**Лабораторна робота №7**

**Тема**: Події миші та клавіатури.

**Мета роботи**: вивчити особливості обробки подій миші та клавіатури

**Завдання 1.**

Надати квадратикам можливість перетягування (draggable) в межах контейнера. При переміщенні курсора, наведеному на даний блок-квадратик, та утриманні кнопки миші даний елемент повинен перетягуватись. Квадратики не повинні заходити навіть частково за межі контейнера.

**Лістинг коду**

**HTML:**

 <div id="container">

        <div class="square" id="square1"></div>

        <div class="square" id="square2"></div>

    </div>

**CSS:**

#container {

            position: relative;

            width: 400px;

            height: 400px;

            border: 1px solid #ccc;

        }

        .square {

            width: 50px;

            height: 50px;

            background-color: green;

            position: absolute;

            cursor: grab;

        }

**JavaScript:**

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

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

            const squares = document.querySelectorAll(".square");

            squares.forEach(square => {

                makeDraggable(square);

            });

            function makeDraggable(element) {

                let isDragging = false;

                let offset = { x: 0, y: 0 };

                element.addEventListener("mousedown", function(e) {

                    isDragging = true;

                    offset = {

                        x: e.clientX - element.getBoundingClientRect().left,

                        y: e.clientY - element.getBoundingClientRect().top

                    };

                    element.style.cursor = "grabbing";

                });

                document.addEventListener("mousemove", function(e) {

                    if (!isDragging) return;

                    const x = e.clientX - offset.x;

                    const y = e.clientY - offset.y;

                    const maxX = container.clientWidth - element.clientWidth;

                    const maxY = container.clientHeight - element.clientHeight;

                    const boundedX = Math.min(Math.max(x, 0), maxX);

                    const boundedY = Math.min(Math.max(y, 0), maxY);

                    element.style.left = `${boundedX}px`;

                    element.style.top = `${boundedY}px`;

                });

                document.addEventListener("mouseup", function() {

                    isDragging = false;

                    element.style.cursor = "grab";

                });

            }

        });

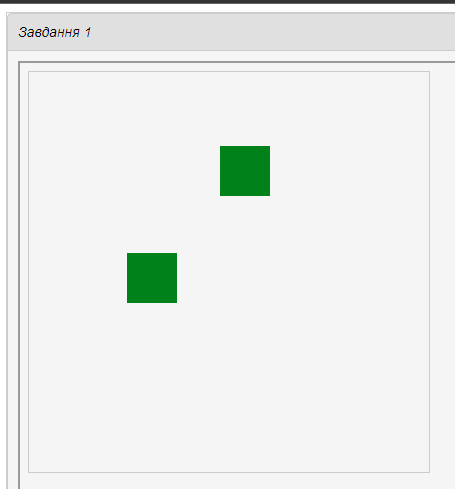


Рис 7.1

**Завдання 2.**

Розмістіть на сторінці випадковим чином 20 кольорових кружечків з радіусами від 10 до 30.

**Лістинг коду**

**HTML:**

    <div id="circle-container"></div>

**CSS:**

        #circle-container {

            position: relative;

            width: 600px;

            height: 400px;

            border: 1px solid #ccc;

            margin: 20px;

            overflow: hidden;

        }

        .circle {

            width: 30px;

            height: 30px;

            border-radius: 50%;

            background-color: brown;

            position: absolute;

            cursor: pointer;

        }

        .active {

            border: 2px solid black;

        }

**JavaScript:**

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

            const container = document.getElementById("circle-container");

            const circles = createRandomCircles(20);

            let activeIndex = 0;

            circles[activeIndex].classList.add('active');

            function createRandomCircles(numCircles) {

                const result = [];

                for (let i = 0; i < numCircles; i++) {

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

                    circle.className = 'circle';

                    const radius = Math.floor(Math.random() \* (30 - 10 + 1)) + 10;

                    const left = Math.floor(Math.random() \* (container.clientWidth - radius \* 2)) + radius;

                    const top = Math.floor(Math.random() \* (container.clientHeight - radius \* 2)) + radius;

                    circle.style.width = `${radius \* 2}px`;

                    circle.style.height = `${radius \* 2}px`;

                    circle.style.left = `${left}px`;

                    circle.style.top = `${top}px`;

                    circle.addEventListener('click', () => setActiveCircle(i));

                    container.appendChild(circle);

                    result.push(circle);

                }

                return result;

            }

            function setActiveCircle(index) {

                circles[activeIndex].classList.remove('active');

                activeIndex = index;

                circles[activeIndex].classList.add('active');

            }

            document.addEventListener('keydown', function (event) {

                if (event.key === 'Tab') {

                    event.preventDefault();

                    const direction = event.shiftKey ? -1 : 1;

                    let newIndex = (activeIndex + direction + circles.length) % circles.length;

                    setActiveCircle(newIndex);

                } else if (event.key.startsWith('Arrow')) {

                    event.preventDefault();

                    const direction = getDirection(event.key);

                    moveActiveCircle(direction);

                }

            });

            function getDirection(arrowKey) {

                switch (arrowKey) {

                    case 'ArrowUp':

                        return 'up';

                    case 'ArrowDown':

                        return 'down';

                    case 'ArrowLeft':

                        return 'left';

                    case 'ArrowRight':

                        return 'right';

                    default:

                        return '';

                }

            }

            function moveActiveCircle(direction) {

                const step = 10;

                const circle = circles[activeIndex];

                let newTop = parseInt(circle.style.top) || 0;

                let newLeft = parseInt(circle.style.left) || 0;

                switch (direction) {

                    case 'up':

                        newTop = Math.max(newTop - step, 0);

                        break;

                    case 'down':

                        newTop = Math.min(newTop + step, container.clientHeight - circle.clientHeight);

                        break;

                    case 'left':

                        newLeft = Math.max(newLeft - step, 0);

                        break;

                    case 'right':

                        newLeft = Math.min(newLeft + step, container.clientWidth - circle.clientWidth);

                        break;

                }

                circle.style.top = `${newTop}px`;

                circle.style.left = `${newLeft}px`;

            }

        });



Рис 7.2

**Завдання 3.**

Реалізуйте «утікаючу» кнопку. В div-елементі знаходиться кнопка. Проте при наведенні на неї курсора миші, вона змінює своє положення так, що унеможливлює натиснення на неї. Кнопка не повинна виходити за межі div-елемента.

**Лістинг коду**

**HTML:**

<div id="container">

        <button id="teleportingButton">нажми</button>

    </div>

**CSS:**

 #container {

            position: relative;

            width: 400px;

            height: 300px;

            border: 1px solid #ccc;

            overflow: hidden;

        }

        #teleportingButton {

            position: absolute;

            padding: 10px;

            cursor: pointer;

        }

**JavaScript:**

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

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

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

            container.addEventListener("mousemove", function(e) {

                teleportingButton.addEventListener("mouseover", function() {

                    const containerRect = container.getBoundingClientRect();

                    const buttonRect = teleportingButton.getBoundingClientRect();

                    const newX = Math.floor(Math.random() \* (containerRect.width - buttonRect.width));

                    const newY = Math.floor(Math.random() \* (containerRect.height - buttonRect.height));

                    teleportingButton.style.left = newX + "px";

                    teleportingButton.style.top = newY + "px";

                });

            });

        });



Рис 7.3

**Завдання 4.**

Розмістіть список з елементів, які можна буде виділяти, як у файлових менеджерах.

**Лістинг коду**

**HTML:**

    <ul id="selectableList">

        <li>Елемент 1</li>

        <li>Елемент 2</li>

        <li>Елемент 3</li>

        <li>Елемент 4</li>

        <li>Елемент 5</li>

    </ul>

**CSS:**

        ul {

            list-style: none;

            padding: 0;

        }

        li {

            padding: 10px;

            border: 1px solid #ccc;

            margin: 5px;

            cursor: pointer;

        }

        .selected {

            background-color: #3498db;

            color: #fff;

        }

**JavaScript:**

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

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

            const listItems = selectableList.getElementsByTagName("li");

            selectableList.addEventListener("click", function(e) {

                const clickedItem = e.target;

                if (e.ctrlKey || e.metaKey) {

                    clickedItem.classList.toggle("selected");

                } else {

                    Array.from(listItems).forEach(item => {

                        if (item !== clickedItem) {

                            item.classList.remove("selected");

                        }

                    });

                    clickedItem.classList.toggle("selected");

                }

            });

        });

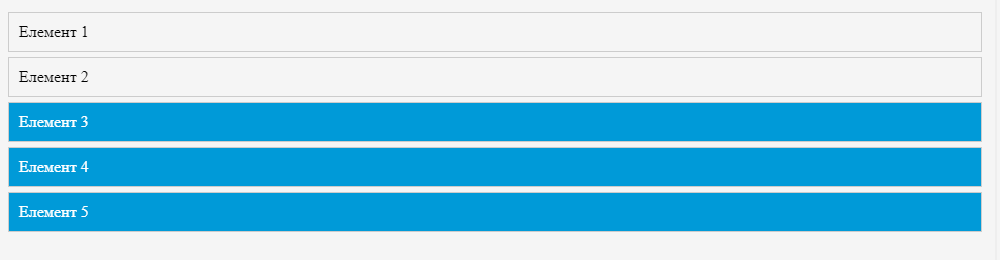


Рис 7.4

**Завдання 5.**

Реалізуйте елемент “слайдер”

**Лістинг коду**

**HTML:**

 <div id="slider">

        <div id="sliderHandle"></div>

    </div>

**CSS:**

  #slider {

            width: 300px;

            height: 20px;

            background-color: #ccc;

            position: relative;

            margin: 50px auto;

            cursor: pointer;

        }

        #sliderHandle {

            width: 20px;

            height: 50px;

            background-color: orange;

            border-radius: 20%;

            position: absolute;

            top: 50%;

            transform: translateY(-50%);

        }

**JavaScript:**

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

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

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

            let isDragging = false;

            sliderHandle.addEventListener("mousedown", function(e) {

                isDragging = true;

            });

            document.addEventListener("mouseup", function() {

                isDragging = false;

            });

            document.addEventListener("mousemove", function(e) {

                if (isDragging) {

                    const sliderRect = slider.getBoundingClientRect();

                    const newPosition = Math.min(Math.max(e.clientX - sliderRect.left, 0), sliderRect.width);

                    sliderHandle.style.left = newPosition + "px";

                }

            });

        });



Рис 7.5