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

Вайцеховський Олександр

Перевірив ст. викл. каф. КН

Северин М.В.

Одеса - 2025

**Лабораторної роботи №4**

**Тема.** Події в JavaScript.

**Мета:** набуття практичного досвіду роботи з подіями та елементами веб-

сторінки в JavaScript.

**Завдання**

1. Створіть слайдер сайту. Забезпечте прокрутку слайдів ліворуч та праворуч, при доходженні до останнього слайду виконувати показ з початку (гортання слайдів по колу).

const slidesList = [

    "./img/1.webp",

    "./img/2.webp",

    "./img/3.jpg",

    "./img/4.webp",

];

    for(item in slidesList)

    {

        document.getElementsByClassName("slide")[0].innerHTML += `<img class="img\_slide" style="opacity: 0; width:40vw; position: absolute; margin-left:30vw;" src="${slidesList[item]}"/>`;

    }

let slideNumber = 0;

document.getElementsByClassName("img\_slide")[0].style.opacity = "1";

function slidePrev() {

    if(slideNumber==0)

    {

        document.getElementsByClassName("img\_slide")[0].style.opacity = "0";

        slideNumber = slidesList.length - 1;

        document.getElementsByClassName("img\_slide")[slideNumber].style.opacity = "1";

    }

    else

    {

        document.getElementsByClassName("img\_slide")[slideNumber].style.opacity = "0";

        slideNumber = slideNumber - 1;

        document.getElementsByClassName("img\_slide")[slideNumber].style.opacity = "1";

    }

}

function slideNext() {

    if(slideNumber==0)

    {

        document.getElementsByClassName("img\_slide")[slideNumber].style.opacity = "0";

        slideNumber = (slideNumber + 1) % slidesList.length;

        document.getElementsByClassName("img\_slide")[slideNumber].style.opacity = "1";

    }

    else

    {

        document.getElementsByClassName("img\_slide")[slideNumber].style.opacity = "0";

        slideNumber = (slideNumber + 1) % slidesList.length;

        document.getElementsByClassName("img\_slide")[slideNumber].style.opacity = "1";

    }

}

Шмат коду 1.1 — Скрипт слайдеру

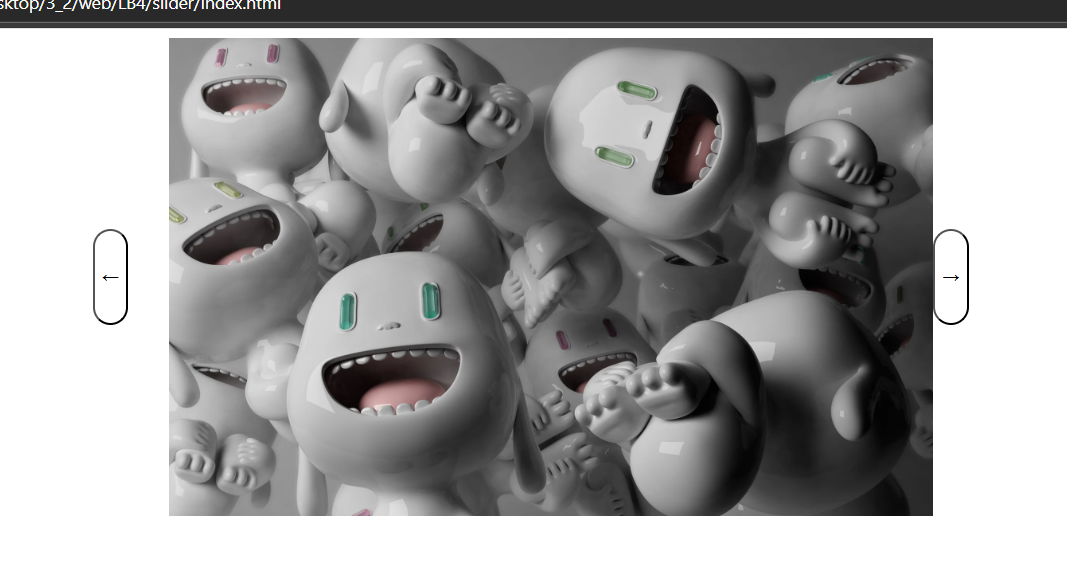


Рисунок 1.1 — Слайдер картинок у браузері

2. Створіть веб сторінку зі зворотнім лічильником до певної події та кнопкою Підписатися. Відлік показує кількість днів, годин, хвилин, секунд. При

натисканні на кнопку Підписатися, кнопка стає недоступною для

натискання, а на веб-сторінці виводиться повідомлення з інформацією:

«Дякуємо за підписку! Ви встигли! До завершення підписки залишалося

XXXc.» Де XXX – кількість секунд до настання події. Якщо час вийде, а

користувач не натисне кнопку Підписатися, то зупинити лічильник (всі

значення встановлено в 0), вивести повідомлення про завершення

реєстрації на подію, видалити кнопку Підписатися.

const button = document.getElementsByClassName("subscribe")[0];

const timeFrame = [

    document.getElementsByClassName("day")[0],

    document.getElementsByClassName("hour")[0],

    document.getElementsByClassName("minute")[0],

    document.getElementsByClassName("second")[0],

];

const time = Math.random() \* 5 + 1;

const timeOfExpire = new Date(Date.now() + time\*60000);

let subscription = false;

function timer() {

    if(timeRemaining > 0) {

    let timeRemaining = new Date(timeOfExpire - Date.now());

        if(Math.floor((timeRemaining/36000000)/24)) {

        timeFrame[0].innerText = Math.floor((timeRemaining/3600000)/24) + "d";

        }

        else {

            timeFrame[0].innerText = 0 + "d";

        }

        timeFrame[1].innerText = Math.floor(timeRemaining/3600000) + "h";

        timeFrame[2].innerText = Math.floor(timeRemaining/60000) + "m";

        timeFrame[3].innerText = Math.floor(timeRemaining/1000)%60 + "s";

        timeRemaining -= 1000;

        setTimeout("timer()", 1000);

    }

    timeout();

}

function timeout() {

    if(subscription) {

        alert("Nice job, but you got no subscription now...");

        button.remove();

    }

    else {

        alert("No subscription - no candy");

        button.remove();

    }

}

function subscribe() {

    alert("Good job! You have subscribed! Now your subscription lasts for a " + (timeOfExpire-Date.now()) \* 1000 + "seconds");

    subscription = true;

    button.disabled = true;

}

Шмат коду 2.1 — скрипт підписки

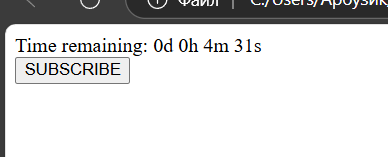


Рисунок 2.1 — Таймер до події

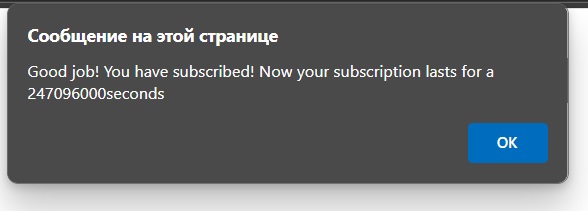


Рисунок 2.2 — повідомлення про підписку

3. Створіть репозиторій для розробки веб-сайту. Завантажте та розпакуйте в

репозиторій архів www\_lb4.zip, що містить веб-сайт (html, css, images).

Створіть каталог js, додайте в нього файл(и) js, виконайте наступні

завдання засобами JavaScript:

На сторінці Contact Us міститься форма, створити скрипт валідації

даних форми перед її відправкою на сервер. В роботі забезпечити перехват

відправки форми при натискання кнопки Send Message та виконання

валідації з JavaScript за наступними умовами:

− Your Name – довжина мінімум два символи, максимум 30, містить лише символи алфавіту, дефіс та пробіл;

− Age – число в діапазоні 18-130;

− E-Mail – відповідає правилам запису електронної почтової скриньки;

− Your Website – пусто, або валідність url сайту;

− Your Message – 10-350 символів.

При виявленні невалідних даних:

− використовуючи css стилі встановити через клас червоний колір для

рамки (border) невалідного поля (стилі створити власноруч в style.css).

− під полем виводити напис червоним кольором: \*інформація з

вказівкою вимог до поля;

− при отриманні фокуса на невалідному полі повернути колір рамки

(border) поля в початковий стан та прибрати інформацію під полем.

const form = document.getElementById('contactform');

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

  event.preventDefault();

  let isValid = true;

  const nameInput = document.getElementById('name');

  if (!validateName(nameInput.value)) {

    showError(nameInput, "\* Ім’я: 2-30 літер, тільки букви, дефіс або пробіл.");

    isValid = false;

  } else {

    clearError(nameInput);

  }

  const emailInput = document.getElementById('email');

  if (!validateEmail(emailInput.value)) {

    showError(emailInput, "\* Некоректний email.");

    isValid = false;

  } else {

    clearError(emailInput);

  }

  const ageInput = document.getElementById('age');

  if (!validateAge(ageInput.value)) {

    showError(ageInput, "\* Вік має бути числом від 18 до 130.");

    isValid = false;

  } else {

    clearError(ageInput);

  }

  const websiteInput = document.getElementById('company');

  if (!validateWebsite(websiteInput.value)) {

    showError(websiteInput, "\* Некоректний URL або поле залиш пустим.");

    isValid = false;

  } else {

    clearError(websiteInput);

  }

  const messageInput = document.getElementById('message');

  if (!validateMessage(messageInput.value)) {

    showError(messageInput, "\* Повідомлення має бути від 10 до 350 символів.");

    isValid = false;

  } else {

    clearError(messageInput);

  }

  if (isValid) {

    //form.submit();

  }

});

const inputs = document.querySelectorAll('.user-data');

inputs.forEach(input => {

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

    clearError(input);

  });

});

function showError(input, message) {

  input.classList.add('input-error');

  let errorElem = input.nextElementSibling;

  if (!errorElem || !errorElem.classList.contains('error-message')) {

    errorElem = document.createElement('div');

    errorElem.classList.add('error-message');

    input.parentNode.insertBefore(errorElem, input.nextSibling);

  }

  errorElem.textContent = message;

}

function clearError(input) {

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

  let errorElem = input.nextElementSibling;

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

    errorElem.textContent = '';

  }

}

function validateName(name) {

  let pattern = /^[a-zA-Zа-яА-ЯёЁіІїЇєЄ\- ]{2,30}$/;

  return pattern.test(name);

}

function validateAge(age) {

  let num = Number(age);

  return Number.isInteger(num) && num >= 18 && num <= 130;

}

function validateEmail(email) {

  let pattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

  return pattern.test(email);

}

function validateWebsite(website) {

  if (website.trim() === "") return true;

  try {

    new URL(website);

    return true;

  } catch (e) {

    return false;

  }

}

function validateMessage(message) {

  let length = message.trim().length;

  return length >= 10 && length <= 350;

}

Шмат коду 3.1 — Серипт валідації данних

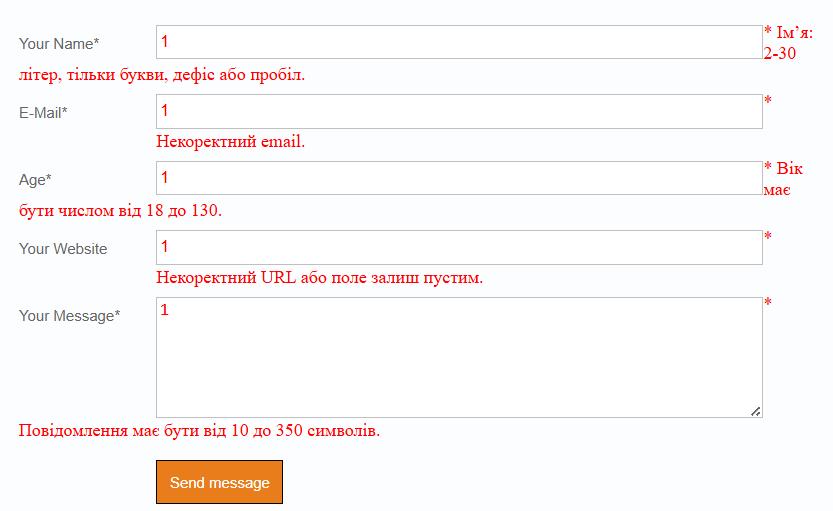


Рисунок 3.1 — Перехоплення форми

4. Розробити систему керування задачами.

Елементи системи керування задачами: поле вводу для опису задачі,

кнопка додати задачу, елемент накладання фільтру за статусом задачі

(випадающий список, або перемикачі (radio), значення за умовчанням

відображає всі задачі).

Функціонал: створення задачі, видалення задачі, переведення задачі в

статус «виконано», фільтрація задач за статусом всі/в

роботі/виконано/видалено.

Опис функціоналу: при введенні даних в поле вводу та натискання кнопки

створити задачу, виконати створення нового елементу та добавлення його

на початку списку задач, який розташовано під полем вводу. Видалення задачі, або переведення задачі в статус «виконано», відбувається шляхом натискання відповідної піктограми розташованої на обраній задачі, після виконання відповідної дії піктограми видалення/виконано становляться недоступними (відсутні на задачі). При видаленні задачі текст задачі перекреслений, до опису задачі додається час видалення, задача переноситься до списку видалених задач і розташовується першою в списку видалених задач. При переведенні задачі в статус «виконано» до її опису додається відмітка «Виконано» та час виконання, задача переводить до списку виконаних задач і розташовується першою в даному списку. При роботі з фільтром забезпечити відображення списку задач у відповідності до обраного статусу.

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

    const taskInput = document.getElementById('task-input');

    const addTaskBtn = document.getElementById('add-task-btn');

    const filterStatus = document.getElementById('filter-status');

    const taskList = document.getElementById('task-list');

    let tasks = [];

    function renderTasks() {

        taskList.innerHTML = '';

        const filter = filterStatus.value;

        let filteredTasks = [];

        if (filter === 'in-progress') {

            filteredTasks = tasks.filter(task => !task.completed && !task.deleted);

        } else if (filter === 'completed') {

            filteredTasks = tasks.filter(task => task.completed);

        } else if (filter === 'deleted') {

            filteredTasks = tasks.filter(task => task.deleted);

        } else {

            filteredTasks = tasks;

        }

        filteredTasks.forEach((task, index) => {

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

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

            taskText.textContent = task.text;

            if (task.completed) {

                li.classList.add('completed');

                taskText.textContent += ` (Виконано: ${task.timeCompleted})`;

            }

            if (task.deleted) {

                li.classList.add('deleted');

                taskText.textContent += ` (Видалено: ${task.timeDeleted})`;

            }

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

            if (!task.completed && !task.deleted) {

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

                completeBtn.textContent = '✅ Виконано';

                completeBtn.addEventListener('click', () => markCompleted(index));

                actions.appendChild(completeBtn);

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

                deleteBtn.textContent = '❌ Видалити';

                deleteBtn.addEventListener('click', () => deleteTask(index));

                actions.appendChild(deleteBtn);

            }

            li.appendChild(taskText);

            li.appendChild(actions);

            taskList.prepend(li); // Новіші зверху

        });

    }

    function addTask() {

        const taskText = taskInput.value.trim();

        if (taskText) {

            const newTask = {

                text: taskText,

                completed: false,

                deleted: false,

                timeCreated: new Date().toLocaleString(),

            };

            tasks.unshift(newTask);

            taskInput.value = '';

            renderTasks();

        }

    }

    function markCompleted(index) {

        const task = tasks[index];

        task.completed = true;

        task.timeCompleted = new Date().toLocaleString();

        renderTasks();

    }

    function deleteTask(index) {

        const task = tasks[index];

        task.deleted = true;

        task.timeDeleted = new Date().toLocaleString();

        renderTasks();

    }

    addTaskBtn.addEventListener('click', addTask);

    filterStatus.addEventListener('change', renderTasks);

    renderTasks();

});

Шмат коду 4.1 — Скрипт Системи керування задачами

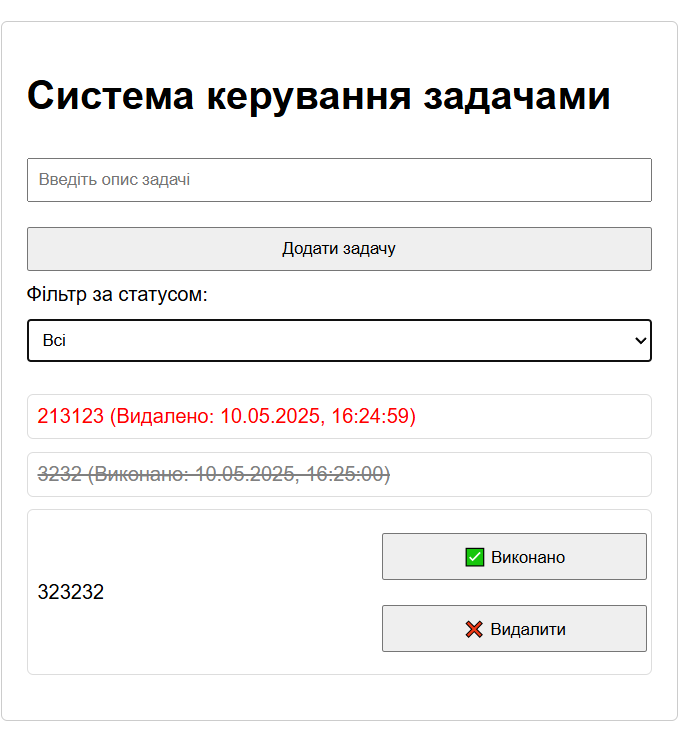


Рисунок 4.1 — Система керування задачами

**Контрольні питання**

1. Що таке подія в JavaScript?

Подія в JavaScript — це сигнал про те, що щось сталося, наприклад, клік миші або натискання клавіші.

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

Обробник можна підключити через HTML-атрибут (onclick), через властивість DOM-елемента (element.onclick = function...) або методом addEventListener.

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

Так, метод addEventListener дозволяє додати кілька обробників на одну подію.

4.Як видалити обробника події?

Щоб видалити обробник, використовують метод removeEventListener з тим самим типом події та функцією.

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

Усередині обробника можна використати event.target, щоб дізнатися елемент, на якому сталася подія.

**Висновок**

Протягом лабороторної роботи були отримані теретичні основи виконання коду сторінок та їх скриптів, особливо взаємодії із в спливаючих вікнах та формах, взаємодії із змінними та їх оголошення і т.д. Після чого було практично затверджено правильність отриманої теорії шляхом створення функціонального веб-додатку.