Державний університет інтелектуальних технологій і зв’язку

Факультет: Інформаційних технологій та кібербезпеки

Кафедра: Комп’ютерних наук

**Протокол лабораторної роботи №4**

**з дисципліни**

**«Веб-технології»**

Виконав студент групи ІПЗ-3.02

Жигальов Володимир Олександрович

Перевірив ст. викл. каф. КН

Северин М.В.

Одеса - 2025

**Лабораторна робота № 4**

**Тема:** Події в JavaScript.

**Мета роботи:** набуття практичного досвіду роботи з подіями та елементами веб-сторінки в JavaScript.

**Завдання**

**1**. Створити скрипт, що забезпечує роботу слайдера сайту з можливістю прокручування слайдів ліворуч та праворуч. При досягненні останнього слайду — виконувати показ з початку (гортання слайдів по колу).

*Рішення:*

Для реалізації слайдера було створено HTML-сторінку з 3 слайдами, стилізовано компоненти за допомогою CSS, а логіку перемикання реалізовано через JavaScript.

HTML (index.html):

<!DOCTYPE html>

<html lang="uk">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Слайдер</title>

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

</head>

<body>

<div class="slider">

<div class="slides">

<div class="slide active">Слайд 1</div>

<div class="slide">Слайд 2</div>

<div class="slide">Слайд 3</div>

</div>

<button id="prev">← Назад</button>

<button id="next">Вперед →</button>

</div>

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

</body>

</html>

CSS (style.css):

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

background-color: #eef;

margin: 0;

}

.slider {

position: relative;

width: 300px;

height: 200px;

overflow: hidden;

background-color: #fff;

border: 2px solid #ccc;

border-radius: 8px;

}

.slides {

display: flex;

transition: transform 0.5s ease-in-out;

}

.slide {

min-width: 100%;

height: 100%;

display: none;

justify-content: center;

align-items: center;

font-size: 24px;

background-color: #dde;

}

.slide.active {

display: flex;

}

button {

position: absolute;

top: 50%;

transform: translateY(-50%);

background-color: #0077cc;

color: white;

border: none;

padding: 8px 12px;

cursor: pointer;

border-radius: 4px;

font-size: 16px;

}

#prev {

left: 10px;

}

#next {

right: 10px;

}

JavaScript (script.js):

const slides = document.querySelectorAll('.slide');

const prevBtn = document.getElementById('prev');

const nextBtn = document.getElementById('next');

let currentIndex = 0;

function showSlide(index) {

slides.forEach((slide, i) => {

slide.classList.toggle('active', i === index);

});

}

prevBtn.addEventListener('click', () => {

currentIndex = (currentIndex - 1 + slides.length) % slides.length;

showSlide(currentIndex);

});

nextBtn.addEventListener('click', () => {

currentIndex = (currentIndex + 1) % slides.length;

showSlide(currentIndex);

});

Результати:

**Изображение выглядит как текст, снимок экрана, Шрифт, логотип

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, Шрифт, Прямоугольник

Контент, сгенерированный ИИ, может содержать ошибки.**

**Изображение выглядит как текст, снимок экрана, Шрифт, Прямоугольник

Контент, сгенерированный ИИ, может содержать ошибки.**

**2.** Створити скрипт, що реалізує зворотній лічильник до події та функцію підписки.

Рішення:

Було реалізовано вебсторінку, яка показує зворотний лічильник до події (дні, години, хвилини, секунди) та кнопку «Підписатися». Після натискання на кнопку виводиться повідомлення про успішну підписку з інформацією, скільки часу залишалось. Якщо користувач не натисне вчасно — відображається повідомлення про завершення реєстрації та кнопка видаляється.

HTML (index.html):

<!DOCTYPE html>

<html lang="uk">

<head>

<meta charset="UTF-8">

<title>Зворотний лічильник</title>

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

</head>

<body>

<div class="container">

<h1>До початку події залишилося:</h1>

<div id="countdown">

<span id="days">0</span> днів

<span id="hours">0</span> год

<span id="minutes">0</span> хв

<span id="seconds">0</span> сек

</div>

<button id="subscribeBtn">Підписатися</button>

<p id="message"></p>

</div>

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

</body>

</html>

#🎨 CSS (style.css):

body {

font-family: Arial, sans-serif;

background-color: #f2f2f2;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

}

.container {

text-align: center;

background-color: #fff;

padding: 30px;

border-radius: 12px;

box-shadow: 0 0 10px rgba(0,0,0,0.2);

}

#countdown {

font-size: 24px;

margin: 20px 0;

}

#subscribeBtn {

font-size: 18px;

padding: 10px 20px;

background-color: #007acc;

color: white;

border: none;

border-radius: 6px;

cursor: pointer;

}

#subscribeBtn:disabled {

background-color: gray;

cursor: not-allowed;

}

#message {

margin-top: 20px;

font-size: 18px;

color: green;

}

#⚙️ JavaScript (script.js):

const countdown = document.getElementById("countdown");

const daysEl = document.getElementById("days");

const hoursEl = document.getElementById("hours");

const minutesEl = document.getElementById("minutes");

const secondsEl = document.getElementById("seconds");

const subscribeBtn = document.getElementById("subscribeBtn");

const messageEl = document.getElementById("message");

// Встановити дату події (через 60 секунд від поточного часу)

const eventDate = new Date(new Date().getTime() + 60 1000);

let interval = setInterval(updateCountdown, 1000);

function updateCountdown() {

const now = new Date();

const diff = eventDate - now;

if (diff <= 0) {

clearInterval(interval);

daysEl.textContent = "0";

hoursEl.textContent = "0";

minutesEl.textContent = "0";

secondsEl.textContent = "0";

subscribeBtn.remove();

messageEl.style.color = "red";

messageEl.textContent = "Реєстрацію завершено. Ви не встигли.";

return;

}

const seconds = Math.floor((diff / 1000) % 60);

const minutes = Math.floor((diff / 1000 / 60) % 60);

const hours = Math.floor((diff / 1000 / 60 / 60) % 24);

const days = Math.floor(diff / 1000 / 60 / 60 / 24);

daysEl.textContent = days;

hoursEl.textContent = hours;

minutesEl.textContent = minutes;

secondsEl.textContent = seconds;

}

subscribeBtn.addEventListener("click", () => {

const now = new Date();

const diff = Math.max(0, Math.floor((eventDate - now) / 1000)); // залишок у секундах

subscribeBtn.disabled = true;

messageEl.textContent = `Дякуємо за підписку! Ви встигли! До завершення підписки залишалося ${diff}с.`;

});

Результати:

Таймер автоматично запускається при завантаженні сторінки.

Якщо користувач натисне на «Підписатися» до завершення таймера — з'являється повідомлення з залишком часу.

Якщо користувач не натисне — лічильник обнуляється, кнопка зникає, і виводиться повідомлення про завершення підписки.

3. Створити скрипт, що реалізує валідацію форми на сторінці Contact Us.

*Рішення:*

Було створено репозиторій для вебсайту, в який завантажено розпакований архів `www\_lb4.zip`. У проєкті додано каталог `js/`, у якому створено файл `validation.js` для реалізації валідації форми засобами JavaScript.

Форма на сторінці Contact Us перевіряє коректність введених даних згідно таких умов:

Your Name: від 2 до 30 символів, лише букви (українські та латинські), пробіли та дефіс.

Age: числове значення від 18 до 130.

E-Mail: відповідає стандартному формату email-адрес.

Your Website: або порожнє, або валідна URL-адреса (http/https).

Your Message: довжина від 10 до 350 символів.

При помилці:

Поле підсвічується червоною рамкою (власні стилі у CSS).

Під полем з'являється пояснення у червоному кольорі.

При повторному фокусі на полі — підсвітка та повідомлення зникають.

Файл `validation.js`:

document.addEventListener('DOMContentLoaded', function () {

const form = document.querySelector('form');

const nameInput = form.querySelector('input[name="name"]');

const ageInput = form.querySelector('input[name="age"]');

const emailInput = form.querySelector('input[name="email"]');

const websiteInput = form.querySelector('input[name="website"]');

const messageInput = form.querySelector('textarea[name="message"]');

form.addEventListener('submit', function (e) {

e.preventDefault(); // зупинити відправку

let valid = true;

clearErrors();

// Name

const nameValue = nameInput.value.trim();

if (!/^[A-Za-zА-Яа-яЇїІіЄєҐґ'’ -]{2,30}$/.test(nameValue)) {

showError(nameInput, '\*Ім’я має містити 2-30 символів: літери, пробіли, дефіс.');

valid = false;

}

// Age

const ageValue = parseInt(ageInput.value, 10);

if (isNaN(ageValue) || ageValue < 18 || ageValue > 130) {

showError(ageInput, '\*Вік повинен бути в межах 18-130.');

valid = false;

}

// Email

const emailValue = emailInput.value.trim();

if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(emailValue)) {

showError(emailInput, '\*Некоректна електронна адреса.');

valid = false;

}

// Website

const websiteValue = websiteInput.value.trim();

if (websiteValue !== "" && !/^https?:\/\/[^\s]+$/.test(websiteValue)) {

showError(websiteInput, '\*URL має бути валідним або залишеним пустим.');

valid = false;

}

// Message

const messageValue = messageInput.value.trim();

if (messageValue.length < 10 || messageValue.length > 350) {

showError(messageInput, '\*Повідомлення має бути 10–350 символів.');

valid = false;

}

if (valid) {

form.submit(); // якщо всі поля валідні

}

});

// Скидання помилок при фокусі

[nameInput, ageInput, emailInput, websiteInput, messageInput].forEach(input => {

input.addEventListener('focus', () => {

removeError(input);

});

});

function showError(input, message) {

input.classList.add('input-error');

const error = document.createElement('div');

error.className = 'error-message';

error.textContent = message;

input.insertAdjacentElement('afterend', error);

}

function removeError(input) {

input.classList.remove('input-error');

const next = input.nextElementSibling;

if (next && next.classList.contains('error-message')) {

next.remove();

}

}

function clearErrors() {

document.querySelectorAll('.input-error').forEach(el => el.classList.remove('input-error'));

document.querySelectorAll('.error-message').forEach(el => el.remove());

}

});

CSS-додаток до `style.css` (власні стилі):

.input-error {

border: 2px solid red;

}

.error-message {

color: red;

font-size: 14px;

margin-top: 4px;

}

Приклади результатів:

* При неправильному email або віці:
  + поле підсвічується червоним;
  + з'являється повідомлення з вказівкою помилки.
* Після фокусування на полі — очищується стиль і повідомлення.

4. Розробити систему керування задачами.

Рішення:

Було реалізовано систему керування задачами на HTML, CSS та JavaScript. Інтерфейс дозволяє додавати задачі, позначати їх як виконані або видаляти, а також фільтрувати список за статусами: всі, в роботі, виконано, видалено.

Функціонал реалізовано наступним чином:

При введенні тексту у поле та натисканні кнопки «Додати задачу» — створюється нова задача, яка з’являється першою в списку.

Для задач зі статусом «в роботі» відображаються кнопки: ✅ (виконано) та 🗑️ (видалити).

При натисканні на ✅ — задача змінює статус на «виконано», додається час виконання, а піктограми зникають.

При натисканні на 🗑️ — задача змінює статус на «видалено», текст перекреслюється, додається час видалення, а піктограми зникають.

Список задач можна фільтрувати за допомогою перемикачів (radio): всі / в роботі / виконано / видалено.

Задачі відображаються у відповідному порядку (останні додані — вгорі).

HTML (index.html)

<!DOCTYPE html>

<html lang="uk">

<head>

<meta charset="UTF-8">

<title>Менеджер задач</title>

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

</head>

<body>

<div class="container">

<h1>Система керування задачами</h1>

<div class="task-input">

<input type="text" id="taskInput" placeholder="Введіть опис задачі...">

<button id="addTaskBtn">Додати задачу</button>

</div>

<div class="filters">

<label><input type="radio" name="filter" value="all" checked> Всі</label>

<label><input type="radio" name="filter" value="active"> В роботі</label>

<label><input type="radio" name="filter" value="done"> Виконано</label>

<label><input type="radio" name="filter" value="deleted"> Видалено</label>

</div>

<ul id="taskList"></ul>

</div>

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

</body>

</html>

CSS (style.css)

body {

font-family: Arial, sans-serif;

background-color: #f0f2f5;

display: flex;

justify-content: center;

padding: 30px;

}

.container {

width: 500px;

background-color: white;

padding: 20px;

border-radius: 10px;

box-shadow: 0 0 15px rgba(0,0,0,0.1);

}

.task-input {

display: flex;

gap: 10px;

margin-bottom: 15px;

}

#taskInput {

flex-grow: 1;

padding: 10px;

font-size: 16px;

}

#addTaskBtn {

padding: 10px;

font-size: 16px;

cursor: pointer;

}

.filters {

margin-bottom: 15px;

}

.filters label {

margin-right: 10px;

}

#taskList {

list-style: none;

padding: 0;

}

.task {

padding: 10px;

border: 1px solid #ddd;

margin-bottom: 8px;

border-radius: 5px;

background-color: #fafafa;

display: flex;

justify-content: space-between;

align-items: center;

}

.task-text {

flex-grow: 1;

}

.task .actions {

margin-left: 10px;

display: flex;

gap: 8px;

}

.task.done .task-text {

color: green;

}

.task.deleted {

text-decoration: line-through;

color: gray;

}

JavaScript (script.js)

document.addEventListener('DOMContentLoaded', () => {

const taskInput = document.getElementById('taskInput');

const addTaskBtn = document.getElementById('addTaskBtn');

const taskList = document.getElementById('taskList');

const filterRadios = document.querySelectorAll('input[name="filter"]');

let tasks = [];

function renderTasks() {

taskList.innerHTML = '';

const selectedFilter = document.querySelector('input[name="filter"]:checked').value;

let filtered = [];

if (selectedFilter === 'all') {

filtered = tasks;

} else {

filtered = tasks.filter(t => t.status === selectedFilter);

}

filtered.forEach(task => {

const li = document.createElement('li');

li.className = `task ${task.status}`;

const taskText = document.createElement('span');

taskText.className = 'task-text';

taskText.textContent = task.text;

if (task.status === 'done') {

taskText.textContent += ` (Виконано о ${task.time})`;

}

if (task.status === 'deleted') {

taskText.textContent += ` (Видалено о ${task.time})`;

}

li.appendChild(taskText);

// Додати дії тільки якщо задача "active"

if (task.status === 'active') {

const actions = document.createElement('div');

actions.className = 'actions';

const doneBtn = document.createElement('button');

doneBtn.textContent = '✅';

doneBtn.title = 'Позначити як виконано';

doneBtn.onclick = () => completeTask(task.id);

const deleteBtn = document.createElement('button');

deleteBtn.textContent = '🗑️';

deleteBtn.title = 'Видалити задачу';

deleteBtn.onclick = () => deleteTask(task.id);

actions.appendChild(doneBtn);

actions.appendChild(deleteBtn);

li.appendChild(actions);

}

taskList.appendChild(li);

});

}

function addTask() {

const text = taskInput.value.trim();

if (!text) return;

const task = {

id: Date.now(),

text: text,

status: 'active',

time: null

};

tasks.unshift(task);

taskInput.value = '';

renderTasks();

}

function completeTask(id) {

const task = tasks.find(t => t.id === id);

if (task) {

task.status = 'done';

task.time = new Date().toLocaleTimeString();

tasks = [task, ...tasks.filter(t => t.id !== id)];

renderTasks();

}

}

function deleteTask(id) {

const task = tasks.find(t => t.id === id);

if (task) {

task.status = 'deleted';

task.time = new Date().toLocaleTimeString();

tasks = [task, ...tasks.filter(t => t.id !== id)];

renderTasks();

}

}

addTaskBtn.addEventListener('click', addTask);

taskInput.addEventListener('keypress', (e) => {

if (e.key === 'Enter') addTask();

});

filterRadios.forEach(radio => {

radio.addEventListener('change', renderTasks);

});

});

Скріншот роботи інтерфейсу:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

**Відповідь на контрольні питання:**

1. Що таке подія в JavaScript?

Подія — це дія, яка відбувається у браузері або на веб-сторінці, і яку можна обробити за допомогою JavaScript. Наприклад: клік миші, натискання клавіші, зміна розміру вікна, завантаження сторінки тощо. Події дозволяють зробити веб-сторінки інтерактивними.

2. Наведіть приклад всіх відомих Вам способів підключення обробника події.

У JavaScript існує кілька способів додавання обробників подій:

а) Через HTML-атрибут:

<button onclick="alert('Натиснуто!')">Натисни мене</button>

б) Через властивість DOM-елемента:

const btn = document.getElementById('myBtn');

btn.onclick = function () {

alert('Натиснуто!');

};

в) За допомогою `addEventListener`:

const btn = document.getElementById('myBtn');

btn.addEventListener('click', () => {

alert('Натиснуто!');

});

3. Чи можна на одну подію підключити більш ніж одного обробника?

Так, але лише якщо використовувати метод `addEventListener`. При використанні властивості `onclick` можна встановити лише один обробник — попередній буде перезаписано.

const btn = document.getElementById('myBtn');

btn.addEventListener('click', () => {

console.log('Перший обробник');

});

btn.addEventListener('click', () => {

console.log('Другий обробник');

});

4. Як видалити обробника події?

Щоб видалити обробник, потрібно передати ідентичну функцію у `removeEventListener`:

function handleClick() {

alert('Клік!');

}

const btn = document.getElementById('myBtn');

btn.addEventListener('click', handleClick);

// Пізніше:

btn.removeEventListener('click', handleClick);

> Анонімні функції не можна видалити, бо до них немає посилання.

5. При підключенні обробника події до групи елементів, як визначити на якому елементі відбулася подія?

Для цього використовується властивість `event.target`, яка вказує на елемент, що ініціював подію.

document.querySelectorAll('.item').forEach(item => {

item.addEventListener('click', function (event) {

console.log('Натиснуто на:', event.target.textContent);

});

});

**Висновок**

У результаті виконання лабораторної роботи №4 було набуте практичне вміння обробляти події в JavaScript для інтерактивної роботи з елементами веб-сторінки. Було реалізовано слайдер з циклічним перемиканням, зворотний лічильник з кнопкою підписки, валідацію форми з повідомленнями про помилки та систему керування задачами з фільтрацією. Усі завдання демонструють вміння застосовувати події, маніпулювати DOM і покращувати зручність взаємодії користувача з вебінтерфейсом.