**Лабораторна робота № 9**

**ООП в JavaScript***.*

**Хід роботи:**

**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).

**Лістинг програми:**

HTML file:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

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

</head>

<body>

    <button onclick="modalInstance.open()">Modal</button>

    <div id="myModal" class="modal">

        <div class="modal-header" onmousedown="modalInstance.dragStart(event)">

            <h2>Модальне вікно</h2>

        </div>

        <div class="modal-content">

            <label for="login">Логін:</label>

            <input type="text" id="login" name="login"><br><br>

            <label for="password">Пароль:</label>

            <input type="password" id="password" name="password"><br><br>

            <button onclick="modalInstance.login()">Вхід</button>

            <button onclick="modalInstance.close()">Закрити</button>

        </div>

    </div>

    <script src="script.js"></script>

</body>

</html>

Java Script file:

class Modal {

    constructor() {

        this.modal = document.getElementById('myModal');

        this.isDragging = false;

    }

    open() {

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

    }

    close() {

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

    }

    dragStart(e) {

        this.isDragging = true;

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

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

        const dragMove = (e) => {

            if (this.isDragging) {

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

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

            }

        };

        const dragEnd = () => {

            this.isDragging = false;

            document.removeEventListener('mousemove', dragMove);

            document.removeEventListener('mouseup', dragEnd);

        };

        document.addEventListener('mousemove', dragMove);

        document.addEventListener('mouseup', dragEnd);

    }

    login() {

        const loginValue = document.getElementById('login').value;

        const passwordValue = document.getElementById('password').value;

        console.log('Логін:', loginValue);

        console.log('Пароль:', passwordValue);

        this.close();

    }

}

const modalInstance = new Modal();

CSS file:

.modal {

    display: none;

    position: fixed;

    top: 50%;

    left: 50%;

    width: 300px;

    height: 240px;

    margin-left: -150px;

    margin-top: -100px;

    border: 1px solid #000000;

    background-color: #e5ff00;

    z-index: 1;

}

label {

    margin-left: 5%;

}

button {

    width: 20%;

    cursor: pointer;

    left: 50%;

    top: 50%;

    position: relative;

}

.modal-header {

    cursor: move;

    cursor: pointer;

    padding: 20px;

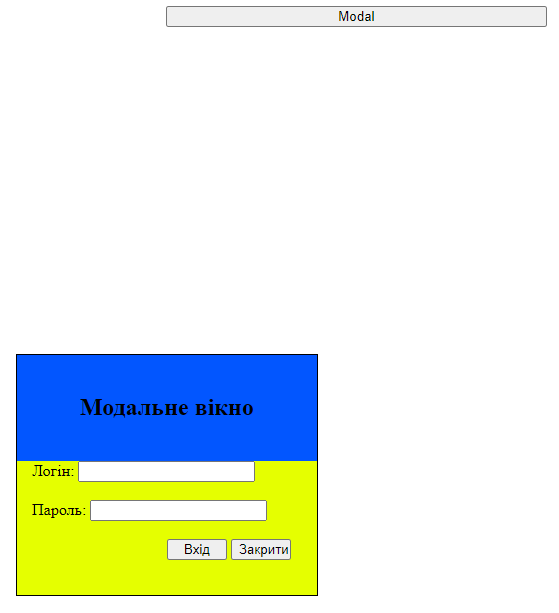
    background-color: #0256ff;

    text-align: center;

}

\

**Результати виконання:**

****

**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.

**Лістинг програми:**

HTML file:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task 2</title>

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

</head>

<body>

    <div id="mySlider" class="slider">

        <div class="slides"></div>

        <div class="slider-btn slider-btn-prev" onclick="mySlider.movePrev()">&#9668;</div>

        <div class="slider-btn slider-btn-next" onclick="mySlider.moveNext()">&#9658;</div>

    </div>

    <script src="script.js"></script>

</body>

</html>

Java Script file:

class Slider {

    constructor(selector, orientation = 'horizontal') {

        this.slider = document.querySelector(selector);

        this.slidesContainer = this.slider.querySelector('.slides');

        this.slides = [];

        this.currentIndex = 0;

        this.orientation = orientation;

        this.initStyles();

    }

    initStyles() {

        this.slider.style.overflow = 'hidden';

        this.slidesContainer.style.display = 'flex';

        this.slidesContainer.style.transition = 'transform 0.5s ease-in-out';

        if (this.orientation === 'vertical') {

            this.slider.style.height = '200px';

            this.slidesContainer.style.flexDirection = 'column';

        } else {

            this.slider.style.width = '200px';

            this.slidesContainer.style.flexDirection = 'row';

        }

    }

    addSlide(content) {

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

        slide.classList.add('slide');

        slide.innerHTML = content;

        this.slides.push(slide);

        this.slidesContainer.appendChild(slide);

        this.updateSlidePosition();

    }

    moveNext() {

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

            this.currentIndex++;

            this.updateSlidePosition();

        }

    }

    movePrev() {

        if (this.currentIndex > 0) {

            this.currentIndex--;

            this.updateSlidePosition();

        }

    }

    updateSlidePosition() {

        const moveAmount = -this.currentIndex \* 200;

        this.slidesContainer.style.transform =

            this.orientation === 'vertical' ? `translateY(${moveAmount}px)` : `translateX(${moveAmount}px)`;

    }

}

const mySlider = new Slider('#mySlider', 'horizontal');

mySlider.addSlide('<img src="1.jpg">');

mySlider.addSlide('<img src="2.jpg">');

mySlider.addSlide('<img src="3.jpg">');

mySlider.addSlide('<img src="4.jpg">');

mySlider.addSlide('<img src="5.jpg">');

mySlider.addSlide('<img src="6.jpg">');

CSS file:

.slider {

    overflow: hidden;

    position: relative;

    width: 200px;

    height: 200px;

    border: 1px solid #ccc;

    top: 50%;

    left: 50%;

}

.slides {

    display: flex;

    transition: transform 0.5s ease-in-out;

}

.slide {

    flex: 0 0 200px;

    width: 200px;

    height: 200px;

}

img {

    max-width: 100%;

    max-height: 100%;

    object-fit: cover;

}

.slider-btn {

    position: absolute;

    top: 50%;

    transform: translateY(-50%);

    cursor: pointer;

    background-color: #f1f1f1;

    border: 1px solid #000000;

    padding: 2px;

    color: black;

}

.slider-btn-prev {

    left: 5px;

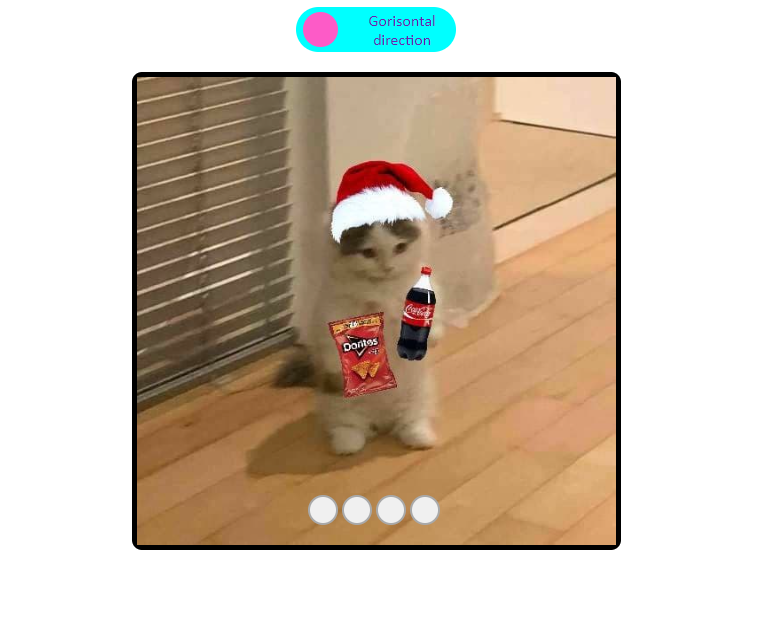
}

.slider-btn-next {

    right: 5px;

}

**Результати виконання:**

****

**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.

**Лістинг програми:**

HTML File:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task 3</title>

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

</head>

    <body>

        <div id="myTable"></div>

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

    </body>

</html>

JavaScript file:

class Table {

    constructor(data, selector) {

        this.data = data;

        this.tableContainer = document.querySelector(selector);

        this.table = document.createElement('table');

        this.tableContainer.appendChild(this.table);

        this.renderTable();

        this.setupManualEntryForm();

    }

    renderTable() {

        this.data.forEach(rowData => this.addRow(rowData));

    }

    addRow(rowData) {

        const row = this.table.insertRow();

        rowData.forEach(cellData => {

            const cell = row.insertCell();

            cell.textContent = cellData;

        });

    }

    addColumn(columnData) {

        this.table.tHead.insertRow().insertCell().textContent = columnData[0];

        columnData.slice(1).forEach((cellData, rowIndex) => {

            const cell = this.table.rows[rowIndex + 1].insertCell();

            cell.textContent = cellData;

        });

    }

    setupManualEntryForm() {

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

        inputRow.classList.add('input-row');

        this.data[0].forEach(header => {

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

            input.setAttribute('placeholder', header);

            inputRow.appendChild(input);

        });

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

        addButton.textContent = 'Add';

        addButton.addEventListener('click', () => this.handleAddManualEntry());

        inputRow.appendChild(addButton);

        this.tableContainer.appendChild(inputRow);

    }

    handleAddManualEntry() {

        const inputRow = this.tableContainer.querySelector('.input-row');

        const inputElements = inputRow.querySelectorAll('input');

        const rowData = Array.from(inputElements).map(input => input.value);

        this.addRow(rowData);

        inputElements.forEach(input => (input.value = ''));

    }

}

const initialData = [

    ['Name', 'Age', 'Country'],

    ['John Doe', '30', 'USA'],

    ['Jane Doe', '25', 'Canada']

];

const myTable = new Table(initialData, '#myTable');

CSS file:

table {

    width: 35%;

    margin-top: 25px;

}

table,

th,

td {

    border: 1px solid #000000;

    margin-bottom: 50px;

}

th,

td {

    padding: 10px;

    text-align: left;

}

input {

    margin-right: 10px;

    margin-bottom: 10px;

}

button {

    margin-bottom: 20px;

}

body{

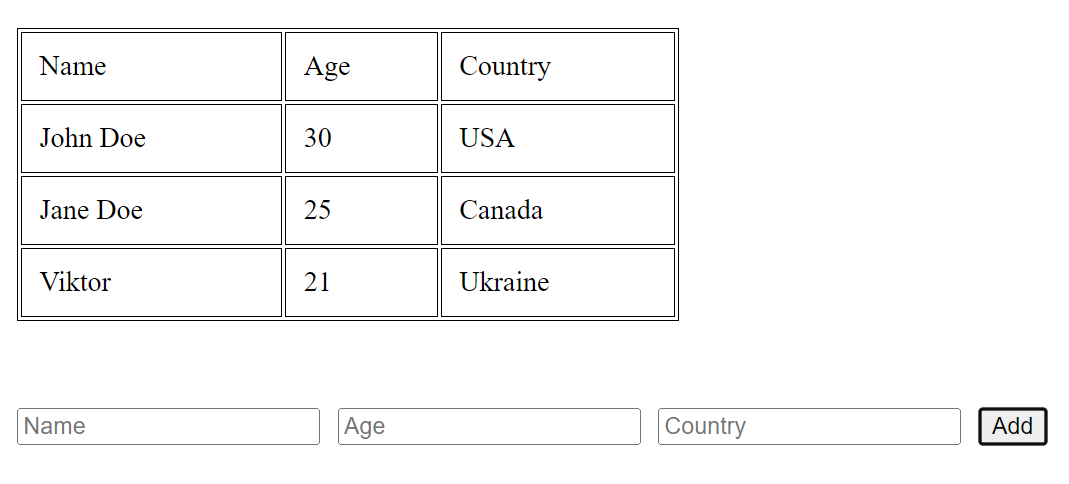
    position: relative;

    top: 35%;

    left: 35%;

}}

**Результати виконання:**



**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.

**Лістинг програми:**

HTML file:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task 4</title>

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

</head>

<body>

    <div id="myForm">

        <h1>My Form</h1>

    </div>

    <script src="script.js"></script>

</body>

</html>

JavaScript file:

class Form {

    constructor(selector) {

        this.formContainer = document.querySelector(selector);

        this.form = document.createElement('form');

        this.fields = [];

        this.formContainer.appendChild(this.form);

    }

    addInput(type, name, placeholder) {

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

        input.type = type;

        input.name = name;

        input.placeholder = placeholder;

        this.fields.push(input);

        this.form.appendChild(input);

    }

    addButton(label, callback) {

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

        button.textContent = label;

        button.addEventListener('click', event => {

            event.preventDefault();

            this.validateAndSubmit(callback);

        });

        this.form.appendChild(button);

    }

    validateAndSubmit(callback) {

        const formData = {};

        for (const field of this.fields) {

            const value = field.value.trim();

            const name = field.name;

            if (!value) {

                alert(`Field "${name}" cannot be empty.`);

                return;

            }

            formData[name] = value;

        }

        if (callback)

            callback(formData);

    }

}

const myForm = new Form('#myForm');

myForm.addInput('Text', 'Name', 'Your Name');

myForm.addInput('Email', 'Email', 'Your Email');

myForm.addInput('Password', 'Password', 'Your Password');

myForm.addButton('Submit', data => {

    console.log('Submitted Data:', data);

});

CSS file:

body {

    font-family: 'Arial', sans-serif;

    background-color: #f4f4f4;

    display: flex;

    align-items: center;

    justify-content: center;

    height: 100vh;

    margin: 0;

}

#myForm {

    background-color: #fffffff4;

    padding: 20px;

    border-radius: 8px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

    width: 500px;

    text-align: center;

}

input {

    width: 100%;

    padding: 10px;

    margin-bottom: 10px;

    box-sizing: border-box;

    border: 1px solid #000000;

    border-radius: 4px;

    font-size: 14px;

}

button {

    background-color: #d200d9;

    color: #fff;

    padding: 10px;

    border: none;

    border-radius: 4px;

    cursor: pointer;

    font-size: 16px;

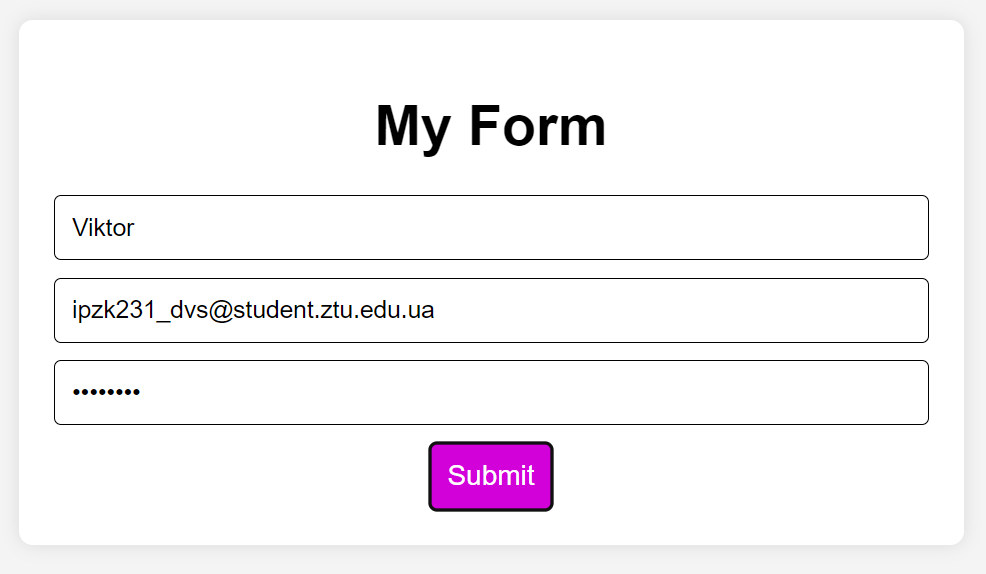
}

button:hover {

    background-color: #b8016f;

}

**Результати виконання:**

****

****

**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.

**Лістинг програми:**

HTML file:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task 5</title>

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

</head>

    <body>

        <div id="myTabs"></div>

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

    </body>

</html>

Java Script file:

class Tab {

    constructor(selector) {

        this.tabsContainer = document.querySelector(selector);

        this.tabs = [];

        this.contents = [];

    }

    addTab(tabName, content) {

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

        tab.classList.add('tab');

        tab.textContent = tabName;

        tab.addEventListener('click', () => this.switchTab(tab));

        this.tabs.push(tab);

        this.tabsContainer.appendChild(tab);

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

        contentContainer.classList.add('tab-content', 'hidden');

        contentContainer.innerHTML = content;

        this.contents.push(contentContainer);

        this.tabsContainer.appendChild(contentContainer);

        if (this.tabs.length === 1) {

            this.switchTab(tab);

        }

    }

    switchTab(selectedTab) {

        this.tabs.forEach((tab, index) => {

            const content = this.contents[index];

            if (tab === selectedTab) {

                tab.classList.add('active');

                content.classList.remove('hidden');

            } else {

                tab.classList.remove('active');

                content.classList.add('hidden');

            }

        });

    }

}

const myTabs = new Tab('#myTabs');

myTabs.addTab('Tab 1', '<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Autem, numquam! Delectus consequuntur repellat eos rerum ut fugit architecto asperiores commodi iste natus aliquam alias, itaque ratione accusantium quod eum at.</p>');

myTabs.addTab('Tab 2', '<p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Unde repudiandae libero repellat provident ex facilis ab expedita aut beatae, est itaque modi labore, totam porro! Quidem ab sequi amet commodi.</p>');

myTabs.addTab('Tab 3', '<p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Vitae voluptatibus ex nam impedit repellendus labore, molestiae quidem fuga esse minima, asperiores possimus eum nisi ipsam voluptas similique tempora culpa repellat?</p>');

CSS file:

.tabs {

    display: flex;

    margin-bottom: 20px;

}

.tab {

    padding: 10px;

    cursor: pointer;

    border: 1px solid #000000;

    border-radius: 4px 4px 0 0;

    margin-right: -1px;

}

.tab-content {

    border: 1px solid #000000;

    padding: 15px;

    border-radius: 0 0 2px 2px;

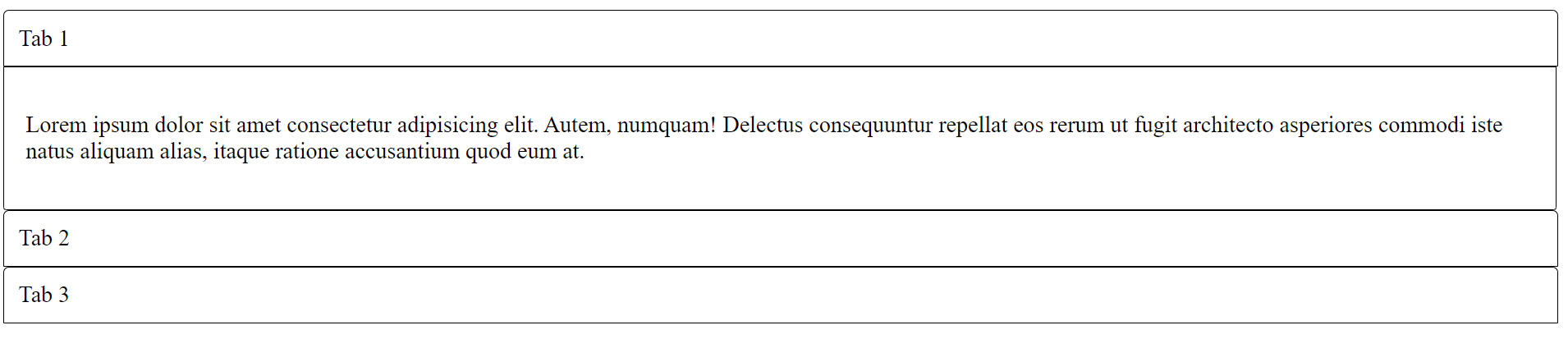
}

.hidden {

    display: none;

}

**Виконання програми:**

****

**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.

**Лістинг програми:**

HTML File:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task 6</title>

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

</head>

    <body>

        <button onclick="showSuccess()">Показати Успішне Повідомлення</button>

        <button onclick="showError()">Показати Помилкове Повідомлення</button>

        <button onclick="showInfo()">Показати Шнформаційне Повідомлення</button>

        <div id="myNotification" class="notification"></div>

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

    </body>

</html>

Java Script file:

class Notification {

    constructor() {

        this.container = document.getElementById('myNotification');

    }

    show(message, type) {

        this.container.textContent = message;

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

        this.container.style.display = 'block';

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

        closeBtn.innerHTML = '&times;';

        closeBtn.className = 'close-btn';

        closeBtn.addEventListener('click', () => this.hide());

        this.container.appendChild(closeBtn);

    }

    hide() {

        this.container.style.display = 'none';

    }

}

const notification = new Notification();

function showSuccess() {

    notification.show('Успішне Повідомлення!', 'success');

}

function showError() {

    notification.show('Помилкове Повідомлення!', 'error');

}

function showInfo() {

    notification.show('Інформаційне Повідомлення!', 'info');

}

CSS file:

body {

    font-family: 'Helvetica Neue', sans-serif;

    background-color: #f4f4f4;

    display: flex;

    align-items: center;

    justify-content: center;

    height: 100vh;

    margin: 0;

}

.notification {

    position: fixed;

    top: 50px;

    padding: 15px;

    border-radius: 25px;

    color: #fff;

    font-size: 15px;

    display: none;

}

.success {

    background-color: #00ff08;

}

.error {

    background-color: #a90b00;

}

.info {

    background-color: #0207f7;

}

.close-btn {

    cursor: pointer;

    margin-left: 10px;

    font-size: 20px;

    font-weight: bold;

}

button {

    background-color: #00d0ff;

    color: #ffffff;

    padding: 15px;

    border: none;

    border-radius: 25px;

    cursor: pointer;

    font-size: 15px;

    margin-right: 20px;

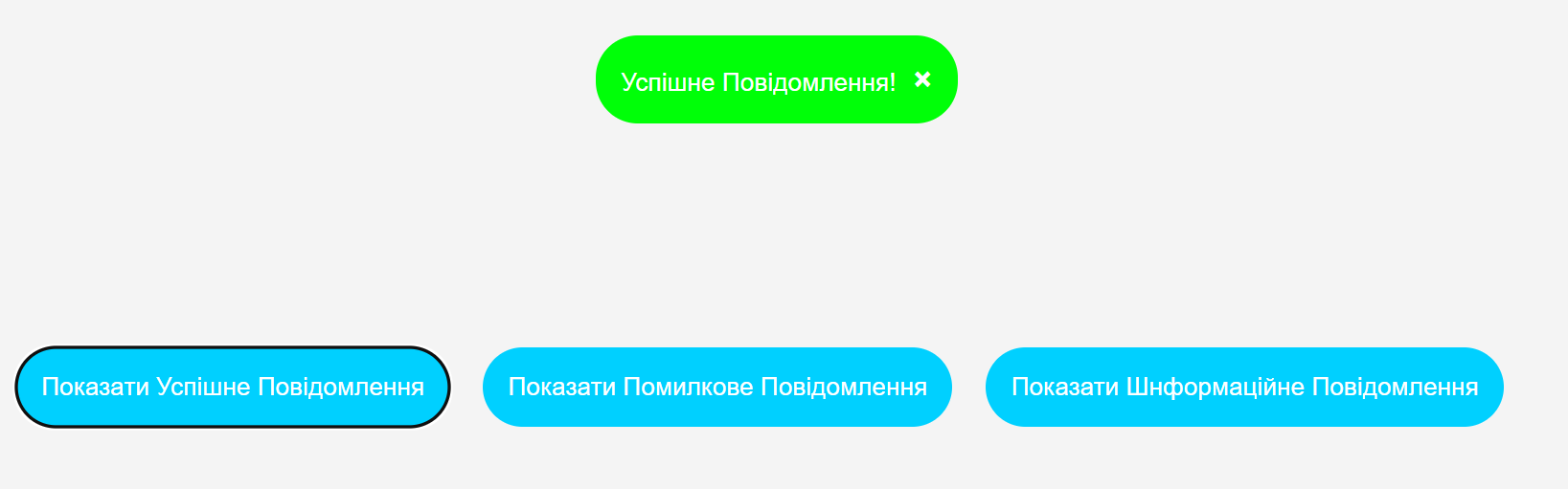
}

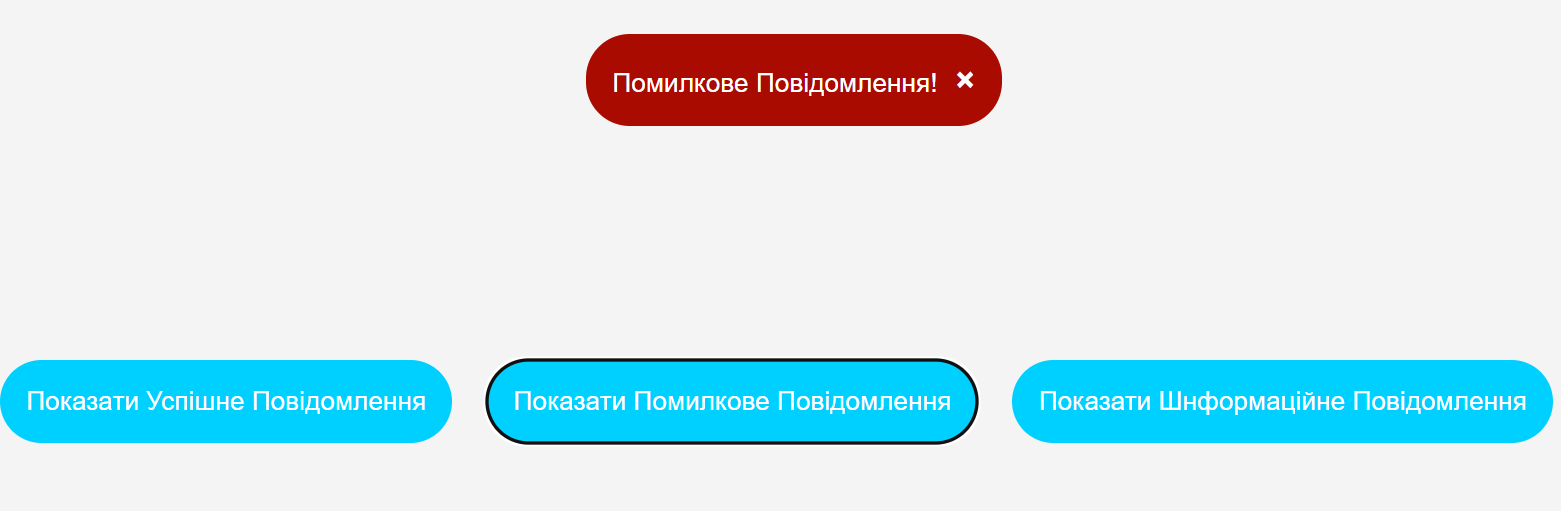
button:hover {

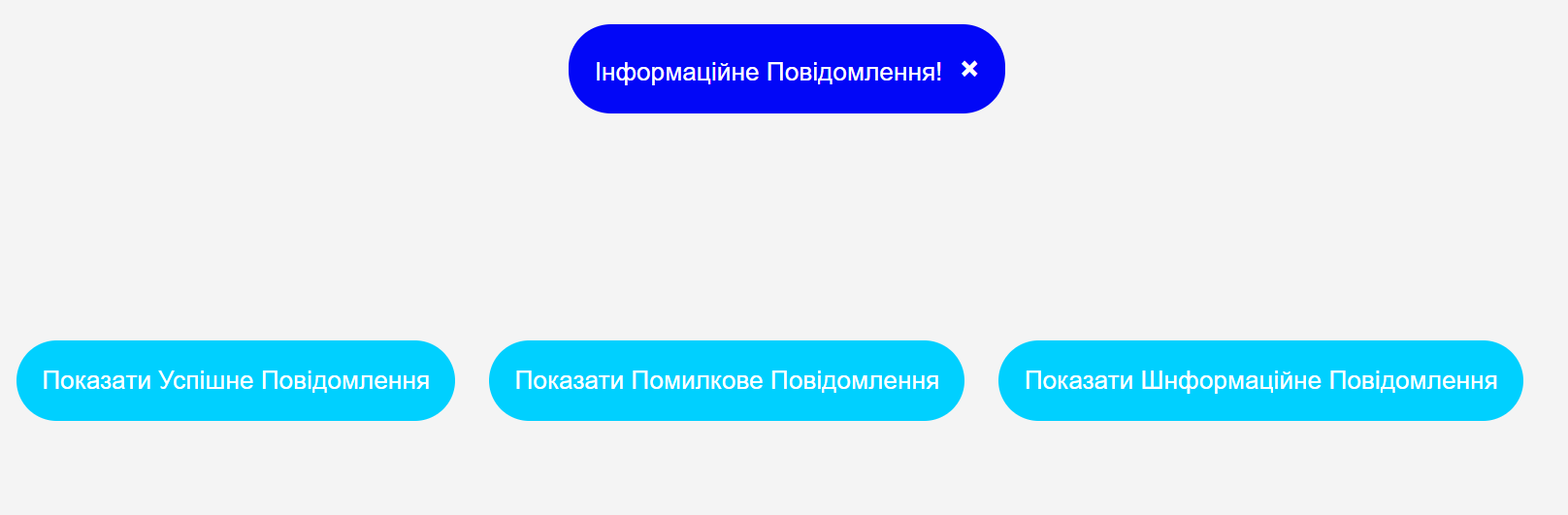
    background-color: #ffe600;

}

**Виконання програми:**

****

****

****

GitHub:

https://github.com/djproblematic/frontend