**Практична робота №6 Java-script**

JavaScript (JS) - це мова програмування, яка дозволяє додавати інтерактивність на веб-сайти. Вона є однією з трьох основних технологій Всесвітньої павутини, поряд з HTML та CSS. JavaScript використовується для створення динамічних веб-сторінок, що включають анімацію, прокрутку сторінок, відтворення медіа, ігри та інтерактивні форми.

**Основні Принципи JavaScript**

1. **Переменні та Типи Даних:**
	* **Переменні:** **let**, **const**, **var** (старий спосіб оголошення).
	* **Типи даних:** числа (**number**), рядки (**string**), логічні значення (**boolean**), об'єкти (**object**), масиви, функції, **undefined**, **null**.
2. **Оператори:**
	* Арифметичні: **+**, **-**, **\***, **/**, **%** (залишок від ділення), **++** (інкремент), **--** (декремент).
	* Порівняння: **==**, **===**, **!=**, **!==**, **<**, **>**, **<=**, **>=**.
	* Логічні: **&&** (і), **||** (або), **!** (не).
3. **Умовні Конструкції:**
	* **if**, **else if**, **else**.
	* Тернарний оператор: **умова ? вираз1 : вираз2**.
4. **Цикли:**
	* **for**, **while**, **do...while**.
	* Методи обходу масивів: **forEach**, **map**, **filter**, **reduce**.
5. **Функції:**
	* Оголошення: **function functionName() { ... }**.
	* Стрілкові функції: **(параметри) => { ... }**.
6. **Об'єкти та Масиви:**
	* Літерали об'єктів: **{ ключ: значення }**.
	* Масиви: **[елемент1, елемент2, ...]**.
7. **DOM Маніпуляції:**
	* Вибір елементів: **document.getElementById()**, **document.querySelector()**, **document.querySelectorAll()**.
	* Модифікація властивостей: **елемент.textContent**, **елемент.innerHTML**, **елемент.style**.
8. **Події:**
	* Додавання обробників подій: **елемент.addEventListener('event', function)**.
	* Розповсюджені події: **click**, **mouseover**, **mouseout**, **submit**, **load**.

**Структура Скрипта JavaScript**

1. **Підключення Скрипта:**
	* В HTML: **<script src="script.js"></script>** або вставка коду безпосередньо у тег **<script> ... </script>**.
2. **Виконання Коду:**
	* Код виконується зверху вниз.
	* Функції можна визначати перед їх використанням, оскільки JavaScript виконує "підняття" (hoisting).
3. **Коментарі:**
	* Однорядкові: **// коментар**.
	* Багаторядкові: **/\* коментар \*/**.
4. **Debugging:**
	* Використання **console.log()** для виведення даних на консоль браузера.

**Базові принципи та кроки для створення простого скрипта:**

**1. Визначення Цілі Скрипта**

Перш за все, необхідно чітко визначити, що саме ви хочете, щоб ваш скрипт робив. Наприклад, може бути завдання змінити текст елемента при натисканні на кнопку або розрахувати суму покупки у формі.

**2. Оголошення Змінних**

Оголосіть змінні, які будуть використовуватися у вашому скрипті. Використовуйте **let** або **const** для оголошення змінних. Наприклад:

let count = 0; const price = 100;

**3. Написання Функцій**

Створіть функції, які будуть виконувати основні дії скрипта. Наприклад:

function incrementCount() { count += 1; }

**4. Взаємодія з HTML (DOM)**

Якщо скрипт пов'язаний з веб-сторінкою, вам може знадобитися взаємодіяти з елементами на сторінці. Використовуйте методи DOM для доступу та маніпуляції елементами. Наприклад:

let messageElement = document.getElementById('message');

**5. Обробка Подій**

Додайте обробники подій для реагування на дії користувача, наприклад, кліки по кнопках.

document.getElementById('myButton').addEventListener('click', incrementCount);

**6. Логіка Скрипта**

Включіть логіку, яка визначає, як ваш скрипт реагує на різні умови та входи. Наприклад:

if (count > 10) { messageElement.innerText = 'Ліміт перевищено'; }

Приклад скрипта:

*«let count = 0;*

*function incrementCount() {*

 *count += 1;*

 *document.getElementById('countDisplay').innerText = count;*

*}*

*document.getElementById('incrementButton').addEventListener('click', incrementCount);»*

**Розбір Коду**

1. **Оголошення Змінної**

let count = 0;

* + **let** - це ключове слово для оголошення змінної. Воно дозволяє змінювати значення змінної у майбутньому.
	+ **count** - ім'я змінної.
	+ **0** - початкове значення змінної **count**. Ця змінна використовуватиметься для зберігання кількості кліків.
1. **Функція**

function incrementCount() { count += 1; document.getElementById('countDisplay').innerText = count; }

* + **function** - ключове слово для оголошення функції.
	+ **incrementCount** - назва функції, яка описує її дію: збільшення змінної **count**.
	+ **count += 1;** - це дія збільшення значення змінної **count** на 1 кожен раз, коли викликається функція.
	+ **document.getElementById('countDisplay').innerText = count;** - знаходить елемент у HTML з ідентифікатором **countDisplay** і змінює його текстовий вміст на поточне значення **count**.
1. **Обробка Подій**

document.getElementById('incrementButton').addEventListener('click', incrementCount);

* + **document.getElementById('incrementButton')** - ця частина коду вибирає елемент HTML з ідентифікатором **incrementButton**.
	+ **addEventListener('click', incrementCount);** - додає обробник події до вибраного елемента. У цьому випадку, коли користувач клікає по кнопці (**click**), викликається функція **incrementCount**.

**Загальна Логіка**

Цей скрипт дозволяє користувачам клікати на кнопку, кожен клік збільшує лічильник (**count**). Значення лічильника відображається в елементі з ідентифікатором **countDisplay**.

Кожен раз, коли виконується функція **incrementCount**, **count** збільшується на 1, і оновлений результат виводиться на екран.

Це простий приклад взаємодії JavaScript з HTML-елементами та відображення динамічно змінюваних даних на веб-сторінці.

У контексті JavaScript, **incrementCount** є назвою функції. Слово "increment" у загальному означає "збільшити" або "додати одиницю", а "Count" у даному випадку можна перекласти як "рахунок" або "лічильник". Таким чином, **incrementCount** може бути перекладено як "збільшити лічильник" або "збільшити рахунок".

У прикладі коду, який я надав раніше, функція **incrementCount** використовується для збільшення значення змінної **count** на одиницю кожного разу, коли вона викликається. Це типовий приклад функції, яка використовується в JavaScript для збільшення або зменшення числового значення, що відслідковує певну кількість (наприклад, кількість кліків по кнопці).

Один з простих елементів використання є повідомлення:

<html>

<body>

 <p>Текст перед скриптом...</p>

 <script>

 alert( 'Привіт, світ!' );

 </script>

 <p>...Після скрипта.</p>

</body>

</html>

Функція **alert** у JavaScript використовується для відображення простого модального вікна з повідомленням для користувача. Це один із найпростіших способів вивести повідомлення на екран у веб-браузері.

Як Використовувати **alert**

1. **Базове Використання:**
	* Синтаксис: **alert(message);**
	* **message** - це текст повідомлення, яке буде відображено користувачу.
	* Приклад: **alert('Це просте повідомлення!');**
2. **Використання Зі Змінними:**
	* Ви можете використовувати змінні для динамічного створення тексту повідомлення.
	* Приклад:

let userName = 'Олексій'; alert('Вітаємо, ' + userName + '!');

1. **Комбінування з Іншими Функціями:**
	* Ви можете використовувати **alert** разом з іншими функціями для відображення результатів або значень.
	* Приклад:

function greetUser(name) { alert('Привіт, ' + name + '!'); } greetUser('Анна');

1. **Умовне Використання:**
	* **alert** може бути використано в умовних конструкціях для відображення повідомлень залежно від певних умов.
	* Приклад:

let score = 75; if (score > 50) { alert('Ви пройшли тест!'); } else { alert('Спробуйте ще раз.'); }

Рекомендації та Обмеження

* **Використання в Навчанні:** **alert** чудово підходить для простих демонстрацій та навчальних цілей.
* **Обмеження в Інтерфейсі:** Слід уникати частого використання **alert** у продуктивних додатках, оскільки воно блокує взаємодію користувача з веб-сторінкою, поки вікно не буде закрито.
* **Стилізація:** **alert** не дозволяє стилізувати текст або використовувати складніші елементи інтерфейсу. Для більш складних завдань краще використовувати модальні вікна, створені за допомогою HTML/CSS і JavaScript.

**alert** є одним з найпростіших і найшвидших способів для відображення інформації, але його використання має бути обмежене сценаріями, де простота є ключовим фактором.

Ви можете використати функцію **setTimeout**. Ця функція дозволяє відкласти виконання коду на вказаний період часу.

setTimeout(function() {

 alert('Це повідомлення з’явиться через 15 секунд');

}, 15000); // 15000 мілісекунд = 15 секунд

У цьому коді:

* **setTimeout** - це вбудована функція JavaScript, яка приймає два аргументи: функцію, яка виконується після затримки, та час затримки в мілісекундах.
* **function()** - це анонімна функція, яка буде виконана після закінчення вказаної затримки.
* Усередині анонімної функції викликається **alert** з текстом повідомлення.
* **15000** - це час затримки у мілісекундах. У цьому випадку 15000 мілісекунд дорівнює 15 секундам.

Ви можете розмістити цей код усередині тега **<script>** у вашому HTML-документі або у зовнішньому JavaScript-файлі, який підключений до вашої HTML-сторінки. Як тільки код буде виконаний (наприклад, при завантаженні сторінки), повідомлення **alert** з'явиться через 15 секунд.

**Завдання:**

Створіть просте повідомлення з будь якою функцією, поясніть її вагомість для вашої веб сторінки

**Додаткова література:**

Підручник Джава Скрипт <https://uk.javascript.info/hello-world>