного порталу та його логотип абсолютно позиціонований з 0 відступом зверху. - Блок із розділом з описанням мети створення порталу та список (перелік) декількох завдань для досягнення мети та має менші зовнішні (сусідні блоки) відступи, ніж внутрішні (від умовної рамки блоку). - Блок з Інформацією про головного редактора: фото, біографія, освіта при наведенні курсору зміщується на 5 px ліворуч та вгору. - При фокусуванні на блоці з Адресою, контактними телефонами через 3 сек блок стає на 20% більшим. - Перший рядок тексту про головного редактора має курсивний шрифт (реалізуємо псевдо-елементом). 2. Сторінка «Новини культури»: - Блок Назва порталу та його логотип фіксований зверху. - Інформація про кожну новину взята у рамку темно-вишневого кольору, у якій ліва межа є пунктирною, решта меж суцільні та товсті - Текст «автор» має меншу вагу і висоту гліфів шрифту та є похилим. - При наведенні курсора на будь-яке ілюстративне фото, воно анімується за допомогою зміни до «його негативу». - Посилання в момент кліку на ньому користувачем має темно-помаранчевий колір шрифту.
4	1. Головна сторінка: - Блок Назва веб-порталу та його логотип абсолютно позиціонований з 0 відступом зверху. - Блок Розділу з описанням мети створення веб-порталу та список (перелік) декількох завдань для досягнення мети та має зовнішні (сусідні блоки) відступи, але немає внутрішніх (від умовної рамки блоку). - Блок із Загальною інформацією про навчання, список тем для вивчення у вигляді таблиці, при наведенні курсору зміщується праворуч на 2% широти області перегляду. - При фокусуванні на блоці з Адресою, контактними телефонами через 2 сек у блоку з'являється тінь. - Перша літера тексту інформації про те чому навчають має тінь та темно-зелений колір (реалізуємо псевдо-елементом). 2. Сторінка «Наші курси»: - Блок Назва веб-порталу та його логотип фіксований зверху. - Інформація про кожну тему взята у рамку темно-фіолетового кольору, у якій є права межа суцільної тонкої товщини та ліва – пунктирна. - Текст «назва теми» має більшу вагу і висоту гліфів шрифту та є похилим. - При наведенні курсора на будь-яке ілюстративне фото, воно анімується за допомогою зростання насиченості його кольорів у 1,5 рази (150%). - Вже відвідані посилання на даній сторінці мають темно-червоний колір шрифту.

5	1. Головна сторінка: • Блок Назва державної установи та її логотип абсолютно позиціонований з 0 відступом зверху. • Блок Розділу з описанням мети створення державної установи, список (перелік) декількох завдань для досягнення мети та місце у держустройі має пропорційні зовнішні (сусідні блоки) та внутрішні (від умовної рамки блоку) відступи. • Блок із Загальною інформацією про керівників установи при наведенні курсору масштабується на 5%. • При фокусуванні на Адресі, контактних телефонах через 3 сек з'являється подвійна рамка чорного кольору навколо тексту. • Перед описом мети створення вивести контент з привітним емодзі. 2. Сторінка «Напрями діяльності»: • Блок Назва державної установи та її логотип фіксований зверху. • Інформація про кожен напрям взятий у рамку темно-вишневого кольору, у якій ліва межа є суцільною та товстою, решта меж є пунктирними • Текст «зв'язок з державними органами влади» має меншу вагу і висоту гліфів шрифту та є похилим. • При наведенні курсора на будь-яке ілюстративне зображення, воно анімується за допомогою зміни яскравості від 1 до 0.75. • Посилання за якими користувач ще не перейшов на даній сторінці мають темно-зелений колір шрифту
6	1. Головна сторінка: • Блок Назва корпорації та її логотип абсолютно позиціонований з 0 відступом зверху. • Блок Розділу з описанням мети створення корпорації та список (перелік) декількох завдань для досягнення мети та має зовнішні (сусідні блоки) відступи, але немає внутрішніх (від умовної рамки блоку). • Блок із Загальною інформацією про засновників корпорації та інформацію про них у вигляді таблиці, де є прізвище та ініціали, досвід та освіта, при наведенні курсору масштабується на 5%. • При наведенні курсору на Адресу, контактні телефони через 2 сек з'являється тінь від літер тексту. • Після опису мети створення корпорації вивести контент з емодзі-директором 2. Сторінка «Напрями діяльності»: • Блок Назва корпорації та її логотип фіксований зверху. • Інформація про кожен напрям взятий у рамку темно-помаранчевого кольору, у якій права межа є суцільною лінією середньої товщини, решта сторін подвійні. • Текст «зв'язок з державними органами» має меншу вагу і висоту гліфів шрифту та є похилим. • При фокусуванні на будь-якому ілюстративному фото, воно анімується за допомогою зростання насиченості його кольорів у двічі (200%). • Посилання в момент кліку на ньому користувачем має темно-зелений колір шрифту

7	1. Головна сторінка: • Блок Назва тематичної он-лайн енциклопедії та його логотип абсолютно позиціонований з 0 відступом зверху. • Блок Розділу з описанням мети створення енциклопедії та список (перелік) декількох завдань для досягнення мети та має більші зовнішні (сусідні блоки) відступи, ніж внутрішні (від умовної рамки блоку). • Блок із Загальною інформацією про предметну область (тематику) та засновників енциклопедії при наведенні курсора зміщується вниз на 3% від його величини. • При фокусуванні на Адресі, контактних телефонах через 3 сек фон заливається лінійним градієнтом від світло-зеленого до зеленого. • Перший рядок тексту опису мети має курсивний шрифт (реалізуємо псевдо-елементом). 2. Сторінка «Розділ про SOME1»: • Блок Назва тематичної он-лайн енциклопедії та її логотип фіксований зверху. • Інформація про кожен лот взята у рамку темно-синього кольору, у якій нижня межа є суцільною середньої товщини, решта сторін є подвійними: ілюстративне фото, коротке повідомлення, автор. • Текст «автор» має меншу вагу і висоту гліфів шрифту та є похилим. • При наведенні курсора на будь-яке ілюстративне фото, воно анімується за допомогою зміни до «імітації старовинного зображення». • Вже відвідані посилання на даній сторінці мають насичено-жовтий колір шрифту.
8	1. Головна сторінка: • Блок назва Он-лайн консультацій та їх логотип абсолютно позиціонований з 0 відступом зверху. • Блок Розділу з описанням мети створення Он-лайн консультацій і, список (перелік) декількох завдань для досягнення мети та її місце у сфері надання послуг та має менші зовнішні (сусідні блоки) відступи, ніж внутрішні (від умовної рамки блоку). • Перша літера тексту з описанням мети має тінь та світло-фіолетовий колір (реалізуємо псевдо-елементом) • Блок із Загальною інформацією про історію виникнення, керівників та інформацію про них у вигляді таблиці при наведенні курсору зміщується на 5 px ліворуч та вниз. • При наведенні курсору на Адресу, контактні телефони через 2 сек шрифт тексту змінюється на червоний напівжирний. 2. Сторінка «Послуги»: • Блок назва Он-лайн консультацій та їх логотип фіксований зверху. • Інформація про кожну послугу взята у рамку темно-голубого кольору, у якій верхня межа є суцільна та тонка, решта меж є крапковими: ілюстративне фото, короткий опис послуги, вартість. • Текст «вартість послуги» має більшу вагу і висоту гліфів шрифту та є похилим. • При наведенні курсора на будь-яке ілюстративне фото, воно анімується за допомогою зміни до «його негативу».

	Посилання за якими користувач ще не перейшов на даній сторінці мають світло-зелений колір шрифту
9	1. Головна сторінка: - Блок сервісного сайту та його логотип абсолютно позиціонований з 0 відступом зверху. - Блок Розділу з описанням мети створення сервісного сайту та список (перелік) декількох завдань для досягнення мети та немає зовнішніх (сусідні блоки) відступи, але має внутрішні (від умовної рамки блоку). - Перед описом мети створення вивести контент з емоді, яке ілюструє сервіс. - Блок із Загальною інформацією про тип сервісів, що можливо замовити на пошив, у вигляді таблиці, при наведенні курсору зміщується вниз на 2% висоти області перегляду. - При наведенні курсору на Адресу, контактні телефони ательє через 2 сек розмір тексту збільшується на 20%. 2. Сторінка «Каталог»: - Блок сервісного сайту та його логотип фіксований зверху. - Інформація про кожен товар взята у рамку темно-жовтого кольору, у якій є нижня межа суцільної середньої товщини та верхня – крапкова - Текст «тип тканини для пошиття» має більшу вагу і висоту гліфів шрифту та є похилим - При фокусуванні на будь-якому ілюстративному фото, воно анімується за допомогою зміни до «чорно-білого зображення». - Посилання в момент кліку на ньому користувачем має темно-зелений колір шрифту
10	1. Головна сторінка: - Блок Назва веб-порталу та його логотип абсолютно позиціонований з 0 відступом зверху. - Блок Розділу з описанням мети створення веб-порталу та список (перелік) декількох завдань для досягнення мети та має зовнішні (сусідні блоки) відступи, але немає внутрішніх (від умовної рамки блоку). - Після опису мети створення вивести контент з емоді-учня. - Блок із Загальною інформацією про мову, список тем для вивчення у вигляді таблиці, при наведенні курсору зміщується праворуч на 2% широти області перегляду. - При наведенні курсору на Адресу, контактні телефони через 2 сек у блоку з'являється тінь. 2. Сторінка «Навчальні теми»: - Блок Назва веб-порталу та його логотип фіксований зверху. - Інформація про кожну тему взята у рамку темно-фіолетового кольору, у якій є права межа суцільної тонкої товщини та ліва – пунктирна. - Текст «назва теми» має більшу вагу і висоту гліфів шрифту та є похилим. - При наведенні курсора на будь-яке ілюстративне фото, воно анімується за допомогою зростання насиченості його кольорів у 1,5 рази (150%). - Вже відвідані посилання на даній сторінці мають насичено-жовтий колір шрифту.

Додаток 4. Перелік вимог згідно варіанту до лабораторної роботи 3

№ варіанту	Вимоги
1	1. Створити 10 записів товарів, які містять артикул товару від виробника (літери та цифри), внутрішній id, назву товару, країну походження, вартість1_недіюча, вартість2_недіюча, вартість3_діюча, кількість на складі. Реалізувати програмно: • впорядкування товарів за країною походження та знаходження середньої вартості товару, з однієї і тієї ж країни. • знаходження товару, який має мінімальну «вартість2_недіюча» та вивід внутрішнього id, що йому відповідає. • додавання нового запису до 10 записів: якщо у нього відсутня хоч якась інфо то додаємо в кінець списку товарів; якщо вся інфо наявна, то на початок списку товарів. • створити список, який містить динаміку зміни ціни на кожен товару (за його id або назвою), виходячи з наявних цін, за правилом якщо ціни утворюють спадну (зростаючу) послідовність, то товар «стабільно дешевшає» («стабільно дорожчає»), у протилежному випадку товар «не стабільні зміни». Вказівка: використовуємо тип даних масив, методи роботи з ними та функції. 2. Створити 10 облікових записів користувачів, які містять прізвище, ім'я, вік, освіту, мету зворотнього зв'язку, дату звернення, час звернення. Реалізувати програмно: • виведення переліку користувачів, який звернулись продовж певного місяця, в деякий заданий момент часу. • вивести середній вік користувачів і якщо він співпадає із віком деякого користувача, то надати його персональні дані про освіту. • поділити користувачів за часом звернення та метою на три класи – «звернулись з претензією у робочий час» (з 09:00 до 18:00), «звернулись з побажанням у робочий час» та «Інші». Знайти кількість користувачів у кожному з класів. • забезпечити сортування користувачів у порядку спадання віку та в алфавітному порядку. Вказівка: використовуємо об'єктно-орієнтований js-код.

№ варіанту	Вимоги
2	1. Створити 10 записів проєктів, що містять їх внутрішній id, назву проєкту, країну реалізації проєкту, кількість осіб у команді, що його реалізує, тривалість проєкту, вартість проєкту. Реалізувати програмно: • впорядкування проєктів за країною реалізації та знаходження середньої вартості проєкту, який реалізує одна й та ж команда. • знаходження проєкту, який має максимальну кількість осіб у команді, та вивід його внутрішнього id. • додавання нового проєкту до 10 записів: якщо у ньому відсутня хоч якась інфо, то додаємо в кінець списку; якщо вся інфо є, то шукаємо його місце у порядку зростання вартості. • обчислити нову вартість проєкту за принципом: якщо існує принаймні один проєкт, що відрізняється від даного, країни реалізації якого належать одному континенту з більшою кількістю осіб у команді та меншою вартістю, то вартість зростає у 1,4 рази; якщо існує принаймні один проєкт, що відрізняється від даного, країни реалізації якого належать різним континентам з більшою кількістю осіб у команді та меншою вартістю, то вартість зростає у 1,6 разів; якщо існує принаймні один проєкт, що відрізняється від даного, країни реалізації якого належать різним континентам з більшою кількістю осіб у команді та більшою вартістю, то вартість зростає у 2 рази; якщо існує принаймні один проєкт, що відрізняється від даного, країни реалізації якого співпадають та меншою вартістю, то вартість спадає у 0,8 рази; в усіх інших випадках вартість не змінна. Надати оновлений список нових вартостей проєктів – назва проєкту та вартість. Вказівка: використовуємо тип даних масив, методи роботи з ними та функції. 2. Створити 10 облікових записів користувачів, які містять прізвище, ім'я, вік, освіту, кількість символів у полі детально, дату звернення, час звернення. Реалізувати програмно: • виведення переліку користувачів, який звернулись продовж певного місяця, в деякий заданий момент часу. • вивести максимальну довжину символів у полі детально і вік користувача, що йому відповідає, та дані про освіту. • поділити користувачів за датою звернення та віком на три класи – «бізнес-активні середнього віку» (вік від 35 до 55 років, місяці звернення - осінь, зима, весна), «бізнес-пасивні похилого» (літо) та «Інші». Знайти кількість користувачів у кожному з класів. • забезпечити сортування користувачів алфавітному порядку із виведенням приблизного року народження. Вказівка: використовуємо об'єктно-орієнтований js-код. Прізвище та ім'я.

№ варіанту	Вимоги
3	1. Створити 10 записів новин, що містять внутрішній id новини, назву новини, автора, кількість символів у короткому повідомленні, кількість користувачів, що прочитали новину за добу_1, кількість користувачів, що прочитали новину за добу_2, час публікації новини вперше (дата, год і хв). Реалізувати програмно: • впорядкування новин за автором та знаходження середньої довжини повідомлення у символах, одного й того ж автора. • знаходження новини, який має максимальну кількість користувачів, що прочитали новину за добу_1 та вивід її внутрішнього id. • додавання нової новини до 10 записів: якщо у ній відсутня хоч якась інфо, то додаємо в кінець списку; якщо вся інфо є, то на початок списку товарів. • обчислити суму оплати роботи автора за добу за принципом: 1 символ у новині, яка з'явилась у робочий час доби (з 10:00 до 18:00) дорівнює 0,5 грн; решта часу – 0,8 грн. Вказівка: використовуємо тип даних масив, методи роботи з ними та функції. 2. Створити 10 облікових записів користувачів, які містять прізвище, ім'я, вік, освіту, мету зворотнього зв'язку, дату звернення, час звернення. Реалізувати програмно: • виведення переліку користувачів, який звернулись продовж певного місяця, в деякий заданий момент часу. • вивести мінімальний вік користувача і якщо він співпадає із віком деякого користувача, то надати його дані про освіту, як типові. • поділити користувачів за прізвищем та освітою на три класи – «Жінки з вищою освітою», «Чоловіки без освіти» та «Інші». Знайти кількість користувачів у кожному з класів. • забезпечити сортування користувачів у порядку спадання віку та в алфавітному порядку. Вказівка: використовуємо об'єктно-орієнтований js-код.

№ варіанту	Вимоги
4	1. Створити 10 записів курсів, що містять внутрішній id , назву, автор, викладачів, тип завдань, кількість користувачів, що прочитали про курс за добу_1, кількість користувачів, що прочитали про курс за добу_2, тривалість вивчення (год, хв). Реалізувати програмно: • впорядкування курсів за тривалістю вивчення та знаходження середньої кількості користувачів, що нею цікавляться. • знаходження курсу, яка має мінімальну кількість користувачів, що прочитали про неї за добу_2 та вивід його внутрішнього id. • додавання нової курсу до 10 записів: якщо у ньому відсутня хоч якась інфо, то додаємо на початок списку; якщо вся інфо є, то шукаємо місце у списку, що впорядкований за автором. • обчислити тривалість вивчення декількох курсів одночасно за принципом: якщо вивчаєш до 3 курсів, то тривалість вивчення кожної з них не змінюється, в іншому випадку тривалість вивчення кожного з них збільшується у 1,5 рази. Вказівка: використовуємо тип даних масив, методи роботи з ними та функції. 2. Створити 10 облікових записів дітей, які містять прізвище, ім'я, вік, E-mail, мету зворотнього зв'язку, дату звернення, час звернення. Реалізувати програмно: • виведення переліку дітей, батьки яких звернулись продовж певного місяця, в деякий заданий момент часу. • вивести мінімальний вік дитини, E-mail, який їй відповідає, та дату звернення. • поділити дітей за віком на класи – «ранній (до 3-х років включно)», «середній (від 3 до 5)», «дошкільний (5-7 років)» та шкільний (більше 7 років) та знайти кількість дітей у кожному з класів. • забезпечити сортування дітей алфавітному порядку E-mail із виведенням мети зворотнього зв'язку. Вказівка: використовуємо об'єктно-орієнтований js-код.

№ варіанту	Вимоги
5	1. Створити 10 записів напрямів діяльності, що містять внутрішній id напрям, назву напрям, назву відділу, що його реалізує, текст про зв'язок з державними органами, кількість користувачів, що прочитали про напрям за добу_1, кількість користувачів, що прочитали про напрям за добу_2, терміни реалізації у деяких одиницях (кількість років, місяців, тощо). Реалізувати програмно: • впорядкування напрямів за терміном реалізації та знаходження середньої кількості користувачів, що читають про ті з них, які мають один і той самий термін. • знаходження напрям, який має максимальну кількість користувачів, що прочитали про нього за добу_1 та вивід назви відділу, що його реалізує. • додавання нового напрям до 10 записів: якщо у нього відсутня хоч якась інфо, то додаємо в кінець списку; якщо вся інфо є, то на початок списку. • обчислити тривалість реалізації декількох напрямів одночасно, відповідним йому відділом, за принципом: якщо 1 напрям, то тривалість дорівнює вказаній, додавання напрям у роботу проєкту збільшує його тривалість на 1,1 від вказаної у комірці «терміни реалізації у деяких одиницях (кількість років, місяців, тощо)». Вказівка: використовуємо тип даних масив, методи роботи з ними та функції. 2. Створити 10 облікових записів користувачів, які містять прізвище, ім'я, вік, освіту, мету зворотнього зв'язку, дату звернення, час звернення. Реалізувати програмно: • виведення переліку користувачів, який звернулись продовж певного місяця, в деякий заданий момент часу. • вивести мінімальний вік користувача і та надати його дані про освіту. • поділити користувачів за часом звернення та освітою на три класи – «з вищою освітою у робочий час (09:00 до 17:00)», «без освіти у неробочий час» та «інші». Знайти кількість користувачів у кожному з класів. • забезпечити сортування користувачів алфавітному порядку із виведенням мети зворотнього зв'язку. Вказівка: використовуємо об'єктно-орієнтований js-код.

№ варіанту	Вимоги
6	1. Створити 10 записів напрямів діяльності, що містять внутрішній id напрям, назву напрям, назву відділу, що його реалізує, текст про зв'язок з державними органами, кількість користувачів, що прочитали про напрям за добу_1, кількість користувачів, що прочитали про напрям за добу_2, тривалість послуги у деяких одиницях (кількість років, місяців, тощо). Реалізувати програмно: • впорядкування напрямів за тривалістю послуги та знаходження середньої кількості користувачів, що читають про ті з них, що мають одну і ту ж тривалість. • знаходження напрям, який має мінімальну кількість користувачів, що прочитали про нього за добу_2 та вивід назви відділу, що його реалізує. • додавання нового напрям до 10 записів: якщо у нього відсутня хоч якась інфо, то додаємо в кінець списку; якщо вся інфо є, то шукаємо місце у списку за внутрішнім id. • обчислити тривалість надання декількох послуг одночасно, відповідним до нього відділом, за принципом: якщо 1 послуга, то вказану тривалість множимо на 0,95; якщо до 3 послуг, то вказану тривалість множимо на 0,95, в іншому випадку вказану тривалість кожної послуги множимо на 0,6. Вказівка: використовуємо тип даних масив, методи роботи з ними та функції. 2. Створити 10 облікових записів користувачів, які містять прізвище, ім'я, вік, освіту, мету зворотнього зв'язку, дату звернення, час звернення. Реалізувати програмно: • виведення переліку користувачів, який звернулись продовж певного місяця, в деякий заданий момент часу. • вивести середній вік користувача. • поділити користувачів за часом звернення та віком на три класи – «середній за віком, зі зверненням у робочий час (09:00 до 17:00)», «похилого віку зі зверненням у неробочий час» та «інші». Знайти кількість користувачів у кожному з класів. • забезпечити сортування користувачів алфавітному порядку із виведенням мети зворотнього зв'язку. Вказівка: використовуємо об'єктно-орієнтований js-код.

№ варіанту	Вимоги
7	1. Створити 10 записів, які є складовими в он-лайн енциклопедії та містять внутрішній id, назву, автора, кількість символів у короткому повідомленні, кількість користувачів, що прочитали за добу_1, кількість користувачів, що прочитали за добу_2, час публікації лоту вперше (дата та год і хв). Реалізувати програмно: • впорядкування за часом публікації та знаходження середньої довжини повідомлення у символах, одного й того часу доби. • знаходження тих складових, які має мінімальну кількість користувачів, що прочитали новину за добу_2 та вивід його внутрішнього id. • додавання нового запису до 10 записів: якщо у ньому відсутня хоч якась інфо, то додаємо в кінець списку; якщо вся інфо є, то шукаємо місце у списку у порядку зростання кількості символів у короткому повідомленні. • обчислити суму оплати роботи автора за принципом: 1 символ у новині, яка з'явилась влітку та взимку дорівнює 0,95 грн, у решта часу – 1,1 грн. 2. Створити 10 облікових записів користувачів, які містять прізвище, ім'я, вік, освіту, мету зворотнього зв'язку, дату звернення, час звернення. Реалізувати програмно: • виведення переліку користувачів, який звернулись продовж певного місяця, в деякий заданий момент часу. • вивести максимальний вік користувача, що звернувся, його освіту та дату звернення. • поділити користувачів за часом звернення та метою на три класи – «претензійні сови», «доброзичливі жайворонки» та «інші». Знайти кількість користувачів у кожному з класів. • забезпечити сортування користувачів алфавітному порядку із виведенням мети зворотнього зв'язку. Вказівка: використовуємо об'єктно-орієнтований js-код.

№ варіанту	Вимоги
8	1. Створити 10 записів послуг, що містять внутрішній id послуги, назву послуги, ім'я лікаря, що її проводить, кількість користувачів, що прочитали про послугу за місяць_1, кількість користувачів, що прочитали про послугу за місяць_1, вартість послуги; тривалість послуги у хв. Реалізувати програмно: • впорядкування послуг за вартістю та знаходження середньої тривалості послуги у хв, серед тих, що відповідають одній і тій же вартості. • знаходження послуги, який має максимальну кількість користувачів, що прочитали про неї за місяць_1 та вивід її внутрішнього id. • додавання нової послуги до 10 записів: якщо у ній відсутня хоч якась інфо, то додаємо в кінець списку; якщо вся інфо є, то шукаємо її місце у порядку спадання вартості послуг. • обчислити оновлену вартість послуги на наступний місяць за принципом: якщо кількість користувачів, що прочитали про послугу за місяць_1 менша за кількість користувачів, що прочитали про послугу за місяць_2, то вартість проєкту множимо на 1,15; якщо кількість користувачів, що прочитали про послугу за місяць_1 дорівнює кількості користувачів, що прочитали про послугу за місяць_2, то вартість проєкту не змінюється; якщо кількість користувачів, що прочитали про послугу за місяць_1 більша за кількість користувачів, що прочитали про послугу за місяць_2, то вартість проєкту множимо на 0,95; Надати оновлений список нових вартостей послуг – назва послуги, вартість, ім'я лікаря\психолога\дизайнера. Вказівка: використовуємо тип даних масив, методи роботи з ними та функції. 2. Створити 10 облікових записів користувачів, які містять прізвище, ім'я, вік, освіту, мету зворотнього зв'язку, дату звернення, час звернення. Реалізувати програмно: • виведення переліку користувачів, який звернулись продовж певного місяця, в деякий заданий момент часу. • вивести мінімальний вік користувача, що звернувся, його освіту та ім'я . • поділити користувачів за часом звернення на два класи – «робочий час (09:00 до 17:00)» та «неробочий час» та знайти кількість користувачів у кожному з класів та середній вік користувача у кожному з них. • забезпечити сортування користувачів алфавітному порядку із виведенням мети зворотнього зв'язку. Вказівка: використовуємо об'єктно-орієнтований js-код.

№ варіанту	Вимоги
9	1. Створити 10 записів про сервіси, що містять внутрішній id товару, його назву, тип сервісу, кількість користувачів, що прочитали про нього за добу_1, кількість користувачів, що прочитали про нього за добу_2, терміни тривалості надання сервісу у деяких одиницях (кількість днів, год, тощо), вартість сервісу. Реалізувати програмно: • впорядкування назв сервісів за терміном тривалості його надання та знаходження середньої кількості користувачів, що читають про них, з одним і тим же терміном (і по добах і по кількості усереднюємо). • знаходження сервісу, який має максимальну кількість користувачів, що прочитали про нього за добу_2 та вивід його типу. • додавання нового запису до 10 записів: якщо у нього відсутня хоч якась інфо, то додаємо в кінець списку; якщо вся інфо є, то шукаємо місце у списку, що відсортований у порядку зростання вартості. • обчислити тривалість надання декількох сервісів одночасно за їх типом: якщо 1 сервіс, то тривалість дорівнює вказаній, якщо їх до 3-х, то тривалість кожного множимо на 1,1 від вказаної початкової, у всіх інших випадках тривалість кожного сервісу множимо на 2. Вказівка: використовуємо тип даних масив, методи роботи з ними та функції. 2. Створити 10 облікових записів користувачів, які містять прізвище, ім'я, вік, E-mail, мету зворотнього зв'язку, дату звернення, час звернення. Реалізувати програмно: • виведення переліку користувачів, який звернулись продовж певного місяця, в деякий заданий момент часу. • вивести мінімальний вік користувача, що звернувся, його E-mail та дату звернення. • поділити користувачів на три класи – «активна молодь» (якщо вік до 25, а мета звернення – побажання), «похилий претензійний» (якщо вік більше 60, а мета звернення – претензія), «середній без претензій» (якщо вік менше 60 і більше 25, а мета звернення – не претензія) та «інші». Знайти кількість користувачів у кожному з класів. • забезпечити сортування користувачів алфавітному порядку адреси E-mail із виведенням мети зворотнього зв'язку. Вказівка: використовуємо об'єктно-орієнтований js-код.

№ варіанту	Вимоги
10	1. Створити 10 записів тем, що містять внутрішній id теми, назву теми, автора, тип завдань, кількість користувачів, що прочитали про тему за добу_1, кількість користувачів, що прочитали про тему за добу_2, тривалість вивчення (год, хв). Реалізувати програмно: • впорядкування тем за тривалістю вивчення та знаходження середньої кількості користувачів, що нею цікавляться. • знаходження теми, яка має мінімальну кількість користувачів, що прочитали про неї за добу_2 та вивід її внутрішнього id. • додавання нової теми до 10 записів: якщо у ньому відсутня хоч якась інфо, то додаємо на початок списку; якщо вся інфо є, то шукаємо місце у списку, що впорядкований за автором. • обчислити тривалість вивчення декількох тем одночасно за принципом: якщо вивчаєш до 3 тем, то тривалість вивчення кожної з них не змінюється, в іншому випадку тривалість вивчення кожної з них збільшується у 1,5 рази. Вказівка: використовуємо тип даних масив, методи роботи з ними та функції. 2. Створити 10 облікових записів користувачів, які містять прізвище, ім'я, вік, E-mail, мету зворотнього зв'язку, дату звернення, час звернення. Реалізувати програмно: • виведення переліку користувачів, який звернулись у певний місяць і в деякий заданий момент часу. • вивести мінімальний вік користувача, що звернувся, його E-mail та дату звернення. • поділити користувачів за віком та датою звернення на три класи – «школярі до початку навчального року» (вік до 16 років), «не школярі впродовж семестру» та «інші». Кількість користувачів у кожному з класів. • забезпечити сортування користувачів алфавітному порядку E-mail із виведенням мети зворотнього зв'язку. Вказівка: використовуємо об'єктно-орієнтований js-код.

Додаток 5. Перелік вимог згідно варіанту до лабораторної роботи 4

№ варіанту	Вимоги
1	1. Відвідуючи головну сторінку, користувач на фоні блоку з назвою інтернет магазину та його логотипом може бачити періодично падаючі донизу цього тематичного блоку декілька зображень товарів, які доступні на сторінці Каталог та розмір яких по висоті не перевищує подвоєної висоти великої літери у назві. 2. Користувач має можливість відвідати сторінку «Каталог», де праворуч у назві товару, який нещодавно доданий до каталогу, самодрукується текст «Новинка!» червоним кольором з шириною гліфу у 1,2 рази більше, ніж текст назви товару. 3. На сторінці «Каталог» користувач може здійснити пошук товарів за назвою. 4. На сторінці «Зворотній зв'язок» при наведенні курсора миші користувачем на поле «Детально» воно змінює колір фону, з'являється рамка та тінь від неї, а праворуч нього з'являється текст - підказка (tooltip) «Ваша думка для нас важлива! Конкретизуйте мету звернення, будь ласка». 5. Для користувача, що зайшов на головну сторінку з пристрою, шириною екрану не більше 768 рх, навігаційне меню розташоване вертикально орієнтованим праворуч. В іншому випадку, навігаційне меню розташоване горизонтально орієнтованим зверху.
2	1. Користувач має можливість відвідати Головну сторінку, де нижче назви Tech start-up самодрукується текст з девізом компанії темно-червоним кольором з товщиною гліфу у 1,2 рази більше, ніж текст назви Tech start-up. 2. Користувач має можливість відвідати сторінку «Проекти», де під блоком із назвою та логотипом, по центру розташована карусель із зображень списку проєктів. Висота елементів каруселі не більше 20% від ширини вікна перегляду для користувача. 3. На сторінці «Зворотній зв'язок» при наведенні курсора миші користувачем на поле «Детально», воно змінює колір фону, з'являється рамка та тінь від неї, а праворуч нього з'являється текст - підказка (tooltip) «Вдячні за Ваш час! Конкретизуйте мету звернення, будь ласка». 4. При наведенні курсора користувачем на зображення засновника, з'являється модальне вікно (появляється) із «тематичною загадкою» від нього (формулювання загадки на розсуд здобувача) та полем для вводу відповіді користувача. Забезпечити перевірку правильності відповіді. 5. Для користувача, що зайшов на головну сторінку в період з 21 год до 6 ранку, колір фону всіх елементів темніший на 40% за колір в інший час.

№ варіанту	Вимоги
3	1. Користувач має можливість натискаючи в навігаційному меню на текст «Новини культури» головної сторінки, бачити випадаюче підменю з переліком тематичних розділів. 2. На сторінці «Новини культури» користувач може здійснити пошук новин за назвою. 3. Користувач має можливість відвідати сторінку «Новини культури», де праворуч у назві новини, яка є суспільно важливою через кожні X сек з'являється текст «Увага!» червоним кольором з шириною гліфу у 1,2 рази більше, ніж тест назви новини, який користувач бачить Y сек (точні значення X та Y здобувач обирає сам). 4. На сторінці «Зворотній зв'язок» при наведенні курсора миші користувачем на поле «Детально», воно змінює колір фону, з'являється рамка та тінь від неї, а праворуч нього з'являється текст - підказка (tooltip) «Дякуючи Вам, ми стаємо кращі! Конкретизуйте мету звернення, будь ласка». 5. Для користувача, що зайшов на головну сторінку з пристрою, шириною екрану не більше 768 px, навігаційне меню розташоване вертикально орієнтованим ліворуч. В іншому випадку, навігаційне меню розташоване горизонтально орієнтованим зверху.
4	1. Відвідуючи головну сторінку, користувач на фоні блоку з назвою веб-порталу для навчання дітей та його логотипом може бачити періодично падаючі донизу елементу декілька типів дитячих розвиваючих іграшок, розмір яких по висоті перевищує подвоєну висоту великої літери у назві. 2. На головній сторінці, при наведенні мишки користувачем на зображення одного із авторів курсу, воно стає фліп-картою, де на зворотній стороні наведені його девіз чи цитата, тощо. 3. На сторінці «Наші курси» у верхньому куті ліворуч реалізувати акордеон (розширюється і згортається, допомагає користувачу швидко переміщатись по тексту та включити великі обсяги в обмежений простір), що містить короткий їх опис. 4. На сторінці «Зворотній зв'язок» при наведенні курсора миші користувачем на поле «Детально», воно змінює колір фону, з'являється рамка та тінь від неї, а праворуч нього з'являється текст - підказка (tooltip) «Ваша думка для нас важлива! Конкретизуйте мету звернення, будь ласка». 5. Для користувача, що зайшов на головну сторінку в період з 6 год до 21, колір фону всіх елементів світліший на 40% за колір в інший час.

№ варіанту	Вимоги
5	1. Користувач має можливість натискаючи на кнопку «Напрями діяльності» головної сторінки, бачити випадаюче підменю. 2. Користувач має можливість відвідати сторінку «Напрями діяльності», де під блоком із назвою установи та логотипом, по центру розташована карусель із зображень ілюстративних фото напрямів. Висота елементів каруселі не більше 20% від висоти вікна перегляду для користувача. 3. На сторінці «Напрями діяльності» користувач може здійснити пошук напрямів діяльності за назвою, користуючись спеціальним полем для вводу тексту та спеціальною іконкою біля нього. 4. На сторінці «Зворотній зв'язок» при наведенні курсора миші користувачем на поле «Детально», воно змінює колір фону, з'являється рамка та тінь від неї, а праворуч нього з'являється текст - підказка (tooltip) «Вдячні за Ваш час! Конкретизуйте мету звернення, будь ласка». 5. Для користувача, що зайшов на головну сторінку з пристрою, шириною екрану не більше 768 рх, навігаційне меню розташоване вертикально орієнтованим праворуч. В іншому випадку, навігаційне меню розташоване горизонтально орієнтованим зверху.
6	1. На головній сторінці нижче назви корпорації, на одному з нею фоні користувач має можливість прочитати інформацію про зміни у роботі (наприклад, різдвяні та новорічні канікули, релокацію головного офісу) текстом, що самодрукується червоним кольором з товщиною гліфу у 1,5 рази більше, ніж текст назви корпорації. 2. При наведенні курсора користувачем на блок одного з напрямів на сторінці «Напрями діяльності» з'являється модальне вікно (роруп) із «попередженням» (формулювання попередження на розсуд здобувача, наприклад, про дію закону який обмежує чимось певну діяльність, військовий стан, конфіденційність інформації, тощо) та полем для підтвердження користувачем розуміння його суті. 3. На сторінці «Напрями діяльності», при наведенні мишки користувачем на ілюстративне фото, воно стає фліп-картою, де на зворотній стороні наданий перелік статей із законодавства, що регулює діяльність, тощо. 4. На сторінці «Зворотній зв'язок» при наведенні курсора миші користувачем на поле «Детально», воно змінює колір фону, з'являється рамка та тінь від неї, а праворуч нього з'являється текст - підказка (tooltip) «Дякуючи Вам, ми стаємо кращі! Конкретизуйте мету звернення, будь ласка». 5. Для користувача, що зайшов на головну сторінку в період з 21 год до 6 ранку, колір фону всіх елементів темніший на 40% за колір в інший час.

№ варіанту	Вимоги
7	1. Відвідуючи головну сторінку, користувач на фоні блоку з назвою он-лайн енциклопедії та її логотипом може бачити сніжинки, що періодично рухаються з ліва на право як на фоні так і по верх тексту, розмір яких по висоті перевищує подвоєну висоту великої літери у назві. 2. На сторінці «Розділ про SOME1» у верхньому куті ліворуч реалізувати акордеон (розширюється і згортається, допомагає користувачу швидко переміщатись по інформації та включити великі обсяги в обмежений простір), що містить короткий опис тем. 3. На сторінці ««Розділ про SOME1» користувач може здійснити пошук пряму діяльності за назвою, користуючись спеціальним полем для вводу тексту і написом біля нього «Пошук». 4. На сторінці «Зворотній зв'язок» при наведені курсора миші користувачем на поле «Детально», воно змінює колір фону, з'являється рамка та тінь від неї, а праворуч нього з'являється текст - підказка (tooltip) «Ваша думка для нас важлива! Конкретизуйте мету звернення, будь ласка». 5. Для користувача, що зайшов на головну сторінку з пристрою, шириною екрану не більше 768 рх, навігаційне меню розташоване вертикально орієнтованим ліворуч. В іншому випадку, навігаційне меню розташоване горизонтально орієнтованим зверху.
8	1. На головній сторінці нижче назви он-лайн консультацій, на одному з нею фоні користувач має можливість прочитати текст (вітання з прийдешнім святом текстом, інфо про графік роботи, зміни у роботі), що самодрукується зеленим кольором з товщиною гліфу у 1,5 рази більше, ніж текст назви. 2. Користувач має можливість відвідати сторінку «Послуги», де під блоком із назвою он-лайн консультацій та її логотипом, по центру розташована карусель із зображень послуг. Висота елементів каруселі не більше 20% від висоти вікна перегляду для користувача. 3. Користувач, який протягом 40 сек фокусується на будь-якому лоті сторінки «Послуги», бачить модальне вікно (поверх сторінки, з її блокуванням), яке задає питання на перевірку відповідності діагнозу\потреби користувача лоту, що переглядається (наприклад, послуга терапевт, може спитати, які і на що скарги, як часто біль, тощо), з полем вводу відповіді (питання формує здобувач з свого власного досвіду). 4. На сторінці «Зворотній зв'язок» при наведені курсора миші користувачем на поле «Детально», воно змінює колір фону, з'являється рамка та тінь від неї, а праворуч нього з'являється текст - підказка (tooltip) «Вдячні за Ваш час! Конкретизуйте мету звернення, будь ласка». 5. Для користувача, що зайшов на головну сторінку в період з 6 год до 21, колір фону всіх елементів світліший на 30% за колір в інший час.

№ варіанту	Вимоги
9	1. Відвідуючи головну сторінку, користувач на фоні блоку з назвою сервісного сайту та його логотипом може бачити зображення предметів, які з ним асоціюються (наприклад, ниток та ножиць для ательє), які періодично з'являються та зникають у різних місцях, розмір яких по висоті перевищує подвоєну висоту великої літери у назві. 2. Користувач має можливість натискаючи тексти в меню на сторінці «Каталог сервісів», бачити випадające підменю з переліком доступних підсервісів. 3. На сторінці «Каталог сервісів», при наведенні мишки користувачем на ілюстративне фото, воно стає фліп-картою, де на зворотній стороні надані умови або уточнення (наприклад, прання виробу з даної тканини для ательє). 4. На сторінці «Зворотній зв'язок» при наведенні курсора миші користувачем на поле «Детально», воно змінює колір фону, з'являється рамка та тінь від неї, а праворуч нього з'являється текст - підказка (tooltip) «Дякуючи Вам, ми стаємо кращі! Конкретизуйте мету звернення, будь ласка». 5. Для користувача, що зайшов на головну сторінку з пристрою, шириною екрану не більше 768 рх, навігаційне меню розташоване вертикально орієнтованим праворуч. В іншому випадку, навігаційне меню розташоване горизонтально орієнтованим зверху.
10	1. Користувач, який протягом 30 сек фокусується на розділі з описом мети на головній сторінці, бачить модальне вікно (поверх сторінки, з її блокуванням), яке повідомляє золоті правила навчання (правила формує здобувач з свого власного досвіду). 2. На сторінці «Навчальні теми» користувач може здійснити пошук теми за назвою. 3. На сторінці «Навчальні теми» у верхньому куті ліворуч реалізувати акордеон (розширюється і згортається, допомагає користувачу швидко переміщатись по інформації та включити великі обсяги в обмежений простір), що містить короткий опис навчальних тем. 4. На сторінці «Зворотній зв'язок» при наведенні курсора миші користувачем на поле «Детально», воно змінює колір фону, з'являється рамка та тінь від неї, а праворуч нього з'являється текст - підказка (tooltip) «Ваша думка для нас важлива! Конкретизуйте мету звернення, будь ласка». 5. Для користувача, що зайшов на головну сторінку в період з 21 год до 6 ранку, колір фону всіх елементів темніший на 50% за колір в інший час.

СПИСОК ЛІТЕРАТУРИ

1. Мельник Р. Програмування веб-застосунків (фронт-енд та бек-енд). — Львів: Львівська політехніка, 2018. — 250 с.
2. Двірничук К.В., Вацек Д.О. Веб-програмування та веб-дизайн: навч. посіб. — Чернівці: Чернівець. нац. ун-т ім. Ю. Федьковича. — 2022. — 472 с.
3. Duckett J. HTML & CSS: Design and Build Web Sites. — O'Reilly, 2011. — 512 pp.
4. Robson E., Freeman E. Head First HTML and CSS: A Learner's Guide to Creating Standards-Based Web Pages. — O'Reilly, 2012. — 762 pp.
5. HTML Tutorial [Електронний ресурс]. Режим доступу: <https://www.w3schools.com/html/>
6. CSS Tutorial [Електронний ресурс]. Режим доступу: <https://www.w3schools.com/css/>
7. Responsive web design basics [Електронний ресурс]. Режим доступу: <https://web.dev/articles/responsive-web-design-basics>
8. Top 10 Front-End Frameworks for Responsive Design 2024 [Електронний ресурс]. Режим доступу: <https://daily.dev/blog/top-10-front-end-frameworks-for-responsive-design-2024>
9. Robson E., Freeman E. Head First HTML and CSS: A Learner's Guide to Creating Standards-Based Web Pages. — O'Reilly, 2012 — 762 pp.
10. Довідник з підтримки тегів різними браузерами [Електронний ресурс]. Режим доступу: https://www.w3schools.com/tags/ref_html_browsersupport.asp
11. Flanagan D. JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language. — O'Reilly, 2020. — 704 pp.
12. McMillan M. Data Structures and Algorithms with JavaScript: Bringing classic computing approaches to the Web. — O'Reilly, 2014. — 370 pp.
13. JavaScript Tutorial [Електронний ресурс]. Режим доступу: <https://www.w3schools.com/js/>
14. Resources for developers, by developers. MDN Web Docs [Електронний ресурс] — Режим доступу: <https://developer.mozilla.org/en-US/docs/Web>
15. React Tutorial [Електронний ресурс]. Режим доступу: <https://www.w3schools.com/js/>
16. Vue Tutorial [Електронний ресурс]. Режим доступу: <https://vuejs.org/guide/quick-start.html>
17. Next Tutorial [Електронний ресурс]. Режим доступу: <https://nextjs.org/>
18. Angular Tutorial [Електронний ресурс]. Режим доступу: <https://angular.dev/>
19. Aluri Y. S., Frontend Performance Optimization of Large-Scale E-commerce Landing Pages: A Comprehensive Analysis // International Journal of Computational and Experimental Science and Engineering (IJCESEN). — Vol. 11 (4). — 2025. — pp. 9123-9134.
20. Fauzan P., Hasbullah, Farhan M., Reni P. Technical Performance Comparison of Modern Frontend frameworks Study on Svelte, React, and Vue // Brilliance: Research of Artificial Intelligence — Vol. 5 (1). — May 2025.

Навчально – методичне видання

РОЗРОБКА ФРОНТЕНДУ ПРОГРАМНИХ СИСТЕМ

Лабораторний практикум
для здобувачів спеціальності
F2 (121) «Інженерія програмного забезпечення»