

Лабораторна робота №2: Основи JavaScript

Мета: Навчитись застосовувати основні функціональні можливості мови програмування JavaScript, такі як маніпуляції з елементами веб-сторінки, обробка подій, цикли та умови, для реалізації інтерактивності та динамічності веб-сторінок.

Теоретичні відомості

Вступ до JavaScript та його роль у веб-розробці

JavaScript (JS) - це мова веб-програмування, що забезпечує динамічні та інтерактивні функції на сучасних веб-сайтах. У той час як **HTML** структурує контент, а **CSS** визначає зовнішній вигляд елементів веб-сторінки, JavaScript додає веб-сторінкам **інтерактивності, логіки та функціональності**. Без JavaScript веб-сайти були б статичними, що обмежувало б залучення користувачів та зручність користування.

JavaScript використовується майже у всіх сучасних веб-програмах. Він забезпечує оновлення в реальному часі в стрічках соціальних мереж, створює інтерактивні форми та випадаючі меню, а також дозволяє користувачам динамічно керувати елементами веб-сторінки без необхідності повного перезавантаження сторінки. Така адаптивність значно покращує взаємодію з користувачем, роблячи веб-додатки швидшими та цікавішими.

JavaScript виконується безпосередньо у браузері, що робить його зручним для розробки на стороні клієнта. Однак він також широко використовується для розробки на стороні сервера за допомогою таких середовищ, як **Node.js**. Така універсальність дозволяє розробникам JavaScript створювати додатки, що працюють як з інтерфейсом, так і з серверним функціоналом.

Одним з основних аспектів JavaScript є його здатність керувати **об'єктною моделлю документа (DOM)**. DOM представляє структуру HTML-документа у вигляді дерева елементів, що дозволяє JavaScript динамічно змінювати вміст, стилі та структуру. За допомогою JavaScript розробники можуть змінювати текст, приховувати або показувати елементи, створювати анімацію та реагувати на дії користувача, такі як кліки, прокрутка або введення з клавіатури.

Ще однією важливою особливістю JavaScript є підтримка логічного програмування. Використовуючи **умовні оператори** (if-else), **цикли** (for, while) та **функції**, розробники можуть створювати інтелектуальні, інтерактивні моделі поведінки, які адаптуються до введених користувачем даних та системних подій. Ці основні конструкції програмування дозволяють динамічно оновлювати контент, перевіряти форми та виконувати складні обчислення у веб-додатках.

JavaScript у реальних програмах

JavaScript є основою сучасних веб-додатків. Від **платформ електронної комерції до веб-сайтів соціальних мереж** він відіграє вирішальну роль у покращенні користувацького досвіду.

- **Веб-сайти електронної комерції:** Інтернет-магазини, такі як Amazon та eBay, використовують JavaScript для динамічної фільтрації продуктів, оновлення кошика для покупок та інтерактивного оформлення замовлення. Такі функції, як "рекомендовані товари" та "оновлення цін в реальному часі", працюють на основі логіки, створеної за допомогою JavaScript.
- **Платформи соціальних мереж:** Такі веб-сайти, як Facebook, Twitter та Instagram, використовують JavaScript для надання сповіщень у режимі реального часу, динамічних стрічок та швидкої взаємодії (наприклад, вподобання публікації або додавання коментаря без оновлення сторінки).
- **Веб-додатки:** Інструменти для підвищення продуктивності, такі як Google Docs, Notion і Trello, значною мірою покладаються на JavaScript для спільної роботи в режимі реального часу, автоматичного збереження та перетягування.
- **Розважальні та потокові сервіси:** Такі платформи, як YouTube, Netflix і Spotify, використовують JavaScript для керування відтворенням відео/аудіо, динамічного завантаження контенту та персоналізованих рекомендацій.
- **Інтерактивні форми та дашборди:** Банківські портали, медичні інформаційні панелі та системи підтримки клієнтів використовують JavaScript для перевірки форм, візуалізації даних і створення динамічного контенту.

Без JavaScript цим платформам не вистачало б інтерактивної та динамічної поведінки, якої користувачі очікують у сучасному цифровому світі.

Керування елементами веб-сторінок

Об'єктна модель документа (DOM) - це програмний інтерфейс, який представляє HTML-документ у вигляді структурованого дерева елементів. JavaScript може взаємодіяти з DOM для динамічного оновлення вмісту сторінки, реагування на дії користувача та зміни стилів елементів у режимі реального часу.

Кожен елемент HTML є частиною DOM і може бути змінений або видалений за допомогою JavaScript. Наприклад, JavaScript може:

- Динамічно змінити вміст абзацу.
- Змінити стилі CSS елемента, щоб змінити його вигляд.
- Приховувати або показувати елементи на основі дій користувача.
- Створювати нові HTML-елементи та динамічно додавати їх на сторінку.

Маніпуляції з DOM мають важливе значення для створення інтерактивних веб-додатків. Замість того, щоб перезавантажувати всю веб-сторінку, JavaScript забезпечує цільове оновлення окремих елементів, підвищуючи ефективність і швидкість взаємодії з веб-сторінкою.

Приклади використання керування DOM

1. **Перевірка форм** - запобігання надсиланню користувачами неповних або некоректних даних форм.

2. **Динамічне оновлення контенту** - автоматичне оновлення частин сторінки (наприклад, стрічки новин, ціни на акції).
3. **Інтерактивні навігаційні меню** - розгортання/згортання пунктів меню при наведенні курсору або кліку.
4. **Анімація та переходи** - зникаючі елементи, динамічне переміщення елементів на екрані.
5. **Взаємодія на основі подій** - запуск дій, коли користувачі натискають кнопки, прокручують або натискають клавіші.

Маніпуляції з DOM є основою інтерфейсного JavaScript-програмування, що дозволяє розробникам створювати динамічний інтерфейс без додаткових запитів до сервера.

Обробка подій

Обробка подій - це основна концепція JavaScript, яка дозволяє веб-програмам реагувати на дії користувача. **Події** відбуваються щоразу, коли користувач взаємодіє з веб-сторінкою - наприклад, натискає кнопку, наводить курсор на елемент або вводить дані в поле введення. JavaScript може виявити ці події і виконати певний код у відповідь.

Наприклад, натискання кнопки **"Відправити"** на формі викликає подію, яку JavaScript може перехопити, щоб підтвердити правильне введення даних користувачем перед відправленням. Також, прокрутка сторінки вниз може викликати подію, яка динамічно завантажує більше даних для відображення, як це відбувається на веб-сайтах з нескінченною прокруткою.

JavaScript має способів створення слухачів подій, що стосуються окремих елементів, гарантуючи, що певний код виконується, коли відбувається певна подія. Метод `addEventListener` є найпоширенішим підходом, що дозволяє розробникам відокремити структуру HTML від логіки JavaScript.

Основні події JavaScript:

- **Події натискання (`onclick`)** - спрацьовують, коли користувач натискає на елемент.
- **Події миші (наведення, відведення миші)** - спрацьовують, коли користувач наводить курсор миші на елемент або віддаляється від нього.
- **Події клавіатури (натискання клавіші вниз, натискання клавіші вгору)** - спрацьовують, коли користувач вводить дані у полі введення.
- **Події форми (`onsubmit`, `onchange`)** - використовуються для валідації та обробки відправлених форм.
- **Події прокрутки (`onscroll`)** - виявлення, коли користувач прокручує сторінку, щоб запустити динамічне завантаження контенту.


Обробка подій має вирішальне значення для того, щоб веб-додатки були інтерактивними та швидко реагували. Без неї користувачам довелося б вручну оновлювати сторінки або переходити через кілька екранів, щоб побачити оновлення.

Цикли та умовні оператори JavaScript

Веб-додатки часто потребують логіки для визначення способу відображення або обробки контенту. JavaScript реалізує **умовні оператори** (if-else) для виконання різних дій на основі певних умов.

Наприклад, інтернет-магазин може використовувати оператор if для перевірки наявності товару на складі, перш ніж дозволити користувачеві додати його до кошика. Якщо товару немає в наявності, на екрані з'явиться відповідне повідомлення.

Приклад оператора if-else:



```
let stock = 10;
if (stock > 0) {
    console.log("Product is available!");
} else {
    console.log("Out of stock.");
}
```

Цикли (for, while) дозволяють JavaScript ефективно виконувати повторювані завдання. Замість того, щоб вручну писати код для кожного елемента в списку, цикл може ітераційно переглядати колекцію елементів, динамічно оновлюючи або змінюючи їх.

Приклад циклу для перебору елементів списку:



```
let items = ["Apple", "Banana", "Cherry"];
for (let i = 0; i < items.length; i++) {
    console.log("Item: " + items[i]);
}
```

Наведені програмні конструкції дозволяють розробникам створювати більш динамічні та інтелектуальні додатки, зменшуючи об'єм написаного коду і підвищуючи продуктивність його роботи.

Чому JavaScript необхідний для сучасної веб-розробки

JavaScript - це необхідний інструмент для створення сучасних інтерактивних веб-додатків. Він дозволяє:

- **Динамічне оновлення контенту** - підвищує адаптивність веб-сайтів без перезавантаження сторінки.
- **Взаємодія з користувачами** - покращення взаємодії за допомогою форм, кнопок та анімації.
- **Покращений користувацький досвід** - забезпечення плавної, інтуїтивно зрозумілої навігації.
- **Функції реального часу** - увімкнення чату, сповіщень та оновлень у реальному часі.
- **Повностекова розробка** - розробка клієнтської частини (за допомогою таких фреймворків, як React, Vue, Angular) та серверної частини (за допомогою Node.js).

Без JavaScript веб-сайти були б статичними, вимагали б постійних ручних дій користувачів для оновлення контенту. Можливість JavaScript маніпулювати DOM, обробляти події та виконувати логіку робить його критично важливим компонентом сучасної веб-розробки.

Порядок виконання роботи

Крок 1: Налаштування JavaScript

Перш ніж реалізувати керування елементами з DOM, циклами та умовами, вам потрібно включити файл JavaScript в HTML-документ. Це гарантує, що браузер завантажить і виконає код JavaScript під час відображення сторінки.

1. Зв'язування **файлу JavaScript з HTML**. Щоб включити JavaScript код, потрібно додати тег `<script>` безпосередньо перед закриваючим тегом `</body>` у HTML-файлі, зв'язавши його з вашим зовнішнім файлом JavaScript.



```
<body>
  <script src="script.js"></script>
</body>
```

Це гарантує, що JavaScript завантажується після вмісту HTML, тому всі елементи доступні для керування елементами.

Крок 2: Керування елементами DOM

Керування елементами DOM дозволяють взаємодіяти з HTML-документом і змінювати його за допомогою JavaScript.

1. **Виділення елементів HTML.** Щоб взаємодіяти з елементами HTML, спочатку потрібно виділити їх за допомогою JavaScript. Використовуйте методи `document.getElementById()`, `document.querySelector()` та `document.querySelectorAll()` для вибору окремих елементів або груп елементів.



```
// Select an element by ID
const header = document.getElementById('header');

// Select an element by class
const navLinks = document.querySelectorAll('.nav-link');
```

2. **Зміна вмісту.** Вибравши елемент, можна змінити його вміст за допомогою властивості `.innerHTML` або `.textContent` для звичайного тексту.



```
// Changing the header text
header.innerHTML = 'Welcome to My Updated Website!';

// Changing text of all navigation links
navLinks.forEach(link => {
  link.textContent = 'New Link Text';
});
```

3. **Модифікація атрибутів** елементів. Також є можливість змінювати атрибути елементів за допомогою `.setAttribute()` для таких атрибутів, як `href`, `src` тощо.



```
// Changing the 'href' attribute of a link
const link = document.querySelector('a');
link.setAttribute('href', 'https://newurl.com');
```

Крок 3: Обробка подій

JavaScript дозволяє взаємодіяти з користувачами за допомогою подій, таких як кліки, натискання клавіш, заповнення форм тощо. Ви можете додавати слухачі подій до елементів і визначати функції, які будуть виконуватися, коли ці події відбуваються.

1. **Додавання слухачів подій.** Потрібно використати `addEventListener()` для додавання обробників подій до елементів. Можна вказати тип події (наприклад, клік, відправлення, наведення миші) і функцію зворотного виклику, яка буде виконуватися при її спрацьовуванні.



```
// Example: Adding a click event to a button
const button = document.getElementById('myButton');
button.addEventListener('click', function() {
    alert('Button clicked!');
});
```

2. **Обробка форм** Якщо веб-сторінка містить форми, є можливість прослуховувати подію відправлення і запобігти відправленню форми за замовчуванням, щоб обробити її за допомогою JavaScript.



```
const form = document.getElementById('myForm');
form.addEventListener('submit', function(event) {
    event.preventDefault(); // Prevent form from submitting
    alert('Form submitted!');
});
```

Крок 4: Цикли в JavaScript

Цикли дозволяють повторювати дії певну кількість разів або до виконання певної умови. У JavaScript найпоширенішими циклами є `for`, `while` та `forEach`.

1. **Використання циклу `for`** Цикл `for` ідеально підходить, коли відомо, скільки разів потрібно повторити дію.



```
// Loop through an array of numbers and log each number
const numbers = [1, 2, 3, 4, 5];
for (let i = 0; i < numbers.length; i++) {
  console.log(numbers[i]);
}
```

2. **Використання циклу while.** Цикл while використовується, коли потрібно повторити дію, доки умова є істинною.



```
// Example: Log numbers until we reach 5
let i = 0;
while (i < 5) {
  console.log(i);
  i++;
}
```

3. **Використання forEach.** Цикл forEach використовується для перебору масивів. Він більш лаконічний, ніж звичайний цикл for, і особливо корисний при роботі з масивами.



```
// Loop through an array and log each element
const fruits = ['apple', 'banana', 'cherry'];
fruits.forEach(function(fruit) {
  console.log(fruit);
});
```


Крок 5: Умовні оператори

Умовні оператори дозволяють керувати потоком вашого коду на основі певних умов. У JavaScript основними умовними операторами є if, else if та else.

1. **Базовий оператор if.** Оператор if перевіряє умову і виконує код, якщо умова істинна.



```
const age = 18;  
if (age >= 18) {  
    console.log('You are an adult.');
```

2. **else if та else.** Існує можливість розширити оператор if за допомогою else if для перевірки декількох умов, а else для виконання коду, коли жодна з умов не є істинною.



```
const temperature = 30;  
if (temperature > 30) {  
    console.log('It\'s very hot outside!');  
} else if (temperature > 20) {  
    console.log('The weather is warm.');
```

3. **Тернарний оператор.** Більш стислим способом запису умов є тернарний оператор, який можна використовувати замість простих операторів if-else.



```
const age = 18;  
const status = (age >= 18) ? 'Adult' : 'Minor';  
console.log(status);
```

Крок 6: Поєднання концепцій

Тепер, після розглянутих способів керування DOM, обробки подій та використання циклів й умов, настав час об'єднати ці навички в інтерактивній веб-сторінці.

1. **Приклад: Зміна кольору фону при натисканні кнопки** У цьому завданні ви додасте слухача подій до кнопки, яка змінює колір фону сторінки при натисканні. Можна використовувати цикл і умовний оператор для циклічного перегляду списку кольорів.



```
<button id="colorButton">Change Color</button>  
  
<script>  
  const colors = ['red', 'blue', 'green', 'yellow'];  
  let currentColor = 0;  
  
  document.getElementById('colorButton').addEventListener('click', function() {  
    document.body.style.backgroundColor = colors[currentColor];  
    currentColor = (currentColor + 1) % colors.length;  
  });  
</script>
```

Цей приклад демонструє:

- Маніпуляції з DOM (зміна кольору фону).
- Обробка подій (натискання кнопки).
- Цикли (циклічне чергування кольорів).

Лабораторне завдання

Завдання 1: Керування DOM за допомогою циклів та умовної логіки

Опис:

Вибір елементів та їх модифікації за допомогою циклів та умовних операторів для створення динамічної поведінки.

Кроки:

1. Використати JavaScript для виділення декількох елементів (наприклад, всіх елементів у списку або всіх зображень).
2. Використати цикл **for** для циклічного перебору колекції елементів (наприклад, для зміни кольору фону кожного елемента списку при завантаженні сторінки).
3. Застосувати оператори **if-else** для зміни вмісту або стилю елементів на основі певних умов (наприклад, зміна текстового вмісту елемента на основі його індексу або атрибутів даних).
4. Використати **querySelectorAll**, щоб вибрати кілька елементів і перебирати їх, динамічно змінюючи їхній стиль або вміст.

Завдання 2: Обробка подій та динамічні оновлення

Опис:

Додавання інтерактивності з подіями та оновлення DOM на основі дій користувача.

Кроки:

1. Створити кнопку, яка при натисканні перемикає видимість розділу сторінки (наприклад, показує/приховує додатковий опис, розділ коментарів або меню).
2. Реалізувати **логіку if-else** для перевірки стану видимості елемента (наприклад, якщо секція прихована, показати її; якщо вона видима, сховати її).
3. Використати **цикл for для** додавання обробників подій до декількох елементів (наприклад, кнопок або посилань у навігаційному меню). Кожна кнопка повинна змінювати вміст певної області на сторінці при натисканні.
4. Створити **ефект наведення**, коли при наведенні на елемент змінюється його вміст або стиль, використовуючи обробник подій і умовну логіку (наприклад, наведення на пункт меню, щоб показати його опис).

Завдання 3: Динамічне керування контентом

• Опис:

Додавання динамічної функціональності до веб-сторінки, наприклад, створення або оновлення вмісту на основі введення користувачем даних.

• Кроки:

1. Створити форму з полями для введення (наприклад, ім'я, електронна пошта або коментар) і кнопкою надсилання.
2. Використати **JavaScript для збору даних з форми** та **оператор if-else** для перевірки даних (наприклад, щоб переконатися, що користувач не заповнив порожні поля).
3. Після валідації використати **маніпуляції з DOM** для динамічного додавання введених користувачем даних до розділу сторінки (наприклад, відобразити коментар під статтею або дані користувача в області підтвердження).

4. Використати **цикл for для** динамічного відображення списку елементів (наприклад, дописів блогу або пунктів меню), з кожним з яких можна взаємодіяти (наприклад, при кліку на заголовок допису, відобразиться його повний зміст).

Контрольні запитання

1. Яку роль відіграє JavaScript у веб-розробці та як він взаємодіє з HTML і CSS?
2. Що таке об'єктна модель документа (DOM) і як JavaScript може змінювати її? Наведіть приклад.
3. Як працює обробка подій у JavaScript? Наведіть приклад використання `addEventListener()`.
4. Які є основні типи циклів у JavaScript і коли варто використовувати `for`, `while` та `forEach`?
5. Як умовні оператори (`if-else`) допомагають створювати динамічну поведінку веб-сторінки? Наведіть приклад.

Вимоги до оформлення звіту

1. Увійти в папку проекту



```
cd username.github.io
```

2. Створити нову папку з назвою **lab2** і помістити всі файли створені під час виконання лабораторної роботи в дану папку.
3. Завантажити зміни у репозиторій



```
git add --all  
git commit -m "Commit lab"  
git push -u origin main
```

Варіанти завдань

Варіант 1

Веб-сайт ресторану

1. Використати цикл for для відображення пунктів меню у таблиці
2. До кожного пункту меню додати кнопку «Інгредієнти», а також прихований блок з списком інгредієнтів страви та змінювати видимість даного блоку при натисканні кнопки «Інгредієнти».
3. На сторінці Про нас реалізувати форму коментарів, де користувачі можуть залишати відгуки про ресторан та динамічно відображати їх після додавання окремим списком.

Варіант 2

Веб-сайт організації онлайн-ігор

1. Використати цикл while для генерації випадкового списку рекомендованих ігор, що змінюються при оновленні сторінки.
2. Додати кнопку "Додати до улюблених", яка при натисканні змінює розмір картки гри та додає рамку іншого кольору. Використати подію подвійного кліку (dblclick) для взаємодії.
3. На сторінці профілю реалізувати відображення анімованого прогресу користувача, що показує, скільки ще потрібно ігор для отримання наступної нагороди.

Варіант 3

Веб-сайт платформи для навчальних курсів

1. Використати do...while для автоматичного оновлення розкладу занять у певний період часу.
2. Додати кнопку "Розпочати курс", яка при натисканні змінює положення блоку з описом курсу на сторінці (переміщує вище). Додати кнопку "Курс пройдено", яка при натисканні змінює колір фону блоку курсу та відображає текст «Пройдено».
3. Реалізувати візуальну шкалу прогресу навчання, яка заповнюється залежно від кількості пройдених курсів.

Варіант 4

Веб-сайт платформи для бронювання турів

1. Використати while для відображення турів у списку.
2. Додати кнопку "Забронювати тур", яка при натисканні збільшує розмір картки туру (робить її більш виразною) та додає в "Мої бронювання".
3. Реалізувати інтерактивну мапу популярних туристичних місць, що дозволяє кліком на маркер переглядати коротку інформацію про кожне місце.

Варіант 5

Веб-сайт онлайн-магазину книг

1. Використати цикл for для генерації списку рекомендованих книг, який змінюється при оновленні сторінки.
2. Додати кнопку "Додати до кошика", яка при натисканні додає книгу в кошик. Кнопка змінює колір після додавання товару.
3. Реалізувати функціональність для підрахунку вартості товарів у кошику в реальному часі, коли кількість товару змінюється.

Варіант 6

Веб-сайт платформи для оренди автомобілів

1. Використати while для генерації списку автомобілів, які доступні для оренди на даний момент.
2. Додати кнопку "Забронювати", яка при натисканні додає автомобіль у список бронювань користувача. Кнопка змінює колір і додає ефект анімації при натисканні.
3. Реалізувати календар для вибору дати початку та кінця оренди автомобіля.

Варіант 7

Веб-сайт платформи для оренди житла

1. Використати do...while для виведення списку доступних квартир.
2. Додати кнопку "Забронювати", яка при натисканні додає квартиру в список бронювань користувача, а також змінює статус доступності цієї квартири (позначає її як заброньовану).
3. Реалізувати інтерактивну мапу для відображення місця розташування квартир на карті. При натисканні на маркер відображати додаткову інформацію про квартиру.

Варіант 8

Веб-сайт платформи для онлайн-бронювання квитків на заходи

1. Використати while для відображення лише майбутніх подій.
2. Додати кнопку "Забронювати квиток", яка при натисканні додає подію в розділ "Мої бронювання" і змінює статус кнопки на "Заброньовано".
3. Реалізувати вибір кількості квитків при бронюванні та підрахунок загальної вартості квитків.

Варіант 9

Веб-сайт платформи для організації волонтерських ініціатив

1. Використати `while` для відображення лише актуальних ініціатив, які ще не завершилися.
2. Додати кнопку "Приєднатися", яка при натисканні додає ініціативу в розділ "Мої ініціативи" і змінює статус кнопки на "Ви приєдналися".
3. Реалізувати оновлення значення кількості волонтерів, яких ще потрібно залучити, після приєднання користувача.

Варіант 10

Веб-сайт онлайн-магазину спортивних товарів

1. Використати `while` для відображення тільки доступних до покупки товарів.
2. Додати кнопку "Додати до кошика", яка при натисканні додає товар до кошика та змінює статус кнопки на "Товар у кошику".
3. Реалізувати можливість зміни кількості товарів у кошику, а також автоматично оновлювати загальну суму покупки.

Варіант 11

Веб-сайт платформи для пошуку робочих вакансій

1. Використати цикл `for` для генерації списку вакансій на основі даних із бази.
2. Додати кнопку "Подати заявку", яка при натисканні змінює статус вакансії на "Подано", відображаючи повідомлення про успішну подачу заявки.
3. Реалізувати систему фільтрації вакансій за категорією, містом і рівнем зарплати, застосовуючи випадючі списки.

Варіант 12

Веб-сайт платформи для замовлення їжі

1. Використати цикл `for` для генерації списку страв у меню та кошику, цикл `while` у списку "Мої замовлення".
2. Додати кнопку "Додати в кошик", яка змінює колір після натискання та додає страву в кошик.
3. Реалізувати таймер очікування доставки замовлення після оформлення.

Варіант 13

Веб-сайт блогу для подорожей

1. Використати цикл `while` для виводу статей у списку.
2. Додати кнопку "Подобається", яка змінює стиль статті після натискання.
3. Реалізувати можливість коментування статей та динамічне оновлення списку коментарів.

Варіант 14

Веб-сайт інтерактивної платформи для вивчення мов

1. Використати масив об'єктів та цикл `for` для генерації списку уроків.
2. Додати кнопку "Відзначити як пройдений", яка змінює стиль уроку після натискання та зберігає стан у локальному сховищі.
3. Реалізувати таймер щоденного навчання, який відстежує витрачений час.

Варіант 15

Веб-сайт платформи для вивчення історії через інтерактивні події

1. Використати масив історичних подій та цикл `while` для генерації контенту на сторінці хронології.
2. Додати кнопку "Дізнатися більше" на часовій шкалі, яка розкриває додаткові деталі події у модальному вікні.
3. Реалізувати тест для перевірки знань, що відкривається натисканням на кнопку «Тест» у розділі «Події».

Варіант 16

Веб-сайт симулятора управління власним стартапом

1. Використати цикл `map` для відображення списків конкурентів та інвесторів.
2. Додати кнопку форму "Моделювання бізнес процесу", яка змінює параметри стартапу (наприклад, збільшує прибуток чи кількість працівників).
3. Реалізувати динамічний звіт у вигляді графіку та таблиць порівняння реальних та змодельованих параметрів бізнесу (доходи, витрати, кількість працівників, тощо).

Варіант 17

Веб-сайт платформи для створення та управління персональними цілями

1. Використати методи масивів (map, filter) для управління списком цілей.
2. Додати кнопку "Виконано", яка змінює статус цілі та оновлює прогрес.
3. Реалізувати щоденний мотивуючий таймер, що нагадує про заплановані завдання.

Варіант 18

Веб-сайт платформи для віртуальних хакатонів та змагань із програмування

1. Використати цикл forEach для генерації списку учасників у рейтингу.
2. Додати кнопку "Подати заявку", яка змінює стиль після натискання та додає користувача до списку учасників.
3. Реалізувати таймер до завершення хакатону, що оновлюється в реальному часі.

Варіант 19

Веб-сайт симулятора управління будівництвом містом

1. Використати цикл do..while для виведення інформації про будівлі та ресурси.
2. Додати кнопку "Покращити будівлю", яка змінює її вигляд (фото) та покращує характеристики, але забирає ресурси у міста.
3. Реалізувати динамічне зменшення ресурсів міста після створення нових чи покращення старих будівель.

Варіант 20

Веб-сайт симулятора космічної експедиції

1. Використати цикл for для генерації списку доступних експедицій.
2. Додати кнопку "Почати місію", яка змінює статус експедиції та додає запис у журнал подорожей.
3. Реалізувати таймер тривалості польоту, що оновлюється в реальному часі.

Варіант 21

Веб-сайт платформи для моніторингу здоров'я та фізичної активності

1. Використати цикл for виведення інформації про тренування.
2. Додати кнопку "Почати тренування", яка змінює статус тренування та оновлює прогрес.
3. Реалізувати журнал тренувань, що показує список проведених тренувань, тип проведеного тренування та тренувальний час.

Варіант 22

Веб-сайт платформи для онлайн-курсу з фотографії

1. Використати цикл while для виведення інформації про уроки та фотографії.
2. Додати кнопку "Завантажити фотографію", яка додає нову фотографію до галереї.
3. Реалізувати фільтрацію фотографій за типами знімків (портрети, пейзажі, макрозйомка тощо).

Варіант 23

Веб-сайт платформи для оренди спортивного обладнання

1. Використати цикл do..while для виведення даних про обладнання та оренди.
2. Додати кнопку "Орендувати", яка змінює колір після натискання та додає товар до кошика.
3. Реалізувати функціонал для додавання дати початку та закінчення оренди.

Варіант 24

Веб-сайт платформи для планування подорожей

1. Використати цикл do..while для виведення інформації про місця та подорожі.
2. Додати кнопку "Додати до моїх подорожей", яка додає місце до запланованої подорожі.
3. Реалізувати функціонал для розрахунку загального бюджету подорожі в залежності від вибраних місць і тривалості.

Варіант 25

Веб-сайт платформи для створення та обміну рецептами

1. Використати цикли `for` для виведення рецептів та відгуків.
2. Додати кнопку "Додати рецепт", яка відкриває форму для створення нового рецепту.
3. Реалізувати функціонал для редагування рецептів після їх створення.