**Web контрольні запитання**

**Lab 1:**

**1)** Якими способами можливо підключити JavaScript код до HTML сторінки?

Вбудований скрипт: <script>...JavaScript code...</script> в HTML.

Зовнішній файл: <script src="path/to/script.js"></script>.

**2)** Як ми можемо отримати доступ до елементів DOM за допомогою JS?

Використовуючи методи: getElementById(), getElementsByClassName(), getElementsByTagName(), querySelector(), querySelectorAll().

**3)** Що таке document.querySelector та чим він відрізняється від document.querySelectorAll?

querySelector повертає перший елемент, який відповідає CSS-селектору.

querySelectorAll повертає всі елементи, що відповідають CSS-селектору, у вигляді NodeList.

**4)** Які методи доступу до DOM елементів повертають колекцію?

getElementsByClassName(), getElementsByTagName(), querySelectorAll().

**5)** Що таке івенти HTML і як їх використовувати?

Івенти (події) - це дії, які відбуваються у браузері. Використовуються з допомогою атрибутів в HTML (onclick, onchange, тощо) або методів в JS (addEventListener).

**6)** Що таке функція у JS і її аргументи?

Функція - це блок коду, який можна виконати, викликавши його. Аргументи - це значення, які передаються в функцію під час її виклику.

**Lab 2:**

**1)** Які можливості дають оператори порівняння в JavaScript?

Оператори порівняння (==, ===, !=, !==, <, >, <=, >=) дозволяють порівнювати значення та визначати логічні відношення між ними.

**2)** Що таке строга рівність у JavaScript? Чому вона важлива?

=== - оператор строгої рівності порівнює як значення, так і тип даних. Важливий для уникнення непередбачуваних результатів.

**3)** Які логічні оператори використовуються для поєднання умов в JavaScript? Наведіть приклади.

Логічні оператори: && (і), || (або), ! (не).

Приклад: if (a > 0 && b < 10) { ... }.

**4)** Що таке каскадні умовні оператори? Як вони використовуються?

Каскадні умовні оператори (if, else if, else) дозволяють перевіряти кілька умов послідовно.

**5)** Що таке тернарний оператор в JavaScript? Як він використовується для короткого запису умовних виразів?

Тернарний оператор (? :) використовується для короткого запису умов: умова ? вираз1 : вираз2.

**6)** Як ви вирішили б складну умовну задачу, де потрібно оцінити студента за кількома критеріями? Які конструкції JavaScript ви використовуватимете?

Використав би каскадні умовні оператори або логічні оператори для поєднання умов:

if (criteria1 && criteria2 || criteria3) {

// action

} else {

// alternative action

}

**Lab 3:**

**1)** Що таке цикл while? Дайте приклад його використання.

Цикл while виконує блок коду доти, доки умова є істинною.

let i = 0;

while (i < 5) {

console.log(i);

i++;

}

**2)** В чому різниця між циклом while та циклом for в JavaScript?

for використовується для циклів з відомою кількістю ітерацій; while - для невідомої кількості ітерацій до виконання певної умови.

**3)** Коли краще використовувати конструкцію switch замість послідовних if-else умов?

Коли є багато умов з одним значенням змінної:

switch (day) {

case 1:

console.log("Monday");

break;

// інші випадки

default:

console.log("Invalid day");

}

**4)** Які переваги мають стрілкові функції порівняно зі звичайними функціями в JavaScript?

Не мають власного this, що зручно для вкладених функцій.

Коротший синтаксис:

const add = (a, b) => a + b;

**5)** Що таке область видимості змінних (scope) в JavaScript? Які типи областей видимості існують?

Область видимості визначає видимість змінних:

Глобальна

Функціональна

Блокова (let, const)

**6)** Що таке рекурсія? Які переваги та недоліки вона має?

Рекурсія - функція викликає сама себе.

Переваги: зручна для алгоритмів, які розбиваються на підзадачі (факториал, Фібоначчі).

Недоліки: може викликати переповнення стека викликів, складність розуміння.

**Lab 4:**

**1)** Які методи роботи з масивами в JavaScript ви знаєте і наведіть приклади їх використання?

push(), pop(), shift(), unshift(), splice(), slice(), forEach(), map(), filter(), reduce().

let arr = [1, 2, 3];

arr.push(4); // [1, 2, 3, 4]

arr.pop(); // [1, 2, 3]

**2)** Які є примітивні типи даних? Які ви знаєте методи примітивів в JavaScript?

Примітивні типи: number, string, boolean, null, undefined, symbol, bigint.

Методи: toFixed(), toString(), charAt(), charCodeAt(), substring().

**3)** Як ви можете додати новий елемент в кінець масиву в JavaScript?

Використовуючи метод push():

let arr = [1, 2, 3];

arr.push(4); // [1, 2, 3, 4]

**4)** Як ви можете видалити останній елемент з масиву в JavaScript?

Використовуючи метод pop():

let arr = [1, 2, 3];

arr.pop(); // [1, 2]

**5)** Як ви можете впорядкувати елементи масиву в JavaScript?

Використовуючи метод sort():

let arr = [3, 1, 2];

arr.sort(); // [1, 2, 3]

**6)** Які методи дозволяють видаляти та додавати елементи в середину масиву в JavaScript?

Використовуючи метод splice():

let arr = [1, 2, 3];

arr.splice(1, 1, 'new'); // [1, 'new', 3]

**7)** Як ви можете перевірити, чи містить масив певний елемент в JavaScript?

Використовуючи метод includes():

let arr = [1, 2, 3];

arr.includes(2); // true

**8)** Як ви можете знайти позицію певного елемента в масиві в JavaScript?

Використовуючи метод indexOf():

let arr = [1, 2, 3];

arr.indexOf(2); // 1

**9)** Як ви можете об'єднати два масиви в JavaScript?

Використовуючи метод concat() або оператор розподілу:

let arr1 = [1, 2];

let arr2 = [3, 4];

let combined = arr1.concat(arr2); // [1, 2, 3, 4]

let combinedSpread = [...arr1, ...arr2]; // [1, 2, 3, 4]

**10)** Як ви можете видалити властивість з об'єкту в JavaScript?

Використовуючи оператор delete:

let obj = {a: 1, b: 2};

delete obj.b; // {a: 1}

**Lab 5:**

**1)** Що таке об'єкт Date у JavaScript і які методи він надає для роботи з датою та часом?

Об'єкт Date представляє дату і час. Методи: getDate(), getDay(), getFullYear(), getHours(), getMinutes(), getSeconds(), getMilliseconds(), getTime(), setDate(), setFullYear(), setHours(), тощо.

**2)** Яка різниця між методами setTimeout та setInterval у JavaScript? Для чого вони використовуються?

setTimeout виконує функцію один раз після затримки.

setInterval виконує функцію повторно через заданий інтервал часу.

**3)** Що таке Map та Set у JavaScript? Яка різниця між ними?

Map зберігає пари ключ-значення, де ключ може бути будь-яким типом.

Set зберігає унікальні значення будь-якого типу.

**4)** Які основні операції можна виконувати з даними, що зберігаються в Map та Set?

Map: set(), get(), delete(), has(), clear().

Set: add(), delete(), has(), clear().

**5)** Що таке WeakMap та WeakSet у JavaScript? Які особливості їхнього використання порівняно з Map та Set?

WeakMap зберігає слабкі посилання на ключі (тільки об'єкти), які можуть бути збирані збирачем сміття.

WeakSet зберігає лише об'єкти і також використовує слабкі посилання.

**6)** В чому основні відмінності між Map та WeakMap?

Map зберігає сильні посилання на ключі.

WeakMap зберігає слабкі посилання, що дозволяє збирачу сміття видаляти непотрібні об'єкти.

**7)** Як можна використовувати setTimeout та setInterval для виконання певних завдань з певною затримкою чи інтервалом?

setTimeout для одноразових затримок:

setTimeout(() => console.log('Hello'), 1000);

setInterval для повторюваних завдань:

setInterval(() => console.log('Hello'), 1000);

**8)** Які сценарії використання WeakMap та WeakSet можна запропонувати для оптимізації роботи вашого додатку?

Використання для зберігання додаткових даних про об'єкти без ризику утримання пам'яті, оскільки ці об'єкти можуть бути автоматично видалені збирачем сміття.

**Lab 6:**

**1)** Що таке DOM, і як він відрізняється від HTML?

DOM (Document Object Model) - програмний інтерфейс для HTML, який представляє структуру документа у вигляді дерева об'єктів. HTML - це мова розмітки.

**2)** Які методи доступу до елементів DOM в JavaScript ви знаєте?

getElementById(), getElementsByClassName(), getElementsByTagName(), querySelector(), querySelectorAll().

**3)** Що таке вузол (node) в DOM?

Вузол - це базовий будівельний блок DOM. Вузли можуть бути елементами, текстом, атрибутами тощо.

**4)** Які методи пошуку елементів в DOM ви знаєте, і що вони роблять?

getElementById(): повертає елемент за ID.

getElementsByClassName(): повертає колекцію елементів за класом.

getElementsByTagName(): повертає колекцію елементів за тегом.

querySelector(): повертає перший елемент за CSS-селектором.

querySelectorAll(): повертає всі елементи за CSS-селектором.

**5)** Як можна змінювати вміст елементів DOM за допомогою JavaScript?

Використовуючи властивість innerHTML або textContent:

document.getElementById('myElement').innerHTML = 'New content';

**6)** Як можна додавати атрибути до елементів DOM за допомогою JavaScript?

Використовуючи метод setAttribute() або властивість:

document.getElementById('myElement').setAttribute('data-test', 'value');

document.getElementById('myElement').dataTest = 'value';

**7)** Як ви можете змінити стилі елементів DOM за допомогою JavaScript?

Використовуючи властивість style:

document.getElementById('myElement').style.color = 'red';

**8)** Які способи існують для додавання та видалення класів у елементів DOM?

Використовуючи методи classList.add(), classList.remove(), classList.toggle(), classList.contains():

let el = document.getElementById('myElement');

el.classList.add('newClass');

el.classList.remove('oldClass');

**el.classList.toggle('toggleClass');**

**9)** Що таке координати елемента в DOM, і як ви можете отримати їх за допомогою JavaScript?

Координати визначають положення елемента на сторінці. Отримати їх можна за допомогою методів getBoundingClientRect() або offsetTop, offsetLeft:

let rect = document.getElementById('myElement').getBoundingClientRect();

console.log(rect.top, rect.left);

**10)** Як можна визначити розмір елемента та його прокрутку за допомогою JavaScript?

Використовуючи властивості offsetWidth, offsetHeight, scrollWidth, scrollHeight:

let el = document.getElementById('myElement');

console.log(el.offsetWidth, el.offsetHeight);

console.log(el.scrollWidth, el.scrollHeight);

**11)** Як можна отримати розміри та прокрутку вікна браузера за допомогою JavaScript?

Використовуючи властивості window.innerWidth, window.innerHeight, window.scrollX, window.scrollY:

console.log(window.innerWidth, window.innerHeight);

console.log(window.scrollX, window.scrollY);

**12)** Що таке затримка затвору (event throttling) і як вона використовується для оптимізації роботи з подіями в DOM?

Затримка затвору - це техніка обмеження кількості викликів функції протягом певного часу. Використовується для оптимізації обробки подій, що часто викликаються (напр., scroll або resize):

function throttle(fn, limit) {

let inThrottle;

return function() {

let context = this, args = arguments;

if (!inThrottle) {

fn.apply(context, args);

inThrottle = true;

setTimeout(() => inThrottle = false, limit);

}

}

}

**Lab 7:**

**1)** Що таке розмір і прокрутка елемента, і як це впливає на веб-розробку?

Розмір елемента визначає його ширину та висоту. Прокрутка визначає, наскільки користувач прокрутив елемент. Важливо для створення адаптивних та зручних інтерфейсів.

**2)** Як можна отримати розміри вікна та прокрутку сторінки у JavaScript?

Використовуючи властивості window.innerWidth, window.innerHeight, window.scrollX, window.scrollY.

**3)** Які існують координатні системи у веб-розробці, і як можна отримати координати елементів у відношенні до цих систем?

Відносні до вікна (clientX, clientY), відносні до документа (pageX, pageY). Метод getBoundingClientRect() повертає координати відносно вікна.

**4)** Які можливості надає знання про розміри, прокрутку та координати для розробки користувацького інтерфейсу та взаємодії з користувачем на веб-сторінці?

Дозволяє створювати адаптивні дизайни, відстежувати прокрутку для завантаження контенту (infinite scroll), позиціювати елементи на сторінці.

**5)** Що таке події браузера і яка їх роль у веб-розробці?

Події браузера (наприклад, click, load, scroll, resize) - це сигнали про дії користувача чи зміни в браузері. Вони дозволяють створювати інтерактивні веб-сторінки.

**6)** Які основні типи подій існують у браузері?

Мишачі (click, dblclick, mousemove), клавіатурні (keydown, keyup), події форми (submit, change), події вікна (load, resize).

**7)** Що таке "бульбашковий механізм" у контексті подій браузера?

"Бульбашковий механізм" (event bubbling) - це процес, при якому подія спочатку обробляється найглибшим елементом, а потім піднімається вгору по DOM-дереву.

**8)** Що таке делегування подій і як воно використовується для оптимізації обробки подій?

Делегування подій дозволяє прикріпити обробник подій до батьківського елемента, а не до кожного дочірнього елемента, що зменшує кількість обробників і підвищує продуктивність.

**9)** Які типові дії браузера можуть викликати події?

Клік миші, натискання клавіш, завантаження сторінки, зміна розміру вікна, прокрутка сторінки.

**10)** Як можна ініціювати користувацькі події у JavaScript?

Використовуючи метод dispatchEvent():

let event = new Event('customEvent');

document.getElementById('myElement').dispatchEvent(event);