Волинський національний університет імені Лесі Українки

Факультет інформаційних технологій і математики
Кафедра комп’ютерних наук та кібербезпеки

Валентина ЮНЧИК

ТЕХНОЛОГІЇ ВЕБРОЗРОБКИ

Методичні рекомендації до виконання лабораторних робіт
для здобувачів освіти спеціальності
125 Кібербезпека та захист інформації
першого (бакалаврського) рівня

Луцьк 2024

УДК 658.512.23:004.4(075)
Ю-72

Рекомендовано до друку науково-методичною радою Волинського національного університету імені Лесі Українки
(Протокол № 3 від 22 листопада 2024 р.)

Рецензент: ***Кальчук І.В.*** – кандидат фіз.-мат. наук, доцент кафедри теорії функцій та методики навчання математики Волинського національного університету імені Лесі Українки;

Юнчик В.Л.

Ю-72 Технології веброзробки: методичні рекомендації до виконання лабораторних робіт для здобувачів освіти спеціальності 125 Кібербезпека та захист інформації першого (бакалаврського) рівня [Електронний ресурс] / Валентина Леонідівна Юнчик; ВНУ імені Лесі Українки. Електронні текстові данні (1 файл: 1,89 МБ). Луцьк : ВНУ імені Лесі Українки, 2024. 52 с.

Методичні рекомендації до виконання лабораторних робіт для освітнього компонента Технології веброзробки містять теоретичні відомості до тем першого та другого модулів, а також завдання до лабораторних танять та запитання для самоконтролю. Методичні рекомендації до виконання лабораторних робіт укладені згідно з силабусом ОК «Технології веброзробки» для студентів ІІІ-го курсу спеціальності 125 Кібербезпека.

УДК 658.512.23:004.4(075)

ЗМІСТ

Лабораторна робота №1. 4
Лабораторна робота №2. 10
Лабораторна робота №3. 14
Лабораторна робота №4. 18
Лабораторна робота №5. 22
Лабораторна робота №6. 25
Лабораторна робота №7. 31
Лабораторна робота №8-9. 34
Лабораторна робота №10-11 39
Лабораторна робота №12-13 44
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 51

Лабораторна робота №1.

Мова гіпертекстової розмітки HTML. Структура HTML- документа. Теги для роботи з текстом. Теги для створення малюнків.

Мета лабораторної роботи. *Навчитися створювати базову структуру HTML-документа та використовувати теги для роботи з текстом і зображеннями. Закріпити навички форматування тексту за допомогою таких тегів, як заголовки, абзаци, списки, цитати, виділення тексту та інші. Опанувати вставку та налаштування зображень на вебсторінці. Підготувати вебсторінку з чітко структурованим контентом, що використовує різні елементи форматування тексту та графічні компоненти.*

Теоретичні відомості

HTML (HyperText Markup Language) — це мова розмітки, що використовується для створення вебсторінок. HTML складається з елементів, які визначають структуру та вміст вебсторінки. Елемент HTML зазвичай складається з відкривального та закривального тегу, між якими розміщується контент. Наприклад:

```
<p>Це абзац тексту</p>
```

Кожен HTML-документ має базову структуру, що включає такі основні теги:

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Назва сторінки</title>
</head>
<body>
  <!-- Контент вебсторінки -->
</body>
</html>
```

<!DOCTYPE html> — визначає тип документа як HTML5.
<html> — кореневий елемент HTML-документа.
<head> — містить метадані документа (назва, кодування, інформація для пошукових систем).
<title> — заголовок, який відображається у вкладці браузера.
<body> — містить видимий вміст вебсторінки.

Теги для роботи з текстом

HTML містить велику кількість тегів для форматування та структуризації тексту.

Заголовки. <h1> - <h6> — визначають заголовки різного рівня. <h1> — найбільший заголовок, <h6> — найменший.

```
<h1>Основний заголовок</h1>
<h2>Підзаголовок</h2>
```

Абзаци. <p> — для створення абзаців тексту.

```
<p>Це абзац тексту.</p>
```

Переноси.
 — використовується для переносу рядка.

```
Текст перед переносом<br>Текст після переносу
```

Жирний текст. — виділяє текст як важливий.

```
<strong>Важлива інформація</strong>
```

Курсив. — виділяє текст курсивом.

```
<em>Цей текст курсивом</em>
```

Виділений текст. <mark> — підсвічує текст.

```
<mark>Виділений текст</mark>
```

Цитати. <q> — для коротких цитат; <blockquote> — для довгих цитат.

```
<q>Це коротка цитата</q>
<blockquote>Це довга цитата, яка займає декілька рядків.</blockquote>
```

Списки.

Нумерований список (<ol>)

```
<ol>
  <li>Перший пункт</li>
  <li>Другий пункт</li>
</ol>
```

Маркований список (<ul>)

```
<ul>
  <li>Пункт один</li>
  <li>Пункт два</li>
</ul>
```

Теги для створення зображень

Для вставки зображень у HTML використовується тег <img>, який не має закривального тегу.

<img src="image.jpg" alt="Опис зображення" width="300" height="200">
src — шлях до файлу зображення.
alt — альтернативний текст, який відображається, якщо зображення не завантажується.
width та height — задають розміри зображення в пікселях.

Семантичні елементи

HTML5 вводить нові семантичні теги, які дозволяють краще організувати структуру вебсторінки:

<header> — заголовок сторінки або секції.
<nav> — навігаційне меню.
<section> — розділ або секція контенту.
<article> — окремий блок контенту, наприклад, стаття.
<aside> — додатковий контент, наприклад, бокова панель.
<footer> — нижній колонтитул сторінки.

Впровадження мультимедіа. Крім зображень, HTML дозволяє вставляти відео та аудіо.

Відео

```
<video controls>
  <source src="video.mp4" type="video/mp4">
```

```
    Ваш браузер не підтримує відтворення відео.
  </video>
```

Аудіо

```
  <audio controls>
    <source src="audio.mp3" type="audio/mpeg">
    Ваш браузер не підтримує відтворення аудіо.
  </audio>
```

Приклад

Ось приклад простої HTML-сторінки з текстовими елементами та зображенням:

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Моя перша сторінка</title>
</head>
<body>
  <h1>Вітання!</h1>
  <p>Це моя перша HTML-сторінка.</p>
  <h2>Зображення</h2>
  <img src="cat.jpg" alt="Кіт" width="300">
  <h2>Цитата</h2>
  <blockquote>Це чудова цитата, яка надихає!</blockquote>
  <h2>Список справ</h2>
  <ul>
    <li>Вивчити HTML</li>
    <li>Навчитися форматувати текст</li>
    <li>Додавати зображення</li>
  </ul>
</body>
</html>
```

Завдання до лабораторної роботи

Виконати індивідуальне завдання відповідно до Вашого порядкового номера групи.

Завдання 1: Створення сторінки події.

Опис завдання. Створити вебсторінку, яка описує подію (наприклад, фестиваль або концерт). Для оформлення тексту потрібно використати відповідні теги, зокрема: <dd>, <small>, <mark>, <dfn>, <abbr>, <q>, , <h1>-<h6>, <section>, <aside>.

На сторінці має бути:
- Назва події у заголовку.
- Опис події, використовуючи різні текстові стилі та форматування.
- Список учасників або виконавців.

• Зображення з події.
• Контактна інформація.

Завдання 2: Створення сторінки біографії

Опис завдання. Створити вебсторінку, яка описує біографію відомої особи (наприклад, письменника, художника або науковця). Використовувати відповідні теги для текстового форматування, зокрема: <dd>, <small>, <mark>, <dfn>, <abbr>, <q>, , <h1>-<h6>, <section>, <aside>.

На сторінці має бути:

- Ім'я та заголовок, що містить роки життя.
- Короткий опис основних досягнень.
- Цитата відомої фрази цієї людини, оформлена за допомогою відповідних тегів.
- Зображення особи.
- Список книг/робіт або досягнень.

Завдання 3: Створення сторінки рецепта

Опис завдання. Створити вебсторінку з рецептом страви. Використовувати різні текстові теги для структурування інформації, зокрема: <dd>, <small>, <mark>, <dfn>, <abbr>, <q>, , <h1>-<h6>, <section>, <aside>.

На сторінці має бути:

- Назва страви у заголовку.
- Інгредієнти у вигляді списку.
- Покрокова інструкція з приготування.
- Малюнок страви.
- Коментар щодо особливостей приготування (тег <small> для приміток).

Завдання 4: Створення сторінки огляду книги або фільму

Опис завдання. Створити вебсторінку з оглядом книги або фільму. Використовувати різні текстові теги для структурування інформації, зокрема: <dd>, <small>, <mark>, <dfn>, <abbr>, <q>, , <h1>-<h6>, <section>, <aside>.

На сторінці має бути:

- Назва книги або фільму у заголовку.
- Короткий огляд сюжету з використанням різних стилів тексту.
- Цитата або відгук критика, оформлена за допомогою тегу <q>.
- Зображення обкладинки книги або постеру фільму.
- Розділ з контактною інформацією автора рецензії або сайту.

Завдання 5: Створення сторінки новини

Опис завдання. Створити вебсторінку новини на певну тему (спорт, технології, музика тощо). Використовувати різні текстові теги для структурування інформації, зокрема: <dd>, <small>, <mark>, <dfn>, <abbr>, <q>, , <h1>-<h6>, <section>, <aside>.

На сторінці має бути:

- Заголовок новини.
- Текст новини з використанням тегів для різних стилів тексту.
- Виділення важливих фактів за допомогою тегу <mark>.
- Зображення, що ілюструє новину.
- Контактна інформація для зворотного зв'язку.

Завдання 6: Створення сторінки розкладу заходів

Опис завдання. Створити вебсторінку, яка показує розклад заходів на певну тему (наприклад, розклад конференції). Використовувати різні текстові теги для структурування інформації, зокрема: <dd>, <small>, <mark>, <dfn>, <abbr>, <q>, , <h1>-<h6>, <section>, <aside>.

На сторінці має бути:

- Заголовок з темою заходу.
- Список заходів або сесій з часом та місцем проведення.
- Виділення важливої інформації за допомогою тегу .
- Зображення або логотип заходу.
- Контактна інформація для реєстрації.

Завдання 7: Створення сторінки навчального курсу

Опис завдання. Створити вебсторінку з описом навчального курсу. Використовувати різні текстові теги для структурування інформації, зокрема: <dd>, <small>, <mark>, <dfn>, <abbr>, <q>, , <h1>-<h6>, <section>, <aside>.

На сторінці має бути:

- Назва курсу у заголовку.
- Короткий опис курсу з використанням різних текстових стилів.
- Список тем, що вивчаються у курсі.
- Малюнок, що ілюструє курс.
- Контактна інформація для реєстрації або додаткових запитань.

Завдання 8: Створення сторінки відгуків користувачів

Опис завдання. Створити вебсторінку з відгуками користувачів про продукт або послугу. Використовувати різні текстові теги для структурування інформації, зокрема: <dd>, <small>, <mark>, <dfn>, <abbr>, <q>, , <h1>-<h6>, <section>, <aside>.

На сторінці має бути:

- Заголовок з назвою продукту або послуги.
- Кілька відгуків користувачів, кожен з яких оформлений у власному блоці з використанням тегів для цитат та форматування тексту.
- Зображення продукту або логотип компанії.
- Контактна інформація для подання відгуків.

Завдання 9: Створення сторінки опису місця для відпочинку

Опис завдання. Створити вебсторінку, що описує певне місце для відпочинку (наприклад, парк, курорт, історичне місце). Використовувати різні текстові теги для структурування інформації, зокрема: <dd>, <small>, <mark>, <dfn>, <abbr>, <q>, , <h1>-<h6>, <section>, <aside>.

На сторінці має бути:

- Назва місця у заголовку.
- Опис основних атракцій або можливостей для відпочинку.
- Список послуг або зручностей.
- Зображення місця.
- Контактна інформація для отримання додаткових даних або бронювання.

Завдання 10: Створення сторінки спортивного заходу

Опис завдання. Створити вебсторінку спортивного заходу (матч, турнір тощо). Використовувати різні текстові теги для структурування інформації, зокрема: <dd>, <small>, <mark>, <dfn>, <abbr>, <q>, , <h1>-<h6>, <section>, <aside>.

На сторінці має бути:

- Назва заходу у заголовку.
- Короткий опис події та її учасників.
- Графік проведення або список матчів.
- Зображення з попередніх подій або логотип команди.
- Контактна інформація для придбання квитків.

Завдання 11: Створення сторінки галереї зображень

Опис завдання. Створити вебсторінку, яка є галереєю зображень на певну тему (наприклад, природа, міста). Використовувати різні текстові теги для структурування інформації, зокрема: <dd>, <small>, <mark>, <dfn>, <abbr>, <q>, , <h1>-<h6>, <section>, <aside>.

На сторінці має бути:

- Назва галереї у заголовку.
- Короткий опис тематики галереї.
- Кілька зображень, що відображають обрану тему.
- Підписи до кожного зображення з використанням тегів для форматування.
- Контактна інформація для замовлення фотографій.

Завдання 12: Створення сторінки з порадами щодо здорового способу життя

Опис завдання. Створити вебсторінку, яка містить поради щодо здорового способу життя (наприклад, фізичні вправи, харчування, сон). Використовувати різні текстові теги для структурування інформації, зокрема: <dd>, <small>, <mark>, <dfn>, <abbr>, <q>, , <h1>-<h6>, <section>, <aside>.

На сторінці має бути:

- Назва сторінки у заголовку.
- Список основних порад, оформлених за допомогою тегів для списків.
- Виділення важливої інформації за допомогою тегу <mark>.
- Малюнок, що ілюструє здоровий спосіб життя (наприклад, людина, що займається спортом).
- Контактна інформація для отримання додаткових консультацій.

Завдання 13: Створення сторінки опису книги рецептів

Опис завдання. Створити вебсторінку, що описує книгу рецептів, яка містить кілька рецептів із зображеннями. Використовувати різні текстові теги для структурування інформації, зокрема: <dd>, <small>, <mark>, <dfn>, <abbr>, <q>, , <h1>-<h6>, <section>, <aside>.

На сторінці має бути:

- Назва книги у заголовку.
- Опис книги, включаючи кілька рецептів з різних розділів.
- Список інгредієнтів для кожного рецепту.
- Зображення страв, що входять до книги.
- Контактна інформація для замовлення книги.

Запитання для самоперевірки

1. Що таке HTML і для чого він використовується?
2. Які основні теги використовуються для створення таблиці? Опишіть їх функції.
3. Як об'єднати комірки таблиці по горизонталі та вертикалі?
4. Який атрибут дозволяє налаштувати гіперпосилання, щоб воно відкривалось у новій вкладці?
5. Як створити кнопку для відправлення форми?
6. Які атрибути використовуються для налаштування текстових полів у формах?
7. Як додати стилі до таблиці? Назвіть приклади CSS-властивостей для таблиці.
8. Чим радіокнопки відрізняються від чекбоксів? Наведіть приклади.
9. Як створити випадаючий список у HTML? Який атрибут дозволяє реалізувати багатокількісний вибір?

Лабораторна робота №2.

Мова гіпертекстової розмітки HTML. Інтерактивні теги. Теги для створення гіперпосилань, таблиць та форм на вебсторінках

Мета лабораторної роботи. *Навчитися створювати вебсторінки з використанням таблиць, форм, гіперпосилань та інтерактивних елементів.*

Теоретичні відомості

Інтерактивні теги HTML

Інтерактивні теги HTML дозволяють створювати елементи, з якими користувач може взаємодіяти, наприклад, гіперпосилання, форми, кнопки, таблиці тощо.

Теги для створення гіперпосилань

Гіперпосилання (або посилання) дозволяють переходити на інші сторінки, сайти або ресурси. Основний тег для створення гіперпосилання — це <a>.

Синтаксис:

Текст посилання

Параметри:

- href — адреса, на яку переходить користувач.
- target — вказує, де відкрити посилання. Значення _blank відкриває посилання у новій вкладці.

Приклад:

Відвідати сайт

Теги для створення таблиць

Таблиці використовуються для організації даних у вигляді рядків і стовпців. Основні теги для роботи з таблицями:

- <table> — створює таблицю.
- <tr> — створює рядок таблиці.
- <th> — створює заголовок стовпця.
- <td> — створює комірку таблиці.
- colspan — об'єднує комірки по горизонталі.
- rowspan — об'єднує комірки по вертикалі.

Синтаксис:

```
<table border="1">
  <tr>
    <th colspan="2">Заголовок таблиці</th>
  </tr>
  <tr>
    <td>Комірка 1</td>
    <td>Комірка 2</td>
  </tr>
</table>
```

Приклад таблиці зі стилями:

```
<table style="border-collapse: collapse; width: 100%;">
  <tr>
    <th style="border: 1px solid #000; padding: 10px;">Назва</th>
    <th style="border: 1px solid #000; padding: 10px;">Опис</th>
    <th style="border: 1px solid #000; padding: 10px;">Ціна</th>
  </tr>
  <tr>
    <td style="border: 1px solid #000; padding: 10px;">Товар 1</td>
    <td style="border: 1px solid #000; padding: 10px;">Опис товару 1</td>
    <td style="border: 1px solid #000; padding: 10px;">100 грн</td>
  </tr>
```

</table>

Теги для створення форм

Форми використовуються для збору даних від користувачів. Основний тег для створення форми — <form>. Елементи форми:

- <input> — текстові поля, паролі, кнопки тощо.
- <textarea> — багаторядкове текстове поле.
- <select> і <option> — випадаючий список.
- <button> — кнопка.
- <label> — мітка для елемента форми.
- <fieldset> і <legend> — групування елементів форми.

Синтаксис:

```
<form action="/submit" method="post">
  <label for="name">Ім'я:</label>
  <input type="text" id="name" name="name"><br><br>

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

  <button type="submit">Відправити</button>
</form>
```

Випадаючий список та багатокількісний вибір

- Випадаючий список створюється за допомогою тегів <select> та <option>.
- Багатокількісний вибір реалізується за допомогою атрибута multiple.

Приклад:

```
<select name="options">
  <option value="option1">Опція 1</option>
  <option value="option2">Опція 2</option>
</select>

<select name="multi_options" multiple>
  <option value="option1">Опція 1</option>
  <option value="option2">Опція 2</option>
</select>
```

Радіокнопки та чекбокси

- **Радіокнопки** дозволяють вибрати один із варіантів:

  ```
  <input type="radio" id="feedback" name="type" value="feedback">
  <label for="feedback">Відгук</label>
  ```

- **Чекбокси** використовуються для вибору кількох варіантів:

  ```
  <input type="checkbox" id="agreement" name="agreement">
  <label for="agreement">Згоден з умовами</label>
  ```

Гіперпосилання з зображенням

Для створення гіперпосилання з зображенням використовується тег <a> із вкладеним тегом <img>.

Приклад:

```
<a href="https://example.com">
   <img src="image.jpg" alt="Зображення">
</a>
```

Завдання до лабораторної роботи

До сторінки створеної на попередньому занятті додайте:

1. Створіть складну таблицю з 3 стовпцями та 5 рядками**.** Об'єднайте комірки у першому рядку для створення заголовка таблиці. Використайте заголовки стовпців для кожного стовпця. Об'єднайте комірки в одному з рядків як по горизонталі, так і по вертикалі. Застосуйте стилі до таблиці (наприклад, рамки, фон).

2. Створіть форму з полями для введення імені, електронної пошти та повідомлення. Додайте кнопку відправлення форми.

3. Створіть випадаючий список, а також продумайте і організуйте багатокількісний вибір**.**

4. Створіть групу радіокнопок для вибору типу повідомлення (наприклад, відгук, питання, скарга).

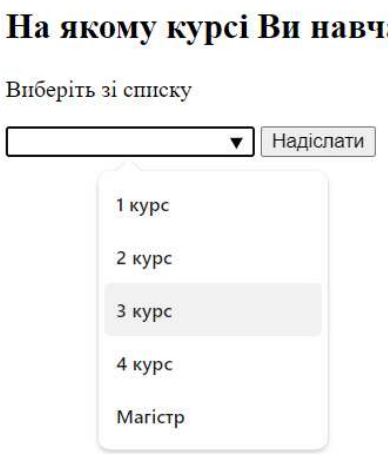
5. Додайте чекбокс для підтвердження згоди на обробку персональних даних.

6. Додайте гіперпосилання на зовнішню вебсторінку та гіперпосилання, яке відкривається у новій вкладці.

7. Додайте зображення з гіперпосиланням.

8. Створіть форму типу

9. Створіть список, як приклад



Запитання для самоперевірки

1. Які етапи потрібно виконати для створення складної таблиці з заголовками та об'єднанням комірок?
2. Як можна додати рамки та змінити фон комірок таблиці?
3. Як створити форму з полями для введення тексту та адреси електронної пошти?
4. Який тег використовується для створення багаторядкового текстового поля? Як його налаштувати?
5. Як можна створити групу радіокнопок? Який атрибут забезпечує взаємну виключність вибору?
6. Які елементи форми використовуються для збору згоди користувача (наприклад, підтвердження згоди на обробку персональних даних)?
7. Як додати гіперпосилання, що веде на зовнішню сторінку, та гіперпосилання, яке відкривається у новій вкладці?
8. Як можна зробити так, щоб при натисканні на зображення відбувався перехід за гіперпосиланням?
9. Який синтаксис HTML використовується для створення нумерованих та ненумерованих списків?
10. Як створити інтерактивний список, що дозволяє користувачу вибирати кілька пунктів одночасно?

Лабораторна робота №3.

Використання каскадних таблиць стилів CSS

Мета лабораторної роботи. Навчитися оформлювати вебсторінки з використанням каскадних таблиць стилів CSS.

Теоретичні відомості

Каскадні таблиці стилів (CSS) — це мова стилів, яка використовується для опису зовнішнього вигляду елементів HTML-документів. CSS дозволяє розділяти зміст (HTML) та оформлення, роблячи код більш структурованим, зрозумілим і гнучким для змін.

Підключення CSS до HTML

CSS може бути підключений до HTML трьома способами:

1. **Вбудований стиль (Inline styles).** Властивості стилю задаються безпосередньо в тегу HTML за допомогою атрибута style.

   Текст із вбудованим стилем

2. **Внутрішній стиль (Internal styles).** Стилі пишуться в секції <style> у заголовку документа.

   ```
   <style>
     p {
       color: blue;
       font-size: 18px;
     }
   </style>
   ```

3. **Зовнішній файл стилів (External stylesheet).** Стилі зберігаються у зовнішньому файлі .css і підключаються до HTML через тег <link>.

   <link rel="stylesheet" href="style.css">

Основна структура CSS

Синтаксис правила CSS:

```
селектор {
  властивість: значення;
}
```

Наприклад:

```
h1 {
  color: green;
  font-size: 24px;
  text-align: center;
}
```

Типи селекторів

1. **Базові селектори**
   - **Селектор тегів** застосовує стиль до всіх елементів певного типу.
     p { color: black; }
   - **Селектор класу** позначається крапкою (.), використовується для елементів із певним класом.
     .highlight { background-color: yellow; }
   - **Селектор ID** позначається символом решітки (#), застосовується до елемента з унікальним ідентифікатором.
     #header { font-size: 32px; }

2. **Комбіновані селектори**
    - **Селектор нащадків** застосовується до елементів, вкладених у певний контейнер.
      div p { color: gray; }

    - **Селектор групи** застосовується до кількох елементів одразу.
      h1, h2, h3 { font-family: Arial; }

3. **Псевдокласи та псевдоелементи**
    - — змінює стиль при наведенні курсора.
      button:hover { background-color: lightblue; }

    - **::before, ::after** — додають вміст перед або після елемента.
      h1::before { content: "★ "; color: gold; }

4. **Псевдокласи для позицій**
    - **:first-child** вибирає перший елемент серед дітей.
      li:first-child { font-weight: bold; }

    - **(n)** вибирає елемент на певній позиції.
      li:nth-child(2) { color: red; }

Структура зовнішнього CSS-файлу

При використанні зовнішнього файлу стилів, усі правила CSS зберігаються в окремому документі. Наприклад:

```
body {
  font-family: 'Arial', sans-serif;
  margin: 0;
  padding: 0;
}

h1 {
  color: navy;
  text-align: center;
}

.container ul li {
  color: #333;
  margin-bottom: 5px;
}
```

Організація стилів

1. Використовуйте зовнішні файли стилів для кращого структурного підходу.
2. Об’єднуйте CSS-правила для схожих елементів.
3. Використовуйте семантичні класи і ID для покращення зрозумілості коду.

Поліпшення дизайну за допомогою CSS

- **Типографіка:** налаштування шрифтів, міжрядкових інтервалів.
- **Кольорова гама:** встановлення кольорів тексту, фону, рамок.
- **Розташування:** використання відступів (margin, padding) і вирівнювання.
- **Анімації:** додавання динамічних ефектів за допомогою CSS-трансформацій або ключових кадрів.

CSS — це потужний інструмент, який дозволяє створювати привабливі, зручні для користувача вебсторінки. Розділення структури та стилю забезпечує гнучкість і полегшує підтримку вебпроєктів.

Завдання до лабораторної роботи

1. Змінити HTML-сторінку, розроблену в попередній роботі, замінивши всі можливі атрибути елементів, що використовуються для її оформлення, CSS-стилями.

2. Всі описані властивості CSS-стилів винести в окремий файл – style.css.

3. Додати додаткові властивості, щоб поліпшити загальний вигляд створеної HTML-сторінки.

4. Використайте базовий селектор для зміни стилю заголовка (h1). Наприклад, змініть колір тексту на синій, розмір шрифту на 24px.

5. Використайте селектор для зміни стилю параграфа (p) – змініть колір фону і тексту, шрифт та міжрядковий інтервал.

6. Створіть селектори класів та ID у CSS і призначте їм стилі (наприклад, зміна кольору, розміру шрифту, відступів).

7. Використайте комбіновані селектори (наприклад, div p, .container ul li) для зміни стилів внутрішніх елементів. Наприклад, змініть колір тексту всіх пунктів списку, які містяться всередині певного контейнера.

8. Створіть селектор, який застосовуватиметься до всіх елементів, які містять спільний клас і належать до одного типу (наприклад, всі параграфи з класом "important").

9. Використайте псевдоклас :hover для зміни стилю кнопки при наведенні курсора (зміна кольору фону, тіні або інших ефектів).

10. Додайте псевдоелемент ::before або ::after для додавання контенту перед або після тексту елемента. Наприклад, додайте декоративний символ перед заголовками.

11. Використайте селектори псевдокласів (:first-child, :nth-child()) для стилізації перших або певних елементів у списку.

12. Продемонструвати виконане завдання викладачу і завантажте до системи moodle.

Запитання для самоперевірки

1. Що таке каскадні таблиці стилів (CSS), і яку функцію вони виконують у веброзробці?
2. Які способи підключення CSS до HTML ви знаєте? Наведіть приклади.
3. Що таке селектор у CSS? Які типи селекторів існують?

4. Як описується правило CSS? Яка структура його синтаксису?
5. У чому різниця між селекторами класу (.classname) та ID (#idname)?
6. Як використовується селектор групи? Наведіть приклад.
7. Як задати колір тексту та фону за допомогою CSS?
8. Як змінити шрифт тексту та його розмір?
9. Які властивості відповідають за відступи, рамки та поля елементів?
10. Як змінити стиль кнопки при наведенні курсора на неї?
11. Що таке комбіновані селектори? Наведіть приклад використання селектора нащадків.
12. Як працює псевдоклас :hover, і де його доцільно застосовувати?
13. У чому різниця між псевдокласами та псевдоелементами? Наведіть приклади їх використання.

Лабораторна робота №4.

Створення вебсторінки з використанням блочної структури, семантичних тегів HTML5 та різних методів позиціонування

Мета лабораторної роботи. *Ознайомитися з основними концепціями блочної структури вебсторінки, семантичної верстки HTML5, а також закріпити навички використання різних методів позиціонування елементів у CSS.*

Теоретичні відомості

Блочна структура вебсторінки

Блочна структура – це основа сучасної веброзробки, яка базується на організації сторінки як набору окремих блоків (секцій). Кожен блок виконує певну функцію і може бути оформлений індивідуально.

Переваги блочної структури:

- Полегшує читання та модифікацію коду.
- Забезпечує повторне використання елементів.
- Сприяє створенню адаптивних і гнучких макетів.

Семантична верстка HTML5

Семантична верстка передбачає використання спеціалізованих тегів HTML5 для створення структури вебсторінки, які вказують на значення контенту.

Основні семантичні теги HTML5:

- <header> – шапка сторінки (логотип, назва сайту, меню).
- <nav> – блок для навігаційного меню.
- <main> – основний контент сторінки (ключова інформація).
- <aside> – додатковий контент (реклама, посилання).
- <footer> – підвал сайту (контакти, авторські права).
- <section> – тематичний розділ сторінки.
- <article> – самостійний блок, який можна відокремити (стаття, пост).

Семантична верстка покращує SEO, доступність сайту і його підтримку.

Методи позиціонування у CSS

CSS пропонує кілька способів позиціонування елементів:

Статичне (static) – стандартне позиціонування за порядком у коді.
Відносне (relative). Елемент розміщується відносно своєї початкової позиції. Використовується для створення локальної системи координат.
Абсолютне (absolute). Елемент позиціонується відносно найближчого предка з позиціонуванням (relative, absolute або fixed).
Фіксоване (fixed). Елемент прикріплений до вікна браузера, навіть при прокрутці сторінки.

Розміщення елементів за допомогою Flexbox та Grid

Flexbox (Flexible Box).
Використовується для вирівнювання та розташування елементів у ряд або стовпець. Основні властивості: display: flex;, justify-content, align-items.

Grid Layout. Сітковий макет для розташування елементів у двовимірному просторі. Основні властивості: display: grid;, grid-template-rows, grid-template-columns.

Адаптивний дизайн і медіазапити

Адаптивний дизайн забезпечує коректне відображення сторінки на різних пристроях (мобільних телефонах, планшетах, десктопах).

Медіазапити використовуються для застосування стилів залежно від ширини екрану.

```
@media (max-width: 768px) {
  body {
    font-size: 14px;
  }
}
```

Принципи адаптивного дизайну:

- Використання відносних одиниць (%, vw, vh) замість пікселів.
- Оптимізація зображень і медіа.
- Перетворення горизонтального меню на випадаюче для мобільних пристроїв.

Реалізація адаптивного меню

Для створення меню, яке змінює вигляд залежно від ширини екрану:

- Використовуйте Flexbox/Grid для базового макету.
- Застосуйте медіазапити для змінення структури (наприклад, приховування елементів і активація випадаючого списку).
- Для мобільних пристроїв використовуйте "гамбургер-меню".

Приклад структури HTML та CSS

HTML

```
<header>
  <h1>Мій сайт</h1>
  <nav>
    <ul>
      <li><a href="#">Головна</a></li>
      <li><a href="#">Про нас</a></li>
      <li><a href="#">Контакти</a></li>
    </ul>
  </nav>
</header>
<main>
  <article>Основний текст</article>
  <aside>Бічна панель</aside>
</main>
<footer>© 2024 Мій сайт</footer>
```

CSS

```
header {
  position: fixed;
  background: #333;
  color: white;
  width: 100%;
  top: 0;
}
nav ul {
  display: flex;
  justify-content: space-around;
  list-style: none;
}
main {
  display: flex;
  margin-top: 60px;
}
@media (max-width: 768px) {
  nav ul {
    flex-direction: column;
  }
}
```

Завдання до лабораторної роботи

Завдання 1. Створіть вебсторінку або удосконалити створену на попередніх заняттях.

Додайте наступні семантичні теги:

- <header> для шапки сайту, що містить логотип і назву сайту.
- <nav> для меню навігації з посиланнями на різні розділи сайту.
- <main> для основного контенту, що містить статтю або текстовий блок.

- <aside> для бічної панелі з додатковим контентом або корисною інформацією.
- <footer> для підвалу сайту з копірайтом та контактною інформацією.

Завдання 2. Удоскональте сторінку за допомогою CSS.

Використайте Flexbox або Grid для розміщення елементів сторінки у блочній моделі.

Застосуйте різні методи позиціонування для окремих блоків:

- Шапка сайту – фіксоване позиціонування (fixed).
- Навігація – відносне позиціонування (relative).
- Один з блоків основного контенту – абсолютне позиціонування (absolute).

Додайте стилі для шапки, контенту та бічної панелі, змінюйте кольори, розміри шрифтів, відступи тощо.

Завдання 3: Створіть адаптивний дизайн

- Створіть медіазапити для різних розмірів екранів:
- Забезпечте коректне відображення сторінки на різних пристроях.

Завдання 4: Реалізувати меню, яке змінює свій вигляд залежно від ширини екрана.

Запитання для самоперевірки

1. Що таке блочна структура вебсторінки, і які її переваги?
2. Які семантичні теги HTML5 ви використовували у своїй вебсторінці? Опишіть їх призначення.
3. У чому полягають переваги використання тегів <header>, <main>, <aside> та <footer> над звичайними ?
4. Які методи позиціонування у CSS існують, і як вони відрізняються один від одного?
5. Чим відрізняються відносне (relative) та абсолютне (absolute) позиціонування? Наведіть приклад.
6. Що таке Flexbox? Для яких задач він найкраще підходить?
7. Які властивості CSS ви використали для організації адаптивного дизайну вашої вебсторінки?
8. Як медіазапити допомагають створювати адаптивні вебдизайни? Наведіть приклад медіазапиту для мобільних пристроїв.
9. Як реалізувати меню, яке змінює свій вигляд залежно від ширини екрана? Опишіть основні етапи.
10. Що таке псевдоклас :hover, і як він може покращити взаємодію користувача зі сторінкою? Наведіть приклад.

Лабораторна робота №5.

Основи скриптової мови програмування JavaScript

Мета лабораторної роботи. Навчитися працювати з JavaScript, створювати та підключати зовнішні скрипти, використовувати функції для запитів у користувача та опрацьовувати різні типи даних.

Теоретичні відомості

JavaScript — це мова програмування, яка дозволяє створювати динамічний і інтерактивний контент на вебсторінках. Вона виконується у браузері та використовується для маніпулювання HTML і CSS.

Підключення скриптів

Внутрішній скрипт розміщується безпосередньо у HTML-документі в секції <script>:

```
<script>
  console.log("Це мій перший скрипт!");
</script>
```

Зовнішній скрипт підключається через атрибут src у тегі <script>:

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

Основні функції

- alert("Повідомлення") — відображає вікно з повідомленням.
- console.log("Текст") — виводить текст у консоль браузера.
- prompt("Питання") — запитує у користувача текстове значення.
- confirm("Питання") — відображає вікно підтвердження (OK/Cancel) і повертає true або false.

Змінні

У JavaScript використовуються три ключові слова для оголошення змінних:

- var — використовується рідко через застарілість.
- let — змінна, яку можна змінювати.
- const — змінна, значення якої не змінюється.

```
let name = "Олексій"; // Рядок
const age = 25; // Число
let isStudent = true; // Булевий тип
console.log(name, age, isStudent);
```

Типи даних

- **Примітивні типи** рядок (string), число (number), булевий тип (boolean), null, undefined, symbol, bigint.
- **Об'єкти** складні структури даних (наприклад, масиви, функції).

Умовні конструкції використовуються для виконання різних дій залежно від умов:

```
let age = prompt("Вкажіть ваш вік");
age = Number(age);
if (age >= 18) {
   console.log("Ви повнолітній");
} else {
   console.log("Ви неповнолітній");
}
```

Функція — це блок коду, який виконує певну задачу.

```
function greet(name) {
   alert(`Привіт, ${name}!`);
}
greet("Олексій");
```

Порядок виконання завдання

Внутрішній скрипт використовується для простих задач без окремого файлу. Розміщується у HTML-документі.

Зовнішній скрипт забезпечує краще структурування коду та полегшує його підтримку.

Для взаємодії з користувачем часто використовуються кнопки та події.

```
<button onclick="checkRole()">Чи ви студент?</button>
javascript
Копіювати код
function checkRole() {
   let isStudent = confirm("Ви студент?");
   if (isStudent) {
      alert("Ви студент");
   } else {
      alert("Ви не студент");
   }
}
```

Функція prompt() повертає текстове значення. Його можна конвертувати у число:

```
let age = Number(prompt("Введіть ваш вік"));
```

Використовуйте console.log() для діагностики та перевірки змінних.

Щоб підключити зовнішній файл, використовуйте:

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

Цей тег слід розміщувати перед закриттям тега <body>.

Завдання до лабораторної роботи

Завдання 1. Створити простий внутрішній скрипт

1. Створіть HTML-документ зі структурою:
   - Заголовок сторінки: **"Перша сторінка з JavaScript"**.
   - Текст на сторінці: **"Привіт, JavaScript!"**.
2. Усередині HTML-документа додайте внутрішній JavaScript (у секції <script>), який:
   - Виведе в консоль рядок "Це мій перший скрипт!".
   - Виведе alert-повідомлення з текстом "JavaScript приєднано успішно!".

Завдання 2. Приєднати зовнішній скрипт

1. Створіть HTML-документ, що матиме наступну структуру

- Заголовок сторінки **"Перевірка користувача"**.
- Текст на сторінці **"Натисніть кнопку для перевірки ролі"**.
- Додайте кнопку з текстом **"Чи ви студент?"**, при натисканні якої викликається JavaScript-функція, що використовує функцію confirm() для підтвердження, чи є користувач студентом.

2. Створіть зовнішній файл JavaScript (script.js), в якому потрібно

- Запитайте ім'я користувача за допомогою функції prompt() та виведіть привітальне повідомлення у форматі **"Привіт, [ім'я користувача]!"** за допомогою alert().
- Реалізуйте функцію для обробки кліку по кнопці, яка використовує функцію confirm(), щоб запитати у користувача **"Ви студент?"**. Якщо користувач підтверджує, виведіть повідомлення **"Ви студент"**. Інакше виведіть **"Ви не студент"**.
- Оголосіть змінні різних типів: рядок, число, булевий тип. Виведіть їх значення у консоль.
- Запитайте вік користувача через функцію prompt(), перетворіть введене значення на числовий тип та перевірте, чи є користувач повнолітнім, використовуючи умовний оператор if...else.

3. Виведіть у консоль наступну інформацію

- Текст: **"Ваш вік: [значення віку]"**.
- Якщо вік більше або дорівнює 18, виведіть **"Ви повнолітній"**.
- Якщо вік менше ніж 18, виведіть **"Ви неповнолітній"**.

3. **Приєднайте зовнішній файл script.js** до HTML-документа, використовуючи тег <script src="script.js"></script> перед закриттям тега <body>.

Запитання для самоперевірки

1. Що таке JavaScript і для чого він використовується у веброзробці?
2. Які способи підключення JavaScript-скриптів до HTML-документа ви знаєте? Наведіть приклади.
3. Яка різниця між внутрішнім та зовнішнім скриптом? У яких випадках краще використовувати кожен із них?
4. Як працює функція alert()? Наведіть приклад її використання.
5. Для чого використовуються функції prompt() і confirm()? У чому різниця між ними?
6. Що таке змінна у JavaScript? Чим відрізняються ключові слова let, const і var?
7. Які основні типи даних існують у JavaScript? Наведіть приклади їх використання.
8. Як виконати перевірку умови у JavaScript за допомогою конструкції if...else? Напишіть простий приклад.
9. Як оголосити та викликати функцію у JavaScript? Чим функція відрізняється від звичайного блоку коду?
10. Як підключити зовнішній файл JavaScript до HTML-документа? Чому важливо розташовувати тег <script> у кінці тега <body>?

Лабораторна робота №6.

Використання умовних операторів в JavaScript

Мета лабораторної роботи. *Навчитися використовувати умовні конструкції та оператори порівняння в JavaScript для перевірки різних умов і прийняття рішень в програмах. Засвоїти принцип роботи умовних операторів для реалізації різних логічних сценаріїв, що включають обробку чисел, рядків та взаємодію з користувачем.*

Теоретичні відомості

Основи умовних операторів

Оператор if Використовується для перевірки умов і виконання коду, якщо умова істинна.

```
if (умова) {
   // код, який виконається, якщо умова істинна
}
```

Приклад:

```
let age = 18;
if (age >= 18) {
   console.log("Ви повнолітній");
}
```

Оператор if...else Додає альтернативну гілку для виконання коду, якщо умова хибна.

```
if (умова) {
   // код, якщо умова істинна
} else {
   // код, якщо умова хибна
}
```

Приклад:

```
let score = 50;
if (score >= 60) {
   console.log("Тест пройдено");
} else {
   console.log("Тест не пройдено");
}
```

Оператор if...else if...else Дозволяє перевіряти кілька умов послідовно.

```
if (умова1) {
   // код для умови1
} else if (умова2) {
   // код для умови2
} else {
   // код, якщо жодна умова не виконується
}
```

Приклад:

```
let score = 85;
if (score >= 90) {
   console.log("Відмінно");
} else if (score >= 75) {
   console.log("Добре");
} else if (score >= 60) {
   console.log("Задовільно");
} else {
   console.log("Погано");
}
```

Тернарний оператор Коротка форма запису if...else.

```
умова ? вираз1 : вираз2;
```

Приклад:

```
let age = 20;
let status = age >= 18 ? "Повнолітній" : "Неповнолітній";
console.log(status);
```

Оператор switch Використовується для перевірки змінної на відповідність кільком можливим значенням.

```
switch (вираз) {
  case значення1:
    // код
    break;
  case значення2:
    // код
    break;
  default:
    // код за замовчуванням
}
```

Приклад:

```
let day = 3;
switch (day) {
  case 1:
    console.log("Понеділок");
    break;
  case 2:
    console.log("Вівторок");
    break;
  case 3:
    console.log("Середа");
    break;
  default:
    console.log("Невірний день");
}
```

Оператори порівняння

Рівність (==, ===)

- == перевіряє рівність значень, не враховуючи тип.
- === перевіряє рівність значень і типів.

```
console.log(5 == "5");  // true
console.log(5 === "5"); // false
```

Нерівність (!=, !==)

- != перевіряє нерівність значень, не враховуючи тип.
- !== перевіряє нерівність значень і типів.

Інші оператори порівняння

- Більше: >
- Менше: <
- Більше або дорівнює: >=
- Менше або дорівнює: <=

Логічні оператори

&& (логічне "І") Повертає true, якщо всі умови істинні.

```
console.log(true && true);  // true
console.log(true && false); // false
```

|| (логічне "АБО") Повертає true, якщо хоча б одна умова істинна.

```
console.log(true || false); // true
```

! (логічне "НЕ") Інвертує значення.

```
console.log(!true);  // false
```

Особливості роботи з умовами

1. JavaScript автоматично приводить типи для порівняння, що може спричинити неочікувані результати. Завжди використовуйте сувору рівність (===) і нерівність (!==).
2. Порожні значення (null, undefined, 0, NaN, "") розглядаються як хибні (false).

Завдання до лабораторної роботи

Виконати індивідуальне завдання відповідно до Вашого порядкового номера групи. Розробити програму, яка використовує умовні оператори для перевірки значень і виведення результатів на основі заданих умов.

Варіант 1. Калькулятор оцінок

Написати програму, яка запитує у користувача оцінку (число від 0 до 100) і виводить результат відповідно до шкали:

- 90-100 - "Відмінно"
- 75-89 - "Добре"
- 60-74 - "Задовільно"
- Менше ніж 60 - "Погано"

Варіант 2. Визначення парності числа

Написати програму, яка запитує у користувача ціле число і визначає:

- Чи є число парним або непарним.
- Якщо число парне, вивести "Число парне."
- Якщо число непарне, вивести "Число непарне."

Варіант 3. Перевірка логіна та пароля

Створити програму, яка запитує у користувача логін та пароль. Якщо логін дорівнює "admin", а пароль "12345", вивести повідомлення "Доступ дозволено". В іншому випадку вивести "Невірний логін або пароль."

Варіант 4. Визначення пори року

Програма запитує номер місяця (1-12) і виводить пору року:

- Зима - 12, 1, 2
- Весна - 3, 4, 5
- Літо - 6, 7, 8
- Осінь - 9, 10, 11

Варіант 5. Калькулятор днів у місяці

Написати програму, яка запитує у користувача номер місяця (1-12) і виводить кількість днів у цьому місяці. Врахувати, що лютий має 28 днів.

Варіант 6. Програма "Числові порівняння"

Створити програму, яка запитує у користувача два числа і виводить яке з них більше, менше або чи є вони рівними.

Варіант 7. Визначення категорії користувача за кількістю годин роботи

Програма запитує у користувача кількість годин, відпрацьованих за тиждень, і виводить відповідне повідомлення:

- Якщо менше ніж 20 годин - "Часткова зайнятість".
- Якщо від 20 до 40 годин - "Повна зайнятість".
- Якщо понад 40 годин - "Перевищення норми".

Варіант 8. Вибір дня тижня

Програма запитує у користувача число від 1 до 7 і виводить відповідний день тижня (1 - понеділок, 2 - вівторок, і т.д.).

Варіант 9. Визначення високосного року

Написати програму, яка запитує у користувача рік і перевіряє, чи є він високосним. Вивести повідомлення:

- Якщо рік високосний, вивести "Цей рік високосний."
- Якщо рік не високосний, вивести "Цей рік не високосний."

Варіант 10. Калькулятор з двома операціями

Програма запитує у користувача два числа та одну з операцій (+, -, *, /). Виконує обрану операцію над числами та виводить результат. Якщо користувач вводить некоректний оператор або спробу ділення на нуль, вивести відповідне повідомлення про помилку.

Варіант 11. Визначення знаку зодіаку

Написати програму, яка запитує дату народження користувача (день і місяць) і виводить його знак зодіаку. Використовувати умовні оператори для визначення знаку за діапазонами дат.

- Овен 21 березня – 19 квітня
- Телець 20 квітня – 20 травня
- І так далі для всіх знаків зодіаку.

Варіант 12. Обчислення вартості квитка

Програма запитує вік користувача та обчислює вартість квитка на транспорт за наступною схемою:

- Якщо вік до 6 років — квиток безплатний.
- Якщо вік від 6 до 18 років — вартість квитка 10 грн.
- Якщо вік від 18 до 60 років — вартість квитка 20 грн.
- Якщо вік понад 60 років — квиток безплатний.

Варіант 13. Визначення пароля по складності

Написати програму, яка запитує у користувача пароль і визначає його складність:

- Якщо пароль містить менше ніж 6 символів, вивести "Пароль занадто короткий."
- Якщо пароль містить тільки букви або тільки цифри, вивести "Пароль повинен містити як букви, так і цифри."
- Якщо пароль містить букви та цифри, вивести "Пароль прийнято."

Запитання для самоперевірки

1. Що таке умовний оператор у JavaScript, і які основні види умовних конструкцій існують?
2. Чим відрізняються оператори == і === у JavaScript? Наведіть приклад.
3. Як працює оператор if...else if...else? У яких випадках доцільно використовувати його замість кількох if?
4. Як використовувати оператор switch для перевірки кількох значень? Чим він відрізняється від if...else if...else?
5. Що таке тернарний оператор? Як він скорочує запис умов? Наведіть приклад.
6. Які оператори порівняння використовуються в JavaScript для перевірки числових або рядкових значень?
7. Як працюють логічні оператори &&, || та ! у JavaScript? Які результати вони повертають у різних випадках?

Лабораторна робота №7.

Масиви та робота з ними

***Мета лабораторної роботи**. Ознайомитися з основними операціями роботи з масивами у JavaScript. Навчитися створювати масиви, працювати з їх елементами, здійснювати пошук, фільтрацію та обчислення на основі масиву. Розвинути навички аналізу даних та написання ефективного коду для вирішення завдань з масивами.*

Теоретичні відомості

Масив — це структура даних, яка дозволяє зберігати множину елементів (значень), організованих за певним порядком. У JavaScript масиви є об'єктами і можуть містити значення будь-якого типу: числа, рядки, об'єкти, функції тощо.

Створення масиву

Масив можна створити кількома способами:

1. **Літерал масиву**

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

2. **Конструктор Array**

   ```
   let arr = new Array(5); // Масив з 5 порожніх елементів
   ```

Основні операції з масивами

1. **Доступ до елементів масиву**

   ```
   let arr = [10, 20, 30];
   console.log(arr[0]); // 10
   ```

2. **Додавання елементів**
   - В кінець масиву: arr.push(value)
   - На початок масиву: arr.unshift(value)
3. **Видалення елементів**
   - З кінця: arr.pop()
   - З початку: arr.shift()
4. **Довжина масиву**

   ```
   console.log(arr.length); // Кількість елементів
   ```

5. **Перебір елементів масиву:**
   - За допомогою циклу for:

     ```
     for (let i = 0; i < arr.length; i++) {
       console.log(arr[i]);
     }
     ```

- За допомогою for...of:

```
for (let value of arr) {
  console.log(value);
}
```

6. **Методи для роботи з масивами:**
   - **Сортування:** arr.sort((a, b) => a - b) — для чисел.
   - **Зворотний порядок:** arr.reverse().
   - **Об'єднання елементів:** arr.join(", ").
   - **Додавання елементів:** arr.concat([6, 7]).

Важливі методи масивів

1. **filter:** створює новий масив з елементів, що відповідають умові.

   ```
   let even = arr.filter(num => num % 2 === 0);
   ```

2. **map:** створює новий масив шляхом перетворення кожного елемента.

   ```
   let squares = arr.map(num => num ** 2);
   ```

3. **reduce:** застосовує функцію для обчислення єдиного значення.

   ```
   let sum = arr.reduce((total, num) => total + num, 0);
   ```

4. **find:** повертає перший елемент, що відповідає умові.

   ```
   let firstEven = arr.find(num => num % 2 === 0);
   ```

5. **indexOf:** повертає індекс першого входження значення або -1.

   ```
   let index = arr.indexOf(30);
   ```

Особливості роботи з масивами

1. Масиви в JavaScript є динамічними, їх розмір може змінюватися в процесі виконання програми.
2. Масиви можуть містити значення різних типів:

   ```
   let arr = [1, "hello", true, null];
   ```

3. Масиви мають числову індексацію, яка починається з 0.

Приклади використання масивів

1. **Пошук найбільшого числа**

   ```
   let max = Math.max(...arr);
   ```

2. **Сортування за зростанням**

```
arr.sort((a, b) => a - b);
```

3. **Перевірка наявності дубліката**

```
let hasDuplicates = arr.length !== new Set(arr).size;
```

4. **Поділ масиву на частини**

```
let firstHalf = arr.slice(0, arr.length / 2);
let secondHalf = arr.slice(arr.length / 2);
```

Масиви є основою для роботи з даними у багатьох програмах. Їхнє ефективне використання дозволяє обробляти великі набори даних і реалізовувати складні алгоритми.

Завдання до лабораторної роботи

Виконати індивідуальне завдання відповідно до Вашого порядкового номера групи.

1. Створити масив з 10 випадкових цілих чисел від 1 до 100. Вивести на екран усі парні числа з масиву. Обчислити кількість парних та непарних чисел у масиві.

2. Створити масив з 15 випадкових чисел. Знайти та вивести найбільше та найменше число з масиву. Знайти суму всіх чисел, що більші середнього значення масиву.

3. Створити масив із заданою кількістю елементів (кількість запитати у користувача). Перевірити, чи є в масиві дублікати елементів, і вивести їх, якщо такі є. Видалити всі дублікати і вивести результат.

4. Створити масив з 20 випадкових цілих чисел. Вивести всі елементи масиву у зворотному порядку. Відсортувати масив за зростанням і вивести результат.

5. Створити масив із 10 елементів. Поміняти місцями перший і останній елемент масиву. Вивести змінений масив.

6. Створити масив з 12 випадкових чисел. Вивести всі числа, які більше за середнє значення елементів масиву. Знайти кількість чисел, які є кратними 5.

7. Створити масив із цілих чисел. Знайти перше число, яке більше за 50, і вивести його індекс. Видалити всі елементи масиву, які менші за 20, і вивести результат.

8. Створити масив із 10 випадкових чисел. Обчислити різницю між найбільшим та найменшим елементами масиву. Замінити всі від'ємні числа масиву на їхні значення по модулю.

9. Створити масив із 15 елементів. Знайти кількість елементів, які є простими числами. Вивести всі прості числа з масиву.

10. Створити масив із випадкових чисел. Знайти кількість елементів, які є кратними як 3, так і 7. Видалити всі елементи, що не є кратними 3, і вивести новий масив.

11. Створити масив з 10 випадкових чисел. Вивести масив на екран. Знайти та вивести на екран максимальне та мінімальне числа з масиву. Обчислити середнє значення всіх елементів масиву.

12. Створити масив з 10 випадкових цілих чисел. Створити новий масив, який містить квадрати чисел з оригінального масиву. Вивести обидва масиви на екран.

13. Створити масив з 20 випадкових чисел. Вивести масив на екран, розділивши його на дві рівні частини. Поміняти місцями першу і другу частину масиву.

Запитання для самоперевірки

1. Які способи створення масиву ви знаєте? Наведіть приклади.
2. Як отримати доступ до елементів масиву? Як змінити значення конкретного елемента?
3. Що таке властивість .length у масиві та як її використовують?
4. Які методи додавання та видалення елементів з масиву існують? Наведіть приклади їх використання.
5. Чим відрізняються методи map, filter та reduce? У яких ситуаціях їх застосовують?
6. Як знайти найбільший і найменший елементи масиву?
7. Як перевірити наявність дублікатів у масиві та видалити їх?
8. Що таке метод sort і як виконати сортування масиву чисел за зростанням?
9. Як розділити масив на дві рівні частини? Наведіть приклад коду.
10. Що таке метод indexOf і чим він відрізняється від методу find?
11. Як можна обчислити суму елементів масиву за допомогою методу reduce?
12. Як вивести масив у зворотному порядку? Який метод для цього використовують?
13. Що таке метод slice і як його використовують? Чим він відрізняється від методу splice?
14. Як додати всі елементи масиву до нового масиву без використання циклів?
15. Як видалити всі елементи масиву, які менші за певне значення?
16. Які особливості роботи з масивами різних типів даних у JavaScript?
17. Що відбувається, якщо у масиві є пропущені індекси? Як це впливає на перебір елементів?
18. Як працює оператор розпакування ... (spread) з масивами?
19. Які переваги використання масивів у порівнянні з іншими структурами даних?

Лабораторна робота №8-9.

Основи скриптової мови програмування JavaScript: об’єкти, методи для роботи з об’єктами. Обробка даних форми

Мета лабораторної роботи. *Ознайомлення з основними принципами роботи з об’єктами в JavaScript. Навчитися створювати об’єкти, працювати з їх властивостями та методами, а також опрацьовувати дані форм для взаємодії з об’єктами.*

Теоретичні відомості

Об'єкти у JavaScript

Об'єкти є однією з основних структур даних у JavaScript. Вони дозволяють зберігати значення у вигляді пар "ключ-значення". Об'єкт можна створити за допомогою літерального синтаксису {} або конструктора new Object().

Приклад створення об'єкта:

```
const car = {
   make: "Toyota",
   model: "Camry",
   year: 2020,
   color: "red",
   mileage: 15000
};
```

Доступ до властивостей об'єкта

До властивостей об'єкта можна звертатися двома способами

Через крапкову нотацію:

```
console.log(car.make); // Toyota
```

Через квадратні дужки:

```
console.log(car['model']); // Camry
```

Методи об'єкта

Методи об'єкта – це функції, які є його властивостями. Вони визначаються як функції в середині об'єкта:

```
const car = {
   make: "Toyota",
   model: "Camry",
   year: 2020,
   color: "red",
   mileage: 15000,
   getCarInfo: function() {
      return `Марка: ${this.make}, Модель: ${this.model}, Рік: ${this.year}, Колір: ${this.color},
Пробіг: ${this.mileage} км`;
   }
};
console.log(car.getCarInfo());
```

Оновлення властивостей об'єкта

Властивості об'єкта можна змінювати через крапкову нотацію:

```
car.color = "blue";
console.log(car.color); // blue
```

Форми та взаємодія з DOM

HTML-форма дозволяє користувачеві вводити дані, які можна обробляти через JavaScript. Для доступу до елементів форми використовують методи:

- document.getElementById()
- document.querySelector()
- document.querySelectorAll()

Приклад:

```
<form id="carForm">
   <input type="text" id="make" placeholder="Марка">
   <input type="text" id="model" placeholder="Модель">
   <input type="number" id="year" placeholder="Рік випуску">
   <input type="text" id="color" placeholder="Колір">
   <input type="number" id="mileage" placeholder="Пробіг">
   <button type="button" id="update">Оновити</button>
</form>
<div id="carInfo"></div>
javascript
Копіювати код
document.getElementById('update').addEventListener('click', function() {
   car.make = document.getElementById('make').value;
   car.model = document.getElementById('model').value;
   car.year = parseInt(document.getElementById('year').value);
   car.color = document.getElementById('color').value;
   car.mileage = parseInt(document.getElementById('mileage').value);
   document.getElementById('carInfo').innerText = car.getCarInfo();
});
```

Робота з prompt та кнопками

Метод prompt() дозволяє отримати дані від користувача через діалогове вікно:

```
const newColor = prompt("Введіть новий колір:");
car.updateColor(newColor);
```

Методи для розрахунків

Методи age() та isOld() виконують розрахунки:

```
const car = {
   year: 2015,
   age: function() {
      return new Date().getFullYear() - this.year;
   },
   isOld: function() {
      return this.age() > 10;
   }
};
console.log(car.age()); // Виведе вік
console.log(car.isOld()); // true або false
```

Валідація даних

Перевірка даних форми перед обробкою:

- Перевірка на заповненість:

```
if (!make || !model || !year || !color || !mileage) {
  alert("Заповніть всі поля!");
  return;
}
```

- Перевірка року:

```
if (year > new Date().getFullYear()) {
  alert("Рік випуску не може бути більшим за поточний!");
  return;
}
```

Динамічне оновлення сторінки

Для оновлення інформації у DOM використовуються методи:

- innerHTML або innerText
- Створення нових елементів через document.createElement().

Ці знання є базовими для роботи з об'єктами та інтерактивністю в JavaScript.

Завдання до лабораторної роботи

Завдання 1. Створити об'єкт car, який містить такі властивості

- make (марка автомобіля, наприклад, "Toyota").
- model (модель автомобіля, наприклад, "Camry").
- year (рік випуску, наприклад, 2020).
- color (колір автомобіля, наприклад, "red").
- mileage (пробіг автомобіля в км, наприклад, 15000).

Дані можна вказувати довільні.

Завдання 2. Додати наступні методи

Додати метод getCarInfo(), який повертає рядок з повною інформацією про автомобіль у форматі: Марка: Toyota, Модель: Camry, Рік випуску: 2020, Колір: red, Пробіг: 15000 км

Додати метод updateColor(newColor), який приймає параметр newColor та змінює колір автомобіля.

Додати метод updateMileage(newMileage), який приймає параметр newMileage та оновлює пробіг автомобіля.

Завдання 3. Створити HTML-сторінку з формою, що дозволяє вводити нові значення для властивостей автомобіля:

- Марка.
- Модель.
- Рік випуску.
- Колір.
- Пробіг.

Додати кнопку "Оновити дані", яка оновлює значення об'єкта car на основі введених користувачем даних та відображає оновлену інформацію про автомобіль на сторінці.

Завдання 4. Реалізувати наступні кнопки

- Реалізувати кнопку "Показати інформацію", яка відображає інформацію про автомобіль у HTML-елементі на сторінці.
- Додати кнопки для зміни кольору та пробігу автомобіля за допомогою prompt.

Завдання 5. Додати метод age(), який розраховує вік автомобіля (різниця між поточним роком та роком випуску).

Додати метод isOld(), який визначає, чи старіший автомобіль за 10 років.

Реалізувати валідацію форми, перевіряти, щоб всі поля були заповнені коректно (наприклад, рік випуску не може бути більшим за поточний рік).

Запитання для самоперевірки

1. Як отримати доступ до властивостей об'єкта? Чим відрізняються крапкова нотація та квадратні дужки?
2. Як додати нову властивість до існуючого об'єкта? Наведіть приклад.
3. Що таке метод об'єкта? Як його можна викликати?
4. Як у методі об'єкта звернутися до його властивостей? Що таке ключове слово this?
5. Як можна змінити значення властивості об'єкта?
6. Для чого використовують метод prompt()? Як за його допомогою оновити властивості об'єкта?
7. Як можна реалізувати функцію, що обчислює вік автомобіля на основі його року випуску?
8. Як перевірити, чи автомобіль старіший за 10 років? Який тип даних поверне така перевірка?
9. Як створити HTML-форму для введення даних і використати її для оновлення об'єкта?
10. Що таке валідація форми? Як перевірити, чи всі поля форми заповнені коректно?
11. Як вивести інформацію про об'єкт у HTML-документ? Які методи для цього використовуються?
12. Що робить метод document.getElementById()? Наведіть приклад його використання.
13. Як додати на сторінку кнопку, яка виконує певну функцію?
14. Які переваги використання об'єктів у JavaScript для моделювання реальних об'єктів?

Лабораторна робота №10-11.

Поняття DOM-структури документа. Навігація в DOM

Мета лабораторної роботи. *Ознайомити студентів із принципами структурування документа через DOM та навчити основам роботи з елементами HTML за допомогою JavaScript, включаючи навігацію в DOM-дереві, зміну стилів елементів та їх взаємне позиціювання.*

Теоретичні відомості

Поняття DOM-структури документа

Document Object Model (DOM) — це об'єктна модель документа, яка відображає HTML-структуру як дерево елементів. Усі елементи документа (теги, текст, коментарі тощо) є вузлами дерева, якими можна керувати за допомогою JavaScript. DOM дозволяє:

- Змінювати структуру HTML-документа.
- Додавати, видаляти, змінювати елементи та їх атрибути.
- Реагувати на події, такі як кліки, зміни тексту або завантаження сторінки.

Основні типи вузлів:

- **Елементи (Element):** Теги HTML, такі як , <p>, <table>.
- **Текстові вузли (Text):** Текст між тегами.
- **Атрибути (Attributes):** Властивості елементів, такі як id, class.
- **Коментарі (Comment):** Коментарі HTML.

Навігація в DOM

Для навігації та доступу до елементів використовуються такі методи та властивості:

Доступ до вузлів

- document.getElementById(id) — отримує елемент за ідентифікатором.
- document.getElementsByTagName(tag) — повертає колекцію елементів за ім’ям тега.
- document.getElementsByClassName(className) — вибирає елементи за класом.
- document.querySelector(selector) — вибирає перший елемент, що відповідає CSS-селектору.
- document.querySelectorAll(selector) — вибирає всі елементи, що відповідають CSS-селектору.

Навігація між вузлами

- element.parentNode — батьківський вузол.
- element.children — дочірні елементи (тільки вузли-елементи).
- element.firstChild, element.lastChild — перший/останній дочірній вузол.
- element.nextSibling, element.previousSibling — сусідні вузли.

- element.nextElementSibling, element.previousElementSibling — сусідні вузли-елементи.

Зміна стилів і контенту елементів

Зміна контенту

- element.innerHTML — встановлює або отримує HTML-вміст елемента.
- element.textContent — змінює текст без HTML-розмітки.

Зміна стилів

- element.style.property — змінює CSS-стиль елемента.
  Наприклад:

  element.style.color = 'red';
  element.style.backgroundColor = 'blue';

Додавання/видалення класів

- element.classList.add('className') — додає клас.
- element.classList.remove('className') — видаляє клас.
- element.classList.toggle('className') — перемикає клас.

Робота з таблицями

Для роботи з таблицями DOM надає зручні інструменти:

- Вибір елементів <table>, <tr>, <td> і маніпуляції з ними.
- Створення таблиць через document.createElement('tag').
- Зміна стилів, заповнення клітинок текстом або HTML.

Робота з атрибутами

- element.getAttribute('attribute') — отримання атрибута.
- element.setAttribute('attribute', 'value') — встановлення атрибута.
- element.removeAttribute('attribute') — видалення атрибута.

Події та їх обробка

Події в DOM дозволяють реагувати на дії користувача:

- Клік (onclick).
- Наведення миші (onmouseover, onmouseout).
- Завантаження сторінки (onload).
  Для обробки подій використовується:
- Встановлення обробника через HTML-атрибути або JavaScript:

```
element.onclick = function() {
  alert('Подія натискання!');
```

```
};
```

Часові функції

- setInterval(function, milliseconds) — виконує функцію з заданою періодичністю.
- setTimeout(function, milliseconds) — виконує функцію один раз через вказаний час.
- clearInterval(id) — зупиняє виконання функції, запущеної через setInterval.

Створення сповіщень

За допомогою DOM можна створювати динамічні елементи на сторінці, наприклад, сповіщення:

- Створити елемент через document.createElement('div').
- Додати його в DOM через appendChild або insertAdjacentHTML.
- Налаштувати стилі через style або класи CSS.

Завдання до лабораторної роботи.

Всі завдання можете додати до сторінки, що була створена раніше, однак потрібно додати елементи, яких вимагатиме завдання.

Завдання 1. Напишіть код, щоб зафарбувати всі клітинки, що знаходяться на діагоналях квадратної таблиці, зеленим кольором. Створіть таблицю 5x5 із координатами в кожній клітинці (як у прикладі).

Використайте наступний код для зафарбування клітинки:
td.style.backgroundColor = 'green';

Результат повинен бути таким:

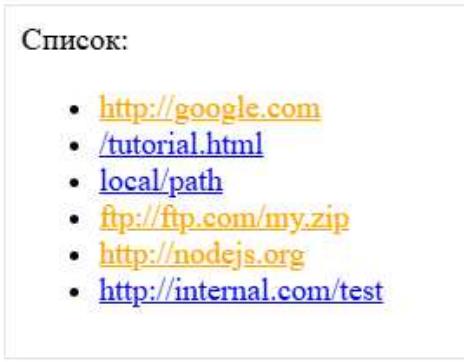
1:1	2:1	3:1	4:1	5:1
1:2	2:2	3:2	4:2	5:2
1:3	2:3	3:3	4:3	5:3
1:4	2:4	3:4	4:4	5:4
1:5	2:5	3:5	4:5	5:5

Завдання 2. Зробіть всі зовнішні посилання іншим кольором (жовтий, помаранчевий, зелений, тощо), змінюючи властивість style. Створіть на сторінці кілька посилань, але саме зовнішні мають відображатися іншим кольором.

Посилання є зовнішнім, якщо:

- У ньому присутній :// у властивості href
- Але воно не починається з https://mysite.com.

Приклад результату:



Завдання 3. Розробити функцію createCalendar(elem, year, month) для генерації календаря місяця Вашого народження в певному DOM-елементі. Результатом має стати таблиця, яка містить дні тижня та числові значення дат.

Календар має бути таблицею, де тиждень це <tr>, а день це <td>. Перший рядок таблиці має бути оформлений як <th> з назвами днів тижня: першим днем має бути понеділок і так до неділі.

Наприклад:

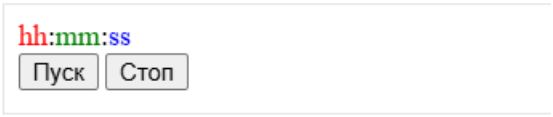
ПН	ВТ	СР	ЧТ	ПТ	СБ	НД
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Вказівки до виконання:

1. Створити заголовок таблиці з <th> та днями тижня.
2. Створити об’єкт дати d = new Date(year, month-1). Це перший день місяця month (беручи до уваги, що місяці в JavaScript починаються з 0, а не з 1).
3. Декілька перших клітинок до d.getDay() можуть бути порожніми. Заповнити їх <td></td>.
4. Збільшити день d: d.setDate(d.getDate()+1). Якщо d.getMonth() ще не досяг наступного місяця, тоді додати нову клітинку <td> в календар. Якщо це НД, тоді додати новий рядок “</tr><tr>”.
5. Якщо місяць завершився, але рядок таблиці ще не заповнений, додати порожні <td>, щоб надати таблиці правильної форми.

Завдання 4. Створити годинник, який буде автоматично оновлюватися кожну секунду. Годинник має відображати поточний час у форматі години:хвилини:секунди. Використовуйте HTML/CSS для стилізації, JavaScript лише оновлення часу в елементах.

Приклад:



Вказівки до виконання:

Функція update оновлюватиме годинник, а метод setInterval викликатиме її щосекунди.

```
function update() {
  let clock = document.getElementById('clock');
  let date = new Date(); // (*)
  // Відображення годин:
  let hours = date.getHours();
  if (hours < 10) hours = '0' + hours;
  clock.children[0].innerHTML = hours;
  // Аналогічно для хвилин та секунд
```

Функція для управління годинником:

```
let timerId;
function clockStart() { // увімкнути годинник
  if (!timerId) { // встановити новий інтервал, якщо годинник не увімкнений
    timerId = setInterval(update, 1000);
  }
  update();
}
function clockStop() {
  clearInterval(timerId);
  timerId = null;
}
```

Завдання 5. Написати функцію showNotification(options), яка створює сповіщення з переданим певним вмістом на сторінці. Сповіщення повинно зникати через певний проміжок часу.

Вхідні параметри такі:

```
// показує елемент з вашим текстом " Ваш варіант" біля правого верхнього кутка вікна
showNotification({
  top: 10, // 10px від верха вікна (усталено має бути 0px)
  right: 10, // 10px від правого краю вікна (усталено — 0px)
  html: "Якийсь текст", // HTML-код сповіщення
  className: "welcome" // додатковий клас для елемента div (необов'язково)
});
```

Запитання для самоперевірки

1. Що таке DOM, і як воно пов’язане з HTML-документом?
2. Назвіть основні типи вузлів DOM і поясніть їх призначення.

3. Які методи JavaScript використовуються для отримання елементів DOM? Наведіть приклади.
4. Як можна змінити текстовий вміст та HTML-код елемента? У чому різниця між innerHTML та textContent?
5. Як працює classList і які операції з класами він дозволяє виконувати?
6. Що таке події в DOM? Як створити обробник подій?
7. Які методи можна використовувати для роботи з таблицями в DOM?
8. Як можна створити та додати новий елемент у DOM-дерево?
9. Які властивості елемента використовуються для зміни його стилю?
10. Як працюють функції setInterval і setTimeout? Чим вони відрізняються?

Лабораторна робота №12-13

Події в JavaScript. Основи роботи з мишею та клавіатурою.

Мета лабораторної роботи. *Засвоїти основи роботи з подіями в JavaScript, зокрема опрацювання подій миші та клавіатури. Розвинути навички створення інтерактивних елементів на вебсторінках, що реагують на дії користувача, навчитися використовувати події click, mouseover, keydown, keyup, mousedown, mousemove, mouseup та їхні обробники для реалізації інтерактивної поведінки елементів.*

Теоретичні відомості

Події в JavaScript

Події в JavaScript – це механізм, за допомогою якого браузер сповіщає про те, що щось сталося (наприклад, клік мишею, натискання клавіші або завершення завантаження сторінки). Реагування на події дозволяє створювати інтерактивність на вебсторінках.

Обробка подій

Обробники подій – це функції, які виконуються у відповідь на певну подію. Їх можна додати до елементів за допомогою атрибутів HTML або JavaScript.

Методи додавання обробників подій

HTML-атрибути

```
<button onclick="alert('Hello!')">Click me</button>
```

Через властивість елемента

```
button.onclick = function() {
  alert('Hello!');
};
```

Через метод addEventListener (рекомендований)

```
button.addEventListener('click', function() {
  alert('Hello!');
});
```

Типи подій

Події миші

- click – клік на елемент.
- dblclick – подвійний клік.
- mousedown – натискання кнопки миші.
- mouseup – відпускання кнопки миші.
- mousemove – переміщення курсору миші.
- mouseover – курсор увійшов у зону елемента.
- mouseout – курсор вийшов із зони елемента.

Події клавіатури

- keydown – натискання клавіші.
- keyup – відпускання клавіші.
- keypress – натискання клавіші (застаріла, не використовується в сучасних браузерах).

Інші події

- scroll – прокручування сторінки.
- focus/blur – отримання/втрата фокусу на елементі.
- submit – відправлення форми.

Основи роботи з подіями миші

Для подій миші використовується об'єкт MouseEvent, який містить корисну інформацію:

- button – яка кнопка натиснута (ліва, права або середня).
- clientX, clientY – координати курсору щодо області перегляду.
- pageX, pageY – координати курсору щодо документа.

Приклад обробки події mousemove:

```
document.addEventListener('mousemove', (event) => {
  console.log(`Курсор: ${event.clientX}, ${event.clientY}`);
});
```

Основи роботи з подіями клавіатури

Об'єкт KeyboardEvent надає доступ до властивостей:

- key – символ натиснутої клавіші.
- code – фізичне розташування клавіші на клавіатурі.
- altKey, ctrlKey, shiftKey, metaKey – логічні значення, які визначають, чи були натиснуті відповідні модифікатори.

Приклад обробки події keydown:

```
document.addEventListener('keydown', (event) => {
   console.log(`Натиснуто клавішу: ${event.key}`);
});
```

Поширення подій (Event Propagation)

Події в DOM проходять три фази:

1. **Фаза захоплення (Capturing Phase):** подія йде від кореневого елемента до цільового.
2. **Фаза таргетування (Target Phase):** подія досягає цільового елемента.
3. **Фаза спливання (Bubbling Phase):** подія поширюється вгору по дереву від цільового елемента до кореневого.

Для зупинки спливання можна використовувати метод event.stopPropagation().

Методи для обробки подій

1. preventDefault() – запобігає дії за замовчуванням (наприклад, відправці форми).
2. stopPropagation() – зупиняє поширення події вгору або вниз по DOM.
3. removeEventListener() – видаляє обробник подій.

Практичні аспекти

1. **Перемикання класів:**
   Подія click часто використовується для додавання/видалення класів:

   ```
   element.classList.toggle('active');
   ```

2. **Прокрутка сторінки:**
   Подія scroll дозволяє створювати динамічні ефекти (наприклад, кнопки "вгору").
3. **Графічні компоненти:**
   Використання подій дозволяє створювати каруселі, інтерактивні меню, сповіщення тощо.

Цей теоретичний матеріал забезпечить розуміння основних концепцій роботи з подіями в JavaScript і підготує до виконання практичних завдань.

Завдання до лабораторної роботи.

Завдання 1. Напишіть код, щоб після натискання на кнопку <button>, елемент зникав.

Завдання 2. Створіть меню, яке відкривається або згортається після натискання.

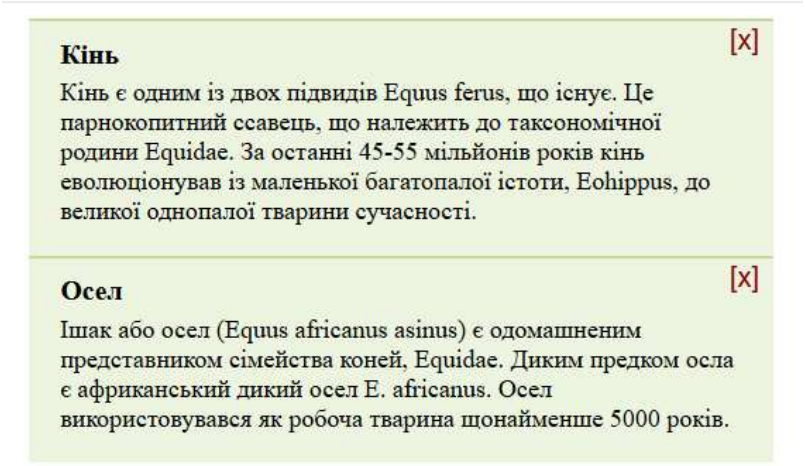
Наприклад:

Функціонал перемикання меню повинен змінювати стрілку та приховувати або показувати список елементів меню.

Вказівки. Всі ці зміни можна реалізувати засобами CSS. За допомогою JavaScript потрібно змінювати вигляд меню, додаючи або видаляючи клас .open.

Завдання 3. Додати кнопки закриття для повідомлень. Реалізуйте код з використанням JavaScript для кожного повідомлення додати у верхній правий кут кнопку для його закриття.

Результат має виглядати, як показано тут:



Вказівки. Щоб додати кнопку закриття, можна використовувати або position:absolute (і зробити плитку (pane) position:relative) або float:right. Перевага варіанта з float:right у тому, що кнопка закриття ніколи не перекриє текст, але варіант position:absolute дає більше свободи для дій. Загалом вибір за вами.

Тоді для кожного повідомлення(pane) код може бути таким:

pane.insertAdjacentHTML("afterbegin", '<button class="remove-button">[x]</button>');

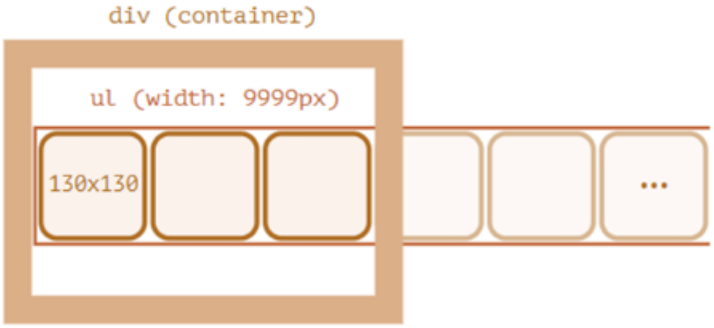
Елемент <button> стає pane.firstChild, таким чином ми можемо додати до нього обробник події:

pane.firstChild.onclick = () => pane.remove();

Завдання 4. Створити “Карусель зображень” – стрічку зображень, яку можна прокручувати, натискаючи на стрілки.

Вказівки. Стрічка зображень може бути представлена як список ul/li з картинками <img>.

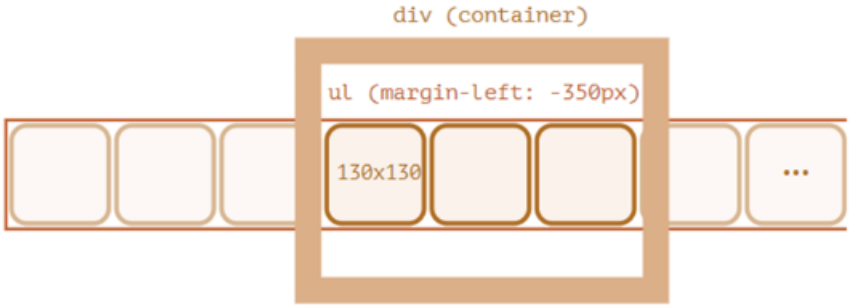
Потрібно розташувати стрічку всередині фіксованого розміру, так щоб одночасно була видна тільки потрібна частина списку:



Щоб список зробити горизонтальним, потрібно застосувати CSS-властивість display: inline-block для <li>.

Для тегу <img> потрібно налаштувати display, оскільки за умовчанням він є inline. У всіх елементах типу inline резервується додаткове місце під “хвости” символів. І щоб його забрати, потрібно прописати display:block.

Для прокручування потрібно переміщати <ul>. Це можна робити по-різному, наприклад, призначенням CSS-властивості transform: translateX() (краще для продуктивності) або margin-left:



У зовнішнього фіксована ширина, тому зайві зображення обрізаються.

Вся карусель – це самостійний «графічний компонент» на сторінці, таким чином краще його «обернути» в окремий і вже модифікувати стилі всередині нього.

Завдання 5. Створіть функцію runOnKeys(func, code1, code2, ... code_n), яка запускає func при одночасному натисканні клавіш із кодами code1, code2, …, code_n.

Наприклад, код нижче показує alert, коли "Q" та "W" натискаються разом (будь-якою мовою, з або без CapsLock)

```
runOnKeys(
  () => alert("Привіт!"),
  "KeyQ",
  "KeyW"
);
```

Вказівки. Потрібно використовувати два обробники: document.onkeydown і document.onkeyup.

Створити набір pressed = new Set(), щоб зберегти поточні натиснуті клавіші.

Перший обробник додає до нього, а другий видаляє. Кожного разу при keydown перевіряється, чи достатньо натиснутих клавіш, і запускається функція, якщо це так.

Завдання 6. Створіть кнопку “вгору”, щоб допомогти прокручувати сторінку.

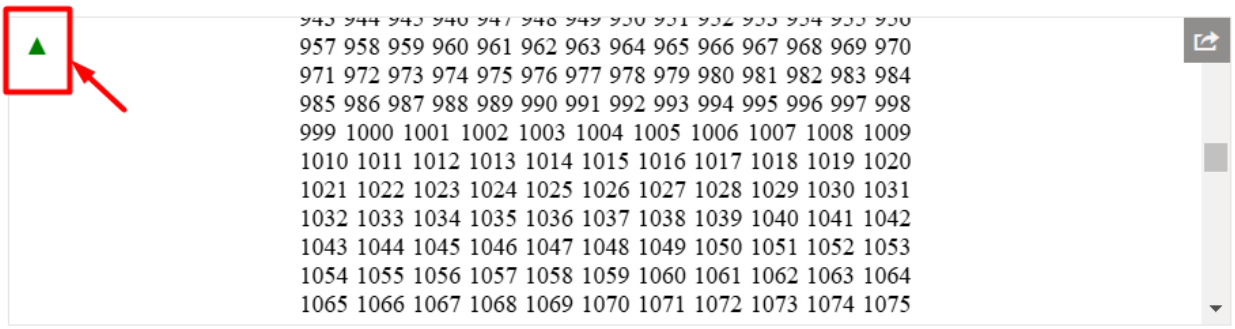
Вона має працювати так:

· Поки сторінка не прокручена принаймні на висоту вікна – вона невидима.

· Якщо сторінка прокручена нижче висоти вікна, у верхньому лівому куті з’являється стрілка “вгору”. Якщо сторінку прокрутити назад, вона зникне.

· При натисканні стрілки сторінка прокручується вгору.

Ось так (верхній лівий кут, прокрутіть, щоб побачити):



Запитання для самоперевірки

1. Що таке події в JavaScript, і для чого вони використовуються?
2. Які є способи додавання обробників подій до елемента? Поясніть їх переваги та недоліки.
3. Що таке фази поширення подій у DOM? У чому різниця між фазою захоплення і фазою спливання?
4. Як за допомогою JavaScript зупинити поширення події?
5. Що таке event.preventDefault()? Наведіть приклад його використання.
6. Чим корисний метод addEventListener порівняно з додаванням обробників через властивість onclick?
7. Які події миші ви знаєте, і як вони використовуються?
8. Чим події mouseover і mouseenter відрізняються одна від одної?

9. Які властивості об’єкта MouseEvent надають інформацію про координати курсора? У чому різниця між clientX і pageX?
10. Як реалізувати подію для відображення координат курсора миші в реальному часі?
11. Які події клавіатури існують, і яка між ними різниця?
12. Що таке event.key і event.code, і як вони використовуються?
13. Як визначити, чи натиснуто клавішу-модифікатор (Shift, Ctrl, Alt)?
14. Як реалізувати функцію, яка реагує на одночасне натискання кількох клавіш?

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Лаптон Е., Коул Філліпс Дж. Графічний дизайн: Нові основи. Київ, ArtHuss, 2020. 264 с.
2. Робін Вільямс. Дизайн. Книга для недизайнерів. Простою мовою про засади графічного дизайну. Харків, Vivat, 2022. 240 с.
3. Бородкіна І.Л., Бородкін Г.О. Web-технології та Web-дизайн : застосування мови HTML для створення електронних ресурсів. Київ, Ліра-К, 2020. 212 с.
4. HTML Підручник. Початок. Уроки для початківців. W3Schools українською *W3SchoolsUA.українською*. URL: https://w3schoolsua.github.io/html/index.html.
5. CSS Підручник. Уроки для початківців. W3Schools українською *W3SchoolsUA.українською.* URL: https://w3schoolsua.github.io/css/index.html#gsc.tab=0
6. HTML і CSS довідник українською. URL: https://html-css.co.ua/.
7. Фрімен Е., Робсон Е. Head First. Програмування на JavaScript. Харків, Фабула, 2022. 672 с.
8. Сучасний підручник з JavaScript. *JAVASCRIPT.INFO.* URL: https://uk.javascript.info/.
9. Яцюк С. М., Юнчик В.Л. Web-дизайн. Безпека Web-ресурсів та додатків: навчальний посібник. Луцьк: ПП Іванюк, 2021. 316 с.
10. Yunchyk V. Fedoniuk Y. (2023). Results of developing the recommendation system for electronic educational resource selection. Manažérska informatika: vedecký časopis o informatike. URL: https://manazerskainformatika.sk/results-of-developing-the-recommendation-system-for-electronic-educational-resource-selection/.
11. Pasichnyk V., Kunanets N., Yunchyk V., Khomyak M., Fedonuyk A. Project of an Educational Content Evaluation Recommender System. Proceedings of the 5th International Workshop IT Project Management (ITPM 2024), Vol. 3709, P. 192-203. (Scopus)
12. Fedonuyk A., Yunchyk V., Mukutuyk I., Duda O., Yatsyuk S. Application of the hierarchy analysis method for the choice of the computer mathematics system for the IT-sphere specialists preparation. Journal of Physics: Conference Series. 2021. Vol. 1840(1), 012065 (Scopus)
13. Яцюк, С., Хомяк, М., Юнчик, В., & Чепрасова, Т. (2021). Особливості навчання веб-технологій розробки навчальних систем майбутніх вчителів інформатики та методика створення на їх основі власних освітніх ресурсів. Молодь і ринок, (7/193).
14. Юнчик, В. Л., Швейгер, Н. Р., Хоменко, В. Ю., & Ковальчук, М. С. (2022). Проектування електронного освітнього ресурсу для навчання математики. In *Математика. Інформаційні технології. Освіта*. Волинський національний університет імені Лесі Українки.

УДК 658.512.23:004.4(075)

Електронне мережне навчальне видання

Юнчик Валентина Леонідівна

ТЕХНОЛОГІЇ ВЕБРОЗРОБКИ

Методичні рекомендації до виконання лабораторних робіт
для здобувачів освіти спеціальності
125 Кібербезпека та захист інформації
першого (бакалаврського) рівня

Видання друкується в авторській редакції