**ЛАБОРАТОРНА РОБОТА № 9**

**Тема:** ООП в JavaScript

**Завдання 1.** Створіть клас "Модальне вікно" (Modal), який створює модальне вікно на сторінці. Клас повинен мати методи для відкриття, закриття та встановлення вмісту модального вікна. Зробіть можливість переміщення вікна по екрану за допомогою мишки (щоб можна було перетягувати за заголовок вікна).

Розв’язок

Html-код:

<body>

    <button

      onclick="modal.**open**('Привіт! Це модальне вікно, яке можна переміщувати.')"

    >

      Відкрити модальне вікно

    </button>

    <script src="script.js"></script>

  </body>

Css-код:

body {

  font-family: Arial, sans-serif;

}

.modal {

  display: none;

  position: fixed;

  width: 300px;

  border: 2px solid #333;

  background-color: #fff;

  border-radius: 8px;

  z-index: 1000;

  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

}

.modal-header {

  padding: 10px;

  background-color: #333;

  color: #fff;

  cursor: move;

  border-top-left-radius: 8px;

  border-top-right-radius: 8px;

}

.modal-body {

  padding: 20px;

}

.modal-close {

  position: absolute;

  top: 10px;

  right: 10px;

  cursor: pointer;

  font-weight: bold;

  color: #fff;

}

Js-код:

class Modal {

  constructor() {

    this.modal = document.**createElement**("div");

    this.modal.className = "modal";

    this.header = document.**createElement**("div");

    this.header.className = "modal-header";

    this.header.textContent = "Модальне вікно";

    this.closeButton = document.**createElement**("span");

    this.closeButton.className = "modal-close";

    this.closeButton.textContent = "×";

    this.closeButton.**onclick** = () => this.**close**();

    this.body = document.**createElement**("div");

    this.body.className = "modal-body";

    this.header.**appendChild**(this.closeButton);

    this.modal.**appendChild**(this.header);

    this.modal.**appendChild**(this.body);

    document.body.**appendChild**(this.modal);

    this.**initDrag**();

  }

**open**(content) {

    this.body.innerHTML = content;

    this.modal.style.display = "block";

    this.modal.style.left = "50%";

    this.modal.style.top = "50%";

    this.modal.style.transform = "translate(-50%, -50%)";

  }

**close**() {

    this.modal.style.display = "none";

  }

**setContent**(content) {

    this.body.innerHTML = content;

  }

**initDrag**() {

    let isDragging = false;

    let offsetX, offsetY;

    this.header.**addEventListener**("mousedown", (e) => {

      isDragging = true;

      offsetX = e.clientX - this.modal.offsetLeft;

      offsetY = e.clientY - this.modal.offsetTop;

      document.**addEventListener**("mousemove", this.handleDrag);

      document.**addEventListener**("mouseup", this.stopDrag);

    });

    this.**handleDrag** = (e) => {

      if (isDragging) {

        this.modal.style.left = e.clientX - offsetX + "px";

        this.modal.style.top = e.clientY - offsetY + "px";

      }

    };

    this.**stopDrag** = () => {

      isDragging = false;

      document.**removeEventListener**("mousemove", this.handleDrag);

      document.**removeEventListener**("mouseup", this.stopDrag);

    };

  }

}

const modal = new **Modal**();



Рис. 1. Результат завдання 1

**Завдання 2.** Створіть клас "Слайдер" (Slider), який дозволяє створювати горизонтальний або вертикальний слайдер з можливістю перегляду зображень або вмісту. Клас повинен мати методи для додавання слайдів та переміщення між ними.

Розв’язок

Html-код:

<body>

    <div class="slider-container" id="sliderContainer"></div>

    <script src="script.js"></script>

  </body>

Css-код:

.slider-container {

  position: relative;

  width: 600px;

  aspect-ratio: 16/9;

  overflow: hidden;

  border: 2px solid #333;

  border-radius: 8px;

}

.slider-track {

  display: flex;

  transition: transform 0.5s ease;

}

.slider-slide,

.slider-slide img {

  min-width: 100%;

  height: 100%;

  display: flex;

  align-items: center;

  justify-content: center;

  font-size: 24px;

  color: #fff;

  background-color: #555;

}

.slider-controls {

  position: absolute;

  top: 50%;

  width: 100%;

  display: flex;

  justify-content: space-between;

  transform: translateY(-50%);

}

.slider-button {

  background-color: rgba(0, 0, 0, 0.7);

  border: none;

  color: white;

  width: 50px;

  height: 50px;

  cursor: pointer;

  font-size: 18px;

  border-radius: 50%;

}

JS-код:

class Slider {

  constructor(container, direction = "horizontal") {

    this.container = container;

    this.direction = direction;

    this.slides = [];

    this.currentSlide = 0;

    this.track = document.**createElement**("div");

    this.track.className = "slider-track";

    this.container.**appendChild**(this.track);

    this.**createControls**();

  }

**createControls**() {

    const controls = document.**createElement**("div");

    controls.className = "slider-controls";

    const prevButton = document.**createElement**("button");

    prevButton.className = "slider-button";

    prevButton.innerHTML = "&#10094;";

    prevButton.**onclick** = () => this.**prevSlide**();

    const nextButton = document.**createElement**("button");

    nextButton.className = "slider-button";

    nextButton.innerHTML = "&#10095;";

    nextButton.**onclick** = () => this.**nextSlide**();

    controls.**appendChild**(prevButton);

    controls.**appendChild**(nextButton);

    this.container.**appendChild**(controls);

  }

**addSlide**(content) {

    const slide = document.**createElement**("div");

    slide.className = "slider-slide";

    slide.innerHTML = content;

    this.track.**appendChild**(slide);

    this.slides.**push**(slide);

    this.**updateTrackLayout**();

  }

**updateTrackLayout**() {

    if (this.direction === "vertical") {

      this.track.style.flexDirection = "column";

      this.container.classList.**add**("vertical");

    } else {

      this.track.style.flexDirection = "row";

      this.container.classList.**remove**("vertical");

    }

  }

**nextSlide**() {

    if (this.currentSlide < this.slides.length - 1) {

      this.currentSlide++;

    } else {

      this.currentSlide = 0;

    }

    this.**updateSlidePosition**();

  }

**prevSlide**() {

    if (this.currentSlide > 0) {

      this.currentSlide--;

    } else {

      this.currentSlide = this.slides.length - 1;

    }

    this.**updateSlidePosition**();

  }

**updateSlidePosition**() {

    if (this.direction === "vertical") {

      this.track.style.transform = `translateY(-${this.currentSlide \* 100}%)`;

    } else {

      this.track.style.transform = `translateX(-${this.currentSlide \* 100}%)`;

    }

  }

}

const sliderContainer = document.**getElementById**("sliderContainer");

const slider = new **Slider**(sliderContainer, "horizontal");

slider.**addSlide**('<img src="img/1.jpg" alt="Slide 1">');

slider.**addSlide**('<img src="img/2.jpg" alt="Slide 2">');

slider.**addSlide**('<img src="img/3.jpg" alt="Slide 3">');



Рис. 2. Результат завдання 2

**Завдання 3.** Створіть клас "Таблиця" (Table), який генерує HTML-таблицю на основі заданих даних. Клас повинен мати методи для додавання рядків та стовпців до таблиці.

Розв’язок

Html-код:

<body>

    <div id="tableContainer"></div>

    <button class="table-button" onclick="**addRow**()">Додати рядок</button>

    <button class="table-button" onclick="**addColumn**()">Додати стовпець</button>

    <script src="script.js"></script>

  </body>

Js-код:

class Table {

  constructor(container, headers = []) {

    this.container = container;

    this.headers = headers;

    this.data = [];

    this.table = document.**createElement**("table");

    this.container.**appendChild**(this.table);

    this.**createHeader**();

  }

**createHeader**() {

    const thead = document.**createElement**("thead");

    const headerRow = document.**createElement**("tr");

    this.headers.**forEach**((header) => {

      const th = document.**createElement**("th");

      th.innerText = header;

      headerRow.**appendChild**(th);

    });

    thead.**appendChild**(headerRow);

    this.table.**appendChild**(thead);

  }

**addRow**(rowData = []) {

    const tbody =

      this.table.**querySelector**("tbody") || document.**createElement**("tbody");

    const row = document.**createElement**("tr");

    for (let i = 0; i < this.headers.length; i++) {

      const cell = document.**createElement**("td");

      cell.innerText = rowData[i] || "";

      row.**appendChild**(cell);

    }

    tbody.**appendChild**(row);

    this.table.**appendChild**(tbody);

  }

**addColumn**(headerName) {

    this.headers.**push**(headerName);

    const th = document.**createElement**("th");

    th.innerText = headerName;

    this.table.**querySelector**("thead tr").**appendChild**(th);

    const rows = this.table.**querySelectorAll**("tbody tr");

    rows.**forEach**((row) => {

      const cell = document.**createElement**("td");

      row.**appendChild**(cell);

    });

  }

}

const tableContainer = document.**getElementById**("tableContainer");

const table = new **Table**(tableContainer, ["Ім'я", "Прізвище", "Вік"]);

function **addRow**() {

  table.**addRow**(["Іван", "Іванов", "25"]);

}

function **addColumn**() {

  const newColumnHeader = `Стовпець ${table.headers.length + 1}`;

  table.**addColumn**(newColumnHeader);

}

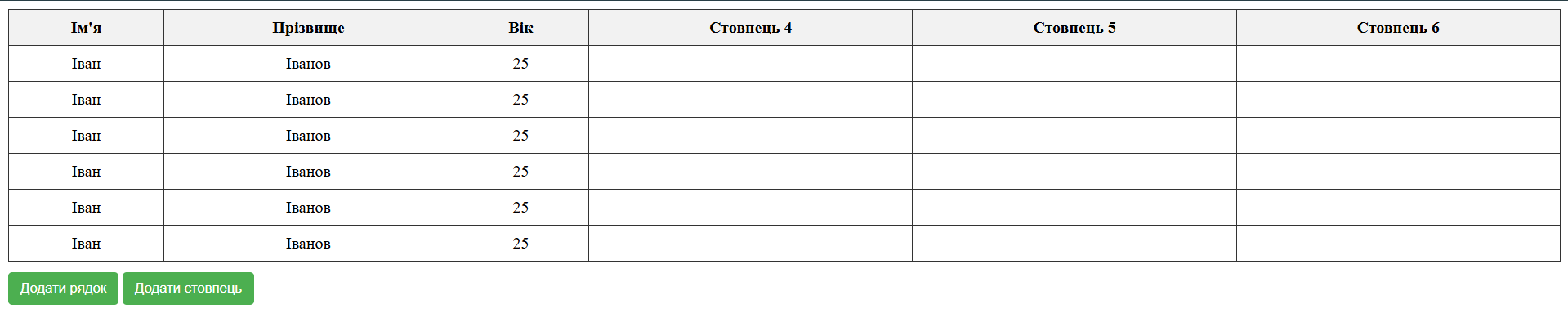


Рис. 3. Результат завдання 3

**Завдання 4.** Створіть клас "Форма" (Form), який дозволяє створювати HTML-форми з заданими полями вводу, кнопками та обробниками подій для них. Клас повинен мати методи для обробки відправки форми та валідації введених даних.

Розв’язок

Html-код:

<body>

    <div id="formContainer"></div>

    <script src="script.js"></script>

  </body>

Js-код:

class Form {

  constructor(container, fields = []) {

    this.container = container;

    this.fields = fields;

    this.form = document.**createElement**("form");

    this.form.className = "form-container";

    this.fields.**forEach**((field) => this.**addField**(field));

    this.**addSubmitButton**();

    this.form.**addEventListener**("submit", (event) => this.**handleSubmit**(event));

    this.container.**appendChild**(this.form);

  }

**addField**(field) {

    const formGroup = document.**createElement**("div");

    formGroup.className = "form-group";

    const label = document.**createElement**("label");

    label.innerText = field.label;

    formGroup.**appendChild**(label);

    const input = document.**createElement**("input");

    input.type = field.type || "text";

    input.name = field.name;

    input.placeholder = field.placeholder || "";

    input.required = field.required || false;

    formGroup.**appendChild**(input);

    const errorMessage = document.**createElement**("span");

    errorMessage.className = "error-message";

    formGroup.**appendChild**(errorMessage);

    this.form.**appendChild**(formGroup);

  }

**addSubmitButton**() {

    const submitButton = document.**createElement**("button");

    submitButton.type = "submit";

    submitButton.className = "form-button";

    submitButton.innerText = "Відправити";

    this.form.**appendChild**(submitButton);

  }

**handleSubmit**(event) {

    event.**preventDefault**();

    let isValid = true;

    const formData = {};

    Array.**from**(this.form.elements).**forEach**((element) => {

      if (element.type !== "submit") {

        const errorMessage =

          element.parentElement.**querySelector**(".error-message");

        if (!element.**checkValidity**()) {

          errorMessage.innerText = `Поле ${element.name} не заповнене коректно`;

          isValid = false;

        } else {

          errorMessage.innerText = "";

          formData[element.name] = element.value;

        }

      }

    });

    if (isValid) {

      this.**processFormData**(formData);

    }

  }

**processFormData**(data) {

    console.**log**("Дані форми:", data);

**alert**("Форма відправлена успішно!");

    this.form.**reset**();

  }

}

const formContainer = document.**getElementById**("formContainer");

const form = new **Form**(formContainer, [

  {

    label: "Ім'я",

    name: "name",

    type: "text",

    placeholder: "Введіть ім'я",

    required: true,

  },

  {

    label: "Електронна пошта",

    name: "email",

    type: "email",

    placeholder: "Введіть email",

    required: true,

  },

  {

    label: "Пароль",

    name: "password",

    type: "password",

    placeholder: "Введіть пароль",

    required: true,

  },

]);

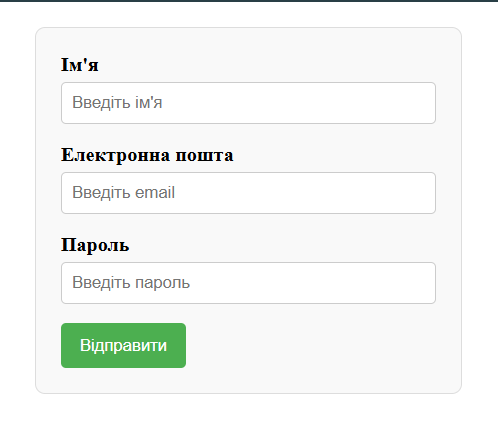


Рис. 4. Результат завдання 4

**Завдання 5.** Створіть клас "Вкладка" (Tab), який створює вкладки для переключення між різними блоками вмісту. Клас повинен мати методи для додавання вкладок та вмісту для них, а також для переключення між вкладками.

Розв’язок

Html-код:

<body>

    <div id="tabContainer" class="tab-container"></div>

    <div class="add-tab-form">

      <h3>Додати нову вкладку</h3>

      <input type="text" id="tabTitle" placeholder="Назва вкладки" />

      <textarea id="tabContent" placeholder="Вміст вкладки"></textarea>

      <button onclick="**addNewTab**()">Додати вкладку</button>

    </div>

    <script src="script.js"></script>

  </body>

Js-код:

class Tab {

  constructor(container) {

    this.container = container;

    this.tabButtons = document.**createElement**("div");

    this.tabButtons.className = "tab-buttons";

    this.container.**appendChild**(this.tabButtons);

    this.tabContents = document.**createElement**("div");

    this.tabContents.className = "tab-contents";

    this.container.**appendChild**(this.tabContents);

    this.tabs = [];

  }

**addTab**(title, content) {

    const tabIndex = this.tabs.length;

    const tabButton = document.**createElement**("button");

    tabButton.className = "tab-button";

    tabButton.innerText = title;

    tabButton.**addEventListener**("click", () => this.**switchTab**(tabIndex));

    this.tabButtons.**appendChild**(tabButton);

    const tabContent = document.**createElement**("div");

    tabContent.className = "tab-content";

    tabContent.innerHTML = content;

    this.tabContents.**appendChild**(tabContent);

    this.tabs.**push**({ button: tabButton, content: tabContent });

    if (this.tabs.length === 1) {

      this.**switchTab**(0);

    }

  }

**switchTab**(index) {

    this.tabs.**forEach**((tab, i) => {

      if (i === index) {

        tab.button.classList.**add**("active");

        tab.content.classList.**add**("active");

      } else {

        tab.button.classList.**remove**("active");

        tab.content.classList.**remove**("active");

      }

    });

  }

}

const tabContainer = document.**getElementById**("tabContainer");

const tabComponent = new **Tab**(tabContainer);

function **addNewTab**() {

  const title = document.**getElementById**("tabTitle").value;

  const content = document.**getElementById**("tabContent").value;

  if (title.**trim**() === "" || content.**trim**() === "") {

**alert**("Будь ласка, заповніть усі поля");

    return;

  }

  tabComponent.**addTab**(title, content);

  document.**getElementById**("tabTitle").value = "";

  document.**getElementById**("tabContent").value = "";

}

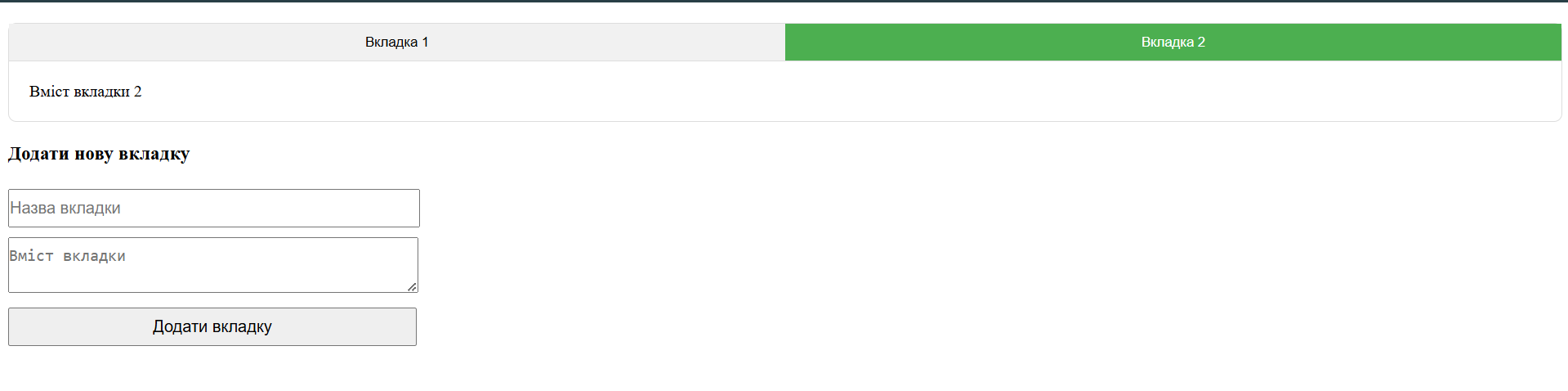


Рис. 5. Результат завдання 5

**Завдання 6.** Створіть клас "Сповіщення" (Notification), який дозволяє створювати спливаючі повідомлення на сторінці. Клас повинен мати методи для створення сповіщень з різними стилями та для їх закриття.

Розв’язок

Html-код:

<body>

    <div id="notificationContainer" class="notification-container"></div>

    <button onclick="notifier.**createNotification**('Успіх!', 'success')">

      Створити успішне сповіщення

    </button>

    <button onclick="notifier.**createNotification**('Помилка!', 'error')">

      Створити сповіщення помилки

    </button>

    <button onclick="notifier.**createNotification**('Попередження!', 'warning')">

      Створити попередження

    </button>

    <button onclick="notifier.**createNotification**('Інформація!', 'info')">

      Створити інформаційне сповіщення

    </button>

    <script src="script.js"></script>

  </body>

Js-код:

class Notification {

  constructor(container) {

    this.container = container;

  }

**createNotification**(

    message,

    type = "info",

    autoClose = true,

    duration = 3000

  ) {

    const notification = document.**createElement**("div");

    notification.className = `notification ${type}`;

    notification.innerHTML = `

            <span>${message}</span>

            <button class="close-btn" onclick="this.parentElement.remove()">×</button>

        `;

    this.container.**appendChild**(notification);

    if (autoClose) {

**setTimeout**(() => {

        notification.**remove**();

      }, duration);

    }

    notification.**addEventListener**("click", () => notification.**remove**());

  }

}

const notificationContainer = document.**getElementById**("notificationContainer");

const notifier = new **Notification**(notificationContainer);

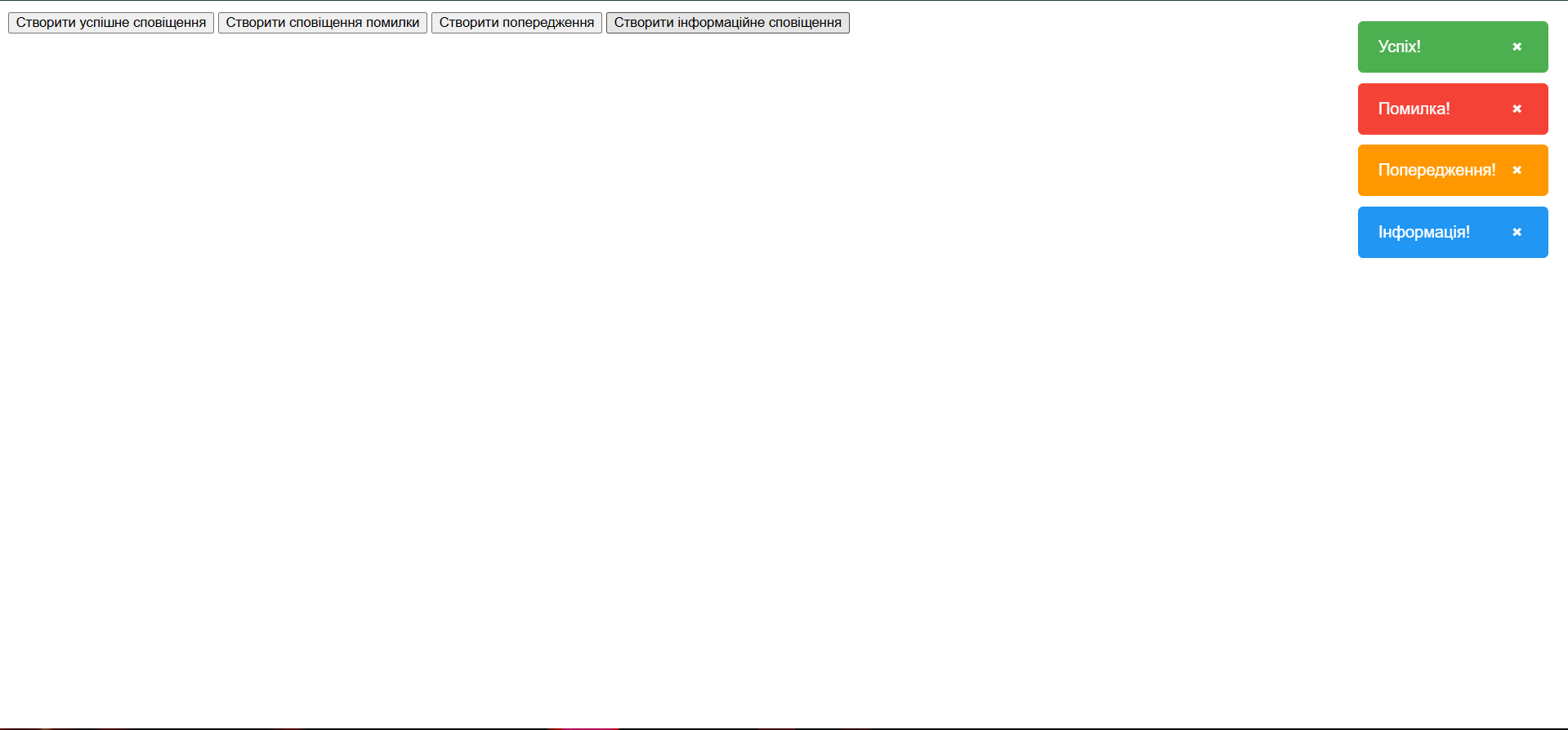


Рис. 6. Результат завдання 6

**Висновок:** В ході виконання лабораторної роботи було реалізовано кілька класів, які демонструють основні принципи об'єктно-орієнтованого програмування (ООП) в JavaScript: інкапсуляцію, спадкування, та поліморфізм. Кожен клас, як-от Notification, Tab, Form, Table, Slider, та Modal, відповідає за певну функціональність і містить методи для управління станом і поведінкою елементів на веб-сторінці. Такі структури роблять код більш читабельним, дозволяють легко розширювати функціональність та забезпечують повторне використання компонентів. ООП дозволяє створювати комплексні веб-інтерфейси, де кожен елемент інкапсулює свою поведінку, що спрощує підтримку та розвиток коду.