UNIVERSITI MALAYSIA TERENGGANU

Faculty of Computer Sciences and Mathematics

Front-End Programming

CSM3103

Lab Report 4

Prepared for:

Dr. Rabiei bin Mamat

Prepared by:

Ahmad Afif Syahmi bin Ahmad Rozali (S65526)

29th April 2024

Bachelor of Computer Science (Mobile Computing) with Honors

Semester II 2023/2024

Task 1 : JavaScript Function

- Codes
    - task1.html

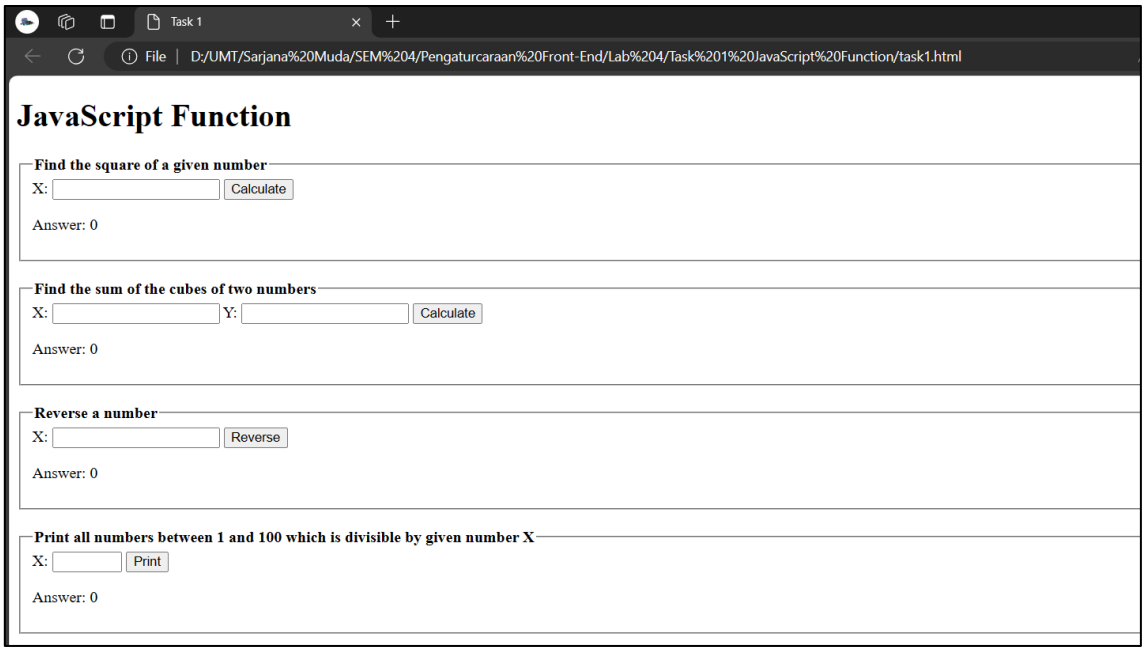
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Task 1</title>
</head>
<body>

    <h1>JavaScript Function</h1>

    <fieldset>
        <legend><b>Find the square of a given
number</b></legend>
        <label for="num1">X: </label>
        <input type="number" id="num1">
        <button onclick="findSquare()">Calculate</button>
        <br>
        <p id="answer1">Answer: 0</p>
    </fieldset>
    <br>

    <fieldset>
        <legend><b>Find the sum of the cubes of two
numbers</b></legend>
        <label for="num2X">X: </label>
        <input type="number" id="num2X">
        <label for="num2Y">Y: </label>
        <input type="number" id="num2Y">
        <button onclick="findSumSquare()">Calculate</button>
        <p id="answer2">Answer: 0</p>
    </fieldset>
    <br>

    <fieldset>
        <legend><b>Reverse a number</b></legend>
        <label for="num3">X: </label>
        <input type="number" id="num3">
        <button onclick="findReversedNumber()">Reverse</button>
        <p id="answer3">Answer: 0</p>
    </fieldset>
    <br>

    <fieldset>
        <legend><b>Print all numbers between 1 and 100 which is
divisible by given number X</b></legend>
        <label for="num4">X: </label>
        <input type="number" id="num4" min="1" max="100">
        <button onclick="findDivisibleNumbers()">Print</button>
        <p id="answer4">Answer: 0</p>
```

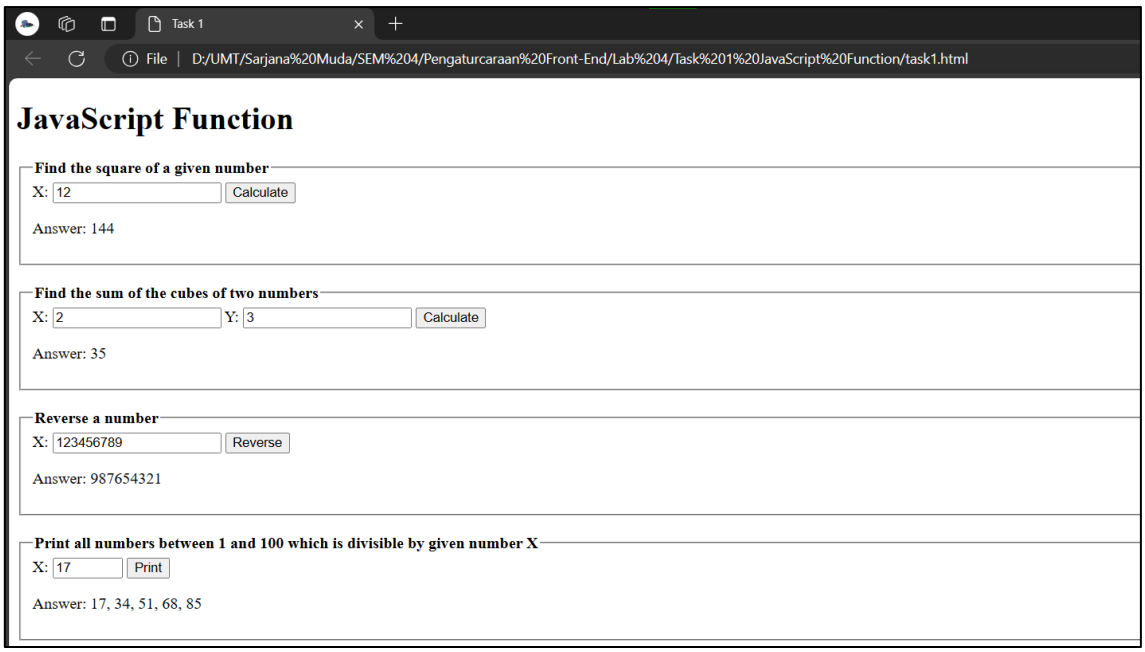
```
    </fieldset>

    <script src="task1.js"></script>

</body>
</html>
```

- task1.js

```
// Function to calculate the square of a number
function findSquare() {
    const x = document.getElementById("num1").value; // Get the
value from input field
    document.getElementById("answer1").innerHTML = "Answer: " +
(x * x); // Display the answer
}

// Function to calculate the sum of cubes of two numbers
function findSumSquare() {
    const x = document.getElementById("num2X").value;
    const y = document.getElementById("num2Y").value;
    document.getElementById("answer2").innerHTML = "Answer: " +
((x * x * x) + (y * y * y)); // Calculate and display the sum of
cubes
}

// Function to reverse a number
function findReversedNumber() {
    const x = document.getElementById("num3").value;
    const reversedStr =
x.toString().split('').reverse().join('');
    document.getElementById("answer3").innerHTML = "Answer: " +
reversedStr;
}

// Function to print numbers divisible by a given number
(between 1 and 100)
function findDivisibleNumbers() {
    const z = document.getElementById("num4").value;
    let nums = "";
    for (let i = 1; i <= 100; i++) {
        if ((i % z) === 0) {
            if (nums === "") {
                nums = i.toString();
            }
            else {
                nums += ", " + i;
            }
        }
    }
    document.getElementById("answer4").innerHTML = "Answer: " +
nums;
}
```

- Output


Task 1
File | D:/UMT/Sarjana%20Muda/SEM%204/Pengaturcaraan%20Front-End/Lab%204/Task%201%20JavaScript%20Function/task1.html
JavaScript Function
Find the square of a given number
X: Calculate
Answer: 0
Find the sum of the cubes of two numbers
X: Y: Calculate
Answer: 0
Reverse a number
X: Reverse
Answer: 0
Print all numbers between 1 and 100 which is divisible by given number X
X: Print
Answer: 0


Task 1
File | D:/UMT/Sarjana%20Muda/SEM%204/Pengaturcaraan%20Front-End/Lab%204/Task%201%20JavaScript%20Function/task1.html
JavaScript Function
Find the square of a given number
X: 12 Calculate
Answer: 144
Find the sum of the cubes of two numbers
X: 2 Y: 3 Calculate
Answer: 35
Reverse a number
X: 123456789 Reverse
Answer: 987654321
Print all numbers between 1 and 100 which is divisible by given number X
X: 17 Print
Answer: 17, 34, 51, 68, 85

Task 2 : JavaScript Recursion Function

- Codes
    - task2.html

```
<!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>
</head>
<body>

    <h1>JavaScript Recursion Function</h1>

    <fieldset>
        <legend><b>Find sum of digits of a number</b></legend>
        <label for="digits">X: </label>
        <input type="number" id="digits">
        <button onclick="findSumOfDigits()">Calculate</button>
        <br>
        <p id="answer1">Answer: 0</p>
    </fieldset>
    <br>

    <fieldset>
        <legend><b>Calculate X raised to the power of
Y</b></legend>
        <label for="base">X: </label>
        <input type="number" id="base">
        <label for="power">Y: </label>
        <input type="number" id="power">
        <button onclick="calculatePower()">Calculate</button>
        <br>
        <p id="answer2">Answer: 0</p>
    </fieldset>
    <br>

    <script src="task2.js"></script>

</body>
</html>
```

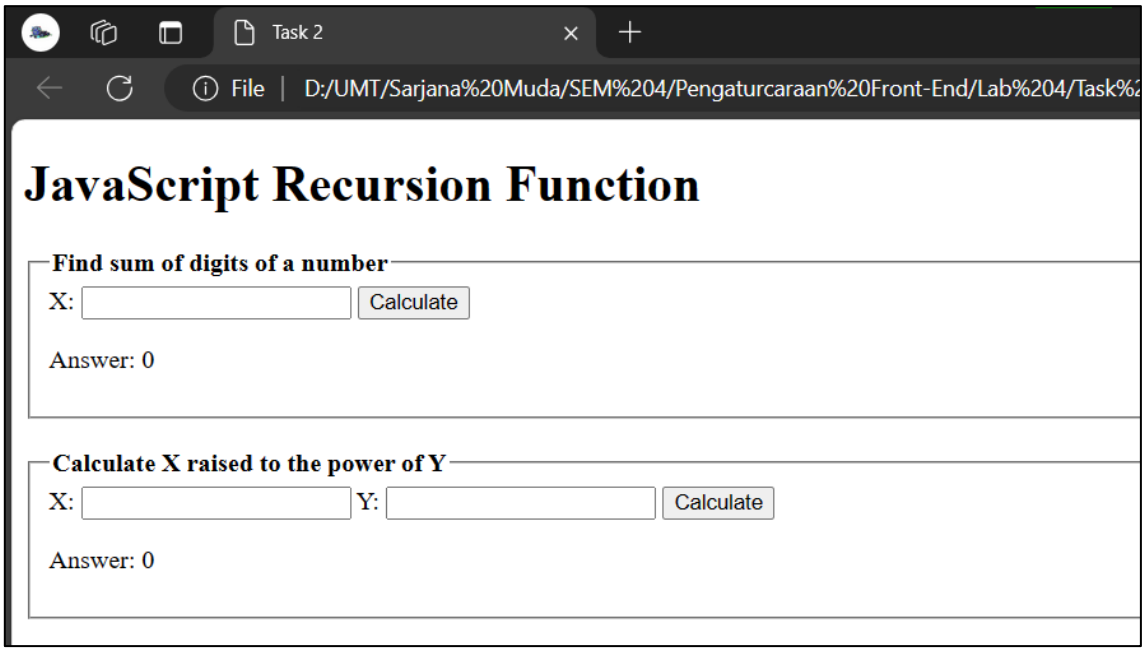
    - task2.js

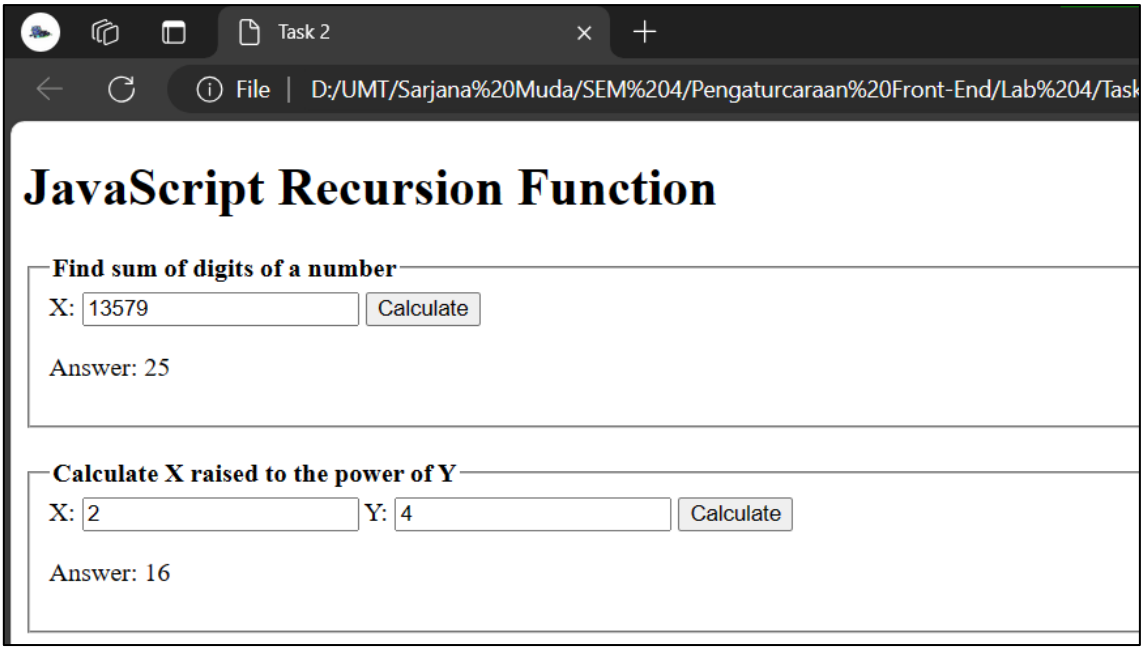
```
// Function to find the sum of digits of a number using
recursion
function findSumOfDigits() {
    const number = document.getElementById("digits").value;

    // Basic check for empty input
    if (number === "") {
```

```
      document.getElementById("answer1").innerHTML = "Error:
Please enter a number.";
      return;
    }

    // Check for non-numeric input
    if (isNaN(number)) {
        document.getElementById("answer1").innerHTML = "Error:
Please enter a valid number.";
        return;
    }

    const sum = calculateSumOfDigits(number);
    document.getElementById("answer1").innerHTML = "Answer: " +
sum;
}

// Function to calculate the sum of digits recursively
function calculateSumOfDigits(num) {
    // Base case: If the number is less than 10, return the
number itself (single digit)
    if (num < 10) {
      return num;
    }

    // Get the last digit using modulo operator (%)
    const lastDigit = num % 10;

    // Recursively call the function with the remaining digits
(excluding the last digit)
    const remainingDigitsSum =
calculateSumOfDigits(Math.floor(num / 10));

    // Calculate the sum of the last digit and the sum from
remaining digits
    const totalSum = lastDigit + remainingDigitsSum;

    return totalSum;
}

// Function to calculate x raised to the power y using recursion
function calculatePower() {
    const base =
parseInt(document.getElementById("base").value);
    const power =
parseInt(document.getElementById("power").value);

    // Basic check for invalid input (non-numeric or negative
power)
    if (isNaN(base) || isNaN(power) || power < 0) {
        document.getElementById("answer2").innerHTML = "Error:
Please enter valid numbers for base and non-negative power.";
        return;
    }

    const result = calculatePowerRecursive(base, power);
    document.getElementById("answer2").innerHTML = "Answer: " +
result;
}
```

```
// Recursive function to calculate x raised to the power y
function calculatePowerRecursive(base, power) {
    // Base case: power is 0, anything raised to the power 0 is
1
    if (power === 0) {
      return 1;
    }

    // Recursive case: x raised to the power y is x multiplied
by itself (y-1) times
    return base * calculatePowerRecursive(base, power - 1);
}
```

- Output




Task 2
File | D:/UMT/Sarjana%20Muda/SEM%204/Pengaturcaraan%20Front-End/Lab%204/Task
JavaScript Recursion Function
Find sum of digits of a number
X: 13579
Calculate
Answer: 25
Calculate X raised to the power of Y
X: 2
Y: 4
Calculate
Answer: 16

Task 3 : JavaScript Object and Prototype

- Codes
    - task3.html

```
<!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>
</head>
<body>

    <h1>JavaScript Object and Prototype</h1>

    <fieldset>
        <legend><b>Object product</b></legend>
        <ul>
            <li id="prod1"></li>
            <li id="prod2"></li>
            <li id="prod3"></li>
        </ul>
    </fieldset>
    <br>

    <fieldset>
        <legend><b>Object book</b></legend>
        <ul>
            <li id="book1"></li>
            <li id="book2"></li>
            <li id="book3"></li>
        </ul>
    </fieldset>
    <br>

    <fieldset>
        <legend><b>Object employee</b></legend>
        <ul>
            <li id="emp1"></li>
            <li id="emp2"></li>
            <li id="emp3"></li>
        </ul>
    </fieldset>
    <br>

    <fieldset>
        <legend><b>Object manager</b></legend>
        <ul>
            <li id="man1"></li>
            <li id="man2"></li>
            <li id="man3"></li>
            <li id="man4"></li>
            <li id="man5"></li>
        </ul>
```

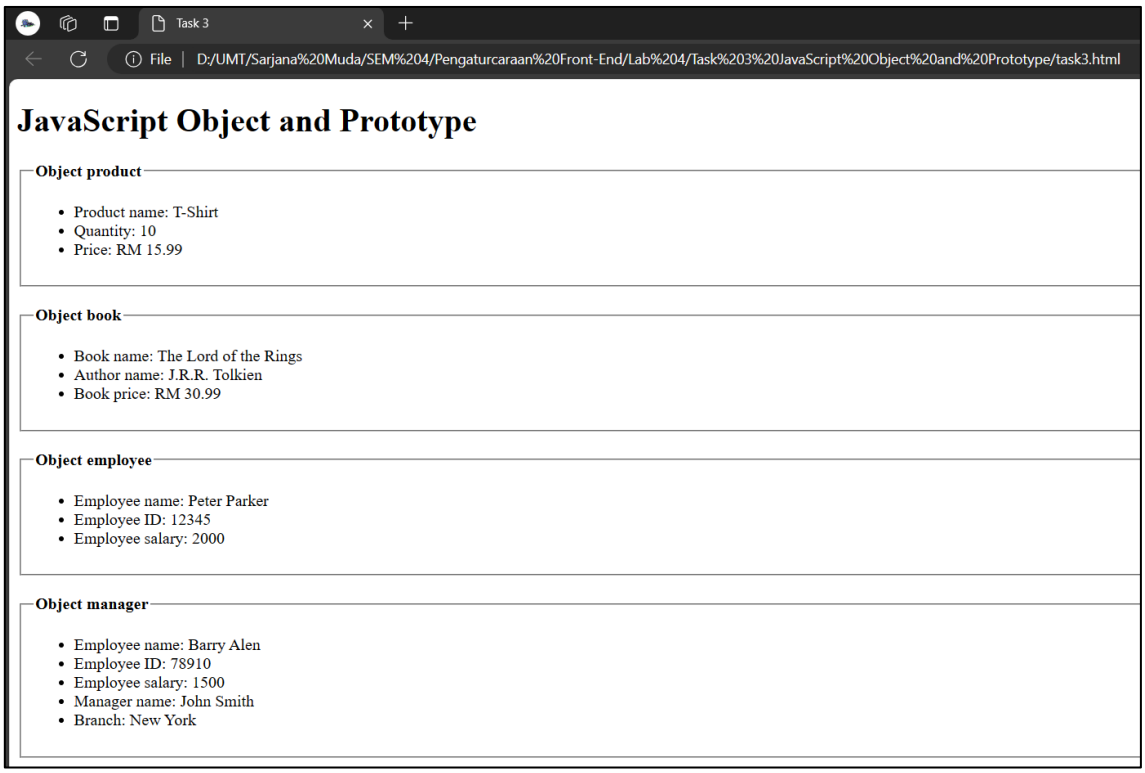
```
    </fieldset>

    <script src="task3.js"></script>

</body>
</html>
```

- task3.js

```
//Instantiate object product
const product = {
    name: "T-Shirt",
    quantity: 10,
    price: 15.99
}

//Display object product
document.getElementById("prod1").innerHTML = "Product name: " +
product.name;
document.getElementById("prod2").innerHTML = "Quantity: " +
product.quantity;
document.getElementById("prod3").innerHTML = "Price: RM " +
product.price.toFixed(2);

//Object book constructor
function Book(name, authorName) {
    this.name = name;
    this.authorName = authorName;
}

//Instantiate object book
const book = new Book("The Lord of the Rings", "J.R.R.
Tolkien");

//Add the prototype property price
Book.prototype.price = 30.99;

//Display object book
document.getElementById("book1").innerHTML = "Book name: " +
book.name;
document.getElementById("book2").innerHTML = "Author name: " +
book.authorName;
document.getElementById("book3").innerHTML = "Book price: RM " +
book.price.toFixed(2);

//Parent object employee constructor
function Employee(name, id, salary) {
    this.name = name;
    this.id = id;
    this.salary = salary;
}

//Child object Manager construtor
function Manager(name, id, salary, managerName, branch) {
    Employee.call(this, name, id, salary);
    this.managerName = managerName;
    this.branch = branch;
```

```
}

//Inherit all properties from Employee
Manager.prototype = Object.create(Employee.prototype);
Manager.prototype.constructor = Manager;

//Instantiate Employee and Manager objects
const employee = new Employee("Peter Parker", 12345, 2000.00);
const manager = new Manager("Barry Alen", 78910, 1500.00, "John
Smith", "New York");

//Display all the properties (employee)
document.getElementById("emp1").innerHTML = "Employee name: " +
employee.name;
document.getElementById("emp2").innerHTML = "Employee ID: " +
employee.id;
document.getElementById("emp3").innerHTML = "Employee salary: "
+ employee.salary;

//Display all the properties (manager)
document.getElementById("man1").innerHTML = "Employee name: " +
manager.name;
document.getElementById("man2").innerHTML = "Employee ID: " +
manager.id;
document.getElementById("man3").innerHTML = "Employee salary: "
+ manager.salary;
document.getElementById("man4").innerHTML = "Manager name: " +
manager.managerName;
document.getElementById("man5").innerHTML = "Branch: " +
manager.branch;
```

- Output



Task 4 : Event Handling

- Codes
    - task4.html

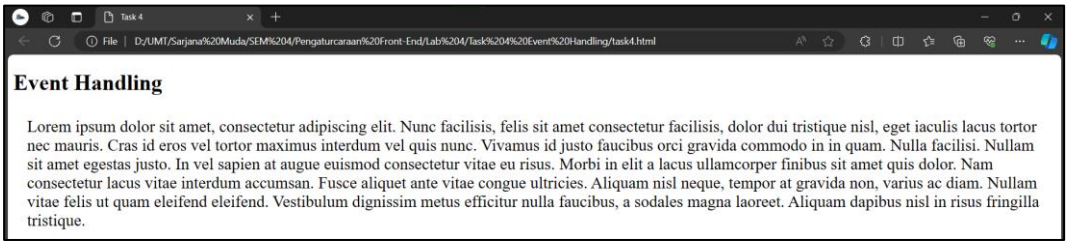
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Task 4</title>
    <style>
        p {
          padding: 10px;
          margin: 10px;
          cursor: pointer;
          font-size: 1.5rem;
        }

        #myText {
            padding: 5px;
            margin: 10px;
            border: 1px solid #ccc;
        }
    </style>
</head>
<body>

    <h1>Event Handling</h1>

    <p id="myParagraph">
        Lorem ipsum dolor sit amet, consectetur adipiscing elit.
        Nunc facilisis, felis sit amet consectetur facilisis,
        dolor dui tristique nisl, eget iaculis lacus tortor nec
mauris.
        Cras id eros vel tortor maximus interdum vel quis nunc.
        Vivamus id justo faucibus orci gravida commodo in in
quam.
        Nulla facilisi. Nullam sit amet egestas justo.
        In vel sapien at augue euismod consectetur vitae eu
risus.
        Morbi in elit a lacus ullamcorper finibus sit amet quis
dolor.
        Nam consectetur lacus vitae interdum accumsan.
        Fusce aliquet ante vitae congue ultricies.
        Aliquam nisl neque, tempor at gravida non, varius ac
diam.
        Nullam vitae felis ut quam eleifend eleifend.
        Vestibulum dignissim metus efficitur nulla faucibus,
        a sodales magna laoreet.
        Aliquam dapibus nisl in risus fringilla tristique.
    </p>

    <input type="text" id="myText" placeholder="Textfield">
```

```
    <script src="task4.js"></script>

</body>
</html>
```

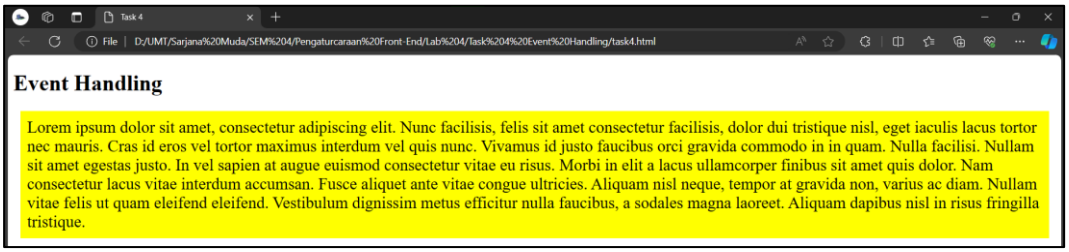
- task4.js

```
/***Mouse events***/
const paragraph = document.getElementById("myParagraph");

//Change background color to yellow when clicked
paragraph.addEventListener("click", event => {
    event.target.style.backgroundColor = "yellow"
});

//Change background color to blue when double-clicked
paragraph.addEventListener("dblclick", event => {
    event.target.style.backgroundColor = "blue"
});

//Change background color to red when mouse hovers over
paragraph.addEventListener("mouseover", event => {
    event.target.style.backgroundColor = "red"
});

//Change background color to green when mouse leaves
paragraph.addEventListener("mouseout", event => {
    event.target.style.backgroundColor = "green"
});



/***Textfield events***/
const textfield = document.getElementById("myText");

//Convert text to uppercase when its value changes
textfield.addEventListener("change", upperCase => {
    textfield.value = textfield.value.toUpperCase();
});

//Change border color to blue when textfield is focused
textfield.addEventListener("focus", event => {
    event.target.style.border = "1px solid #00f";
});

//Change border color to default when focus is removed from
textfield
textfield.addEventListener("blur", event => {
    event.target.style.border = "1px solid #ccc";
});
```

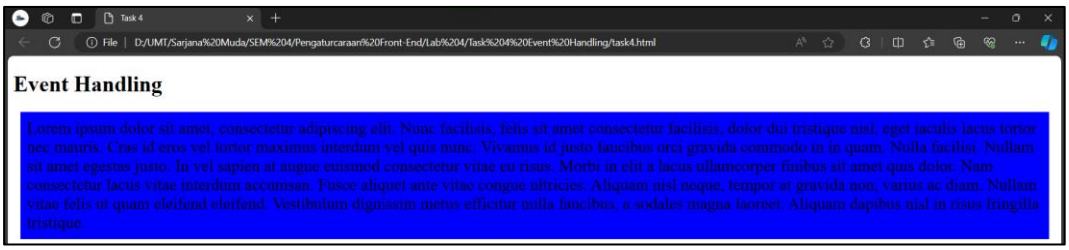
- Output (mouse events)
    - Default



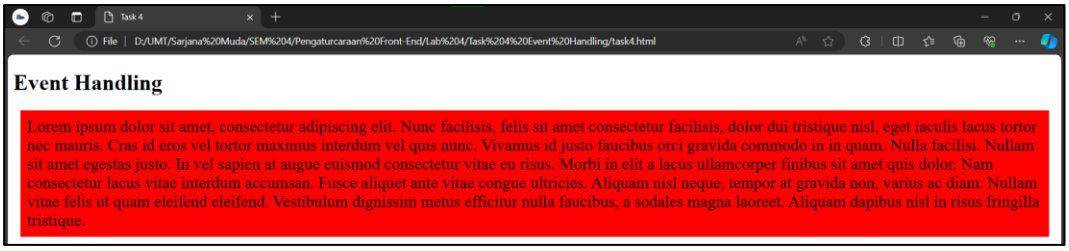
    - Onclick



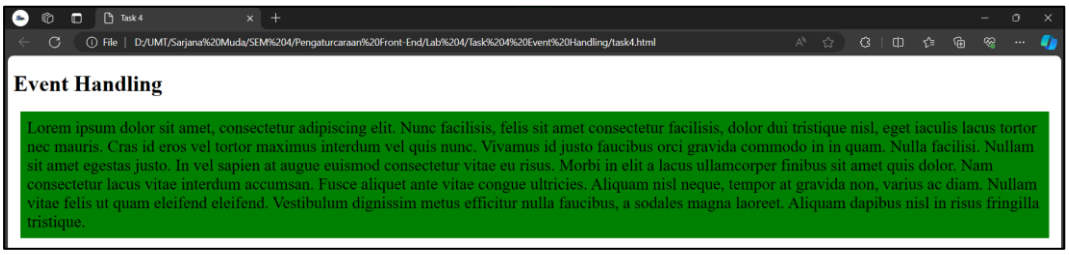
    - Ondblclick



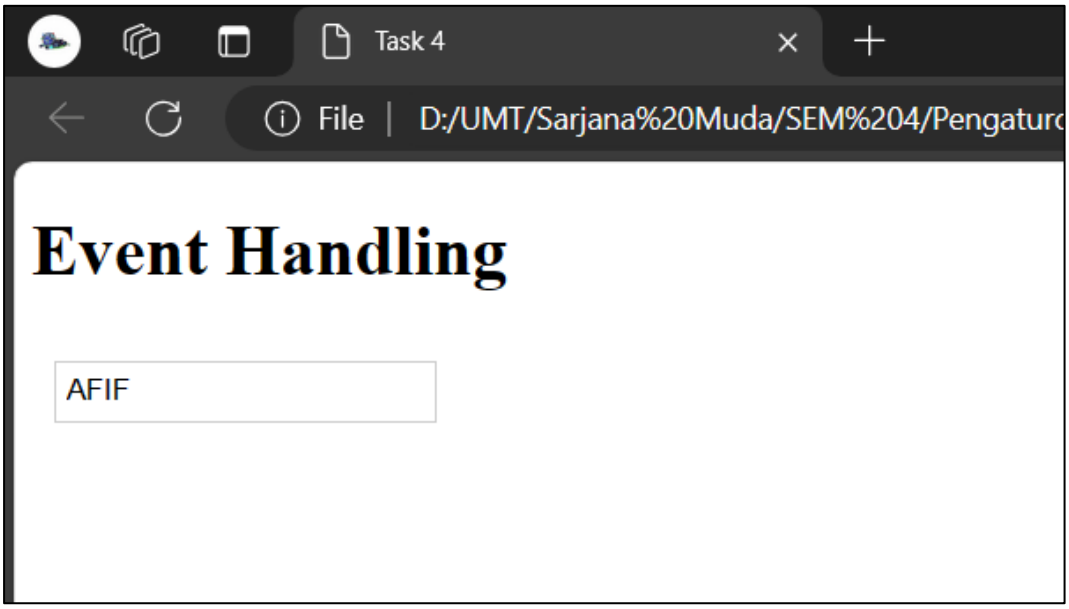
    - Onmouseover



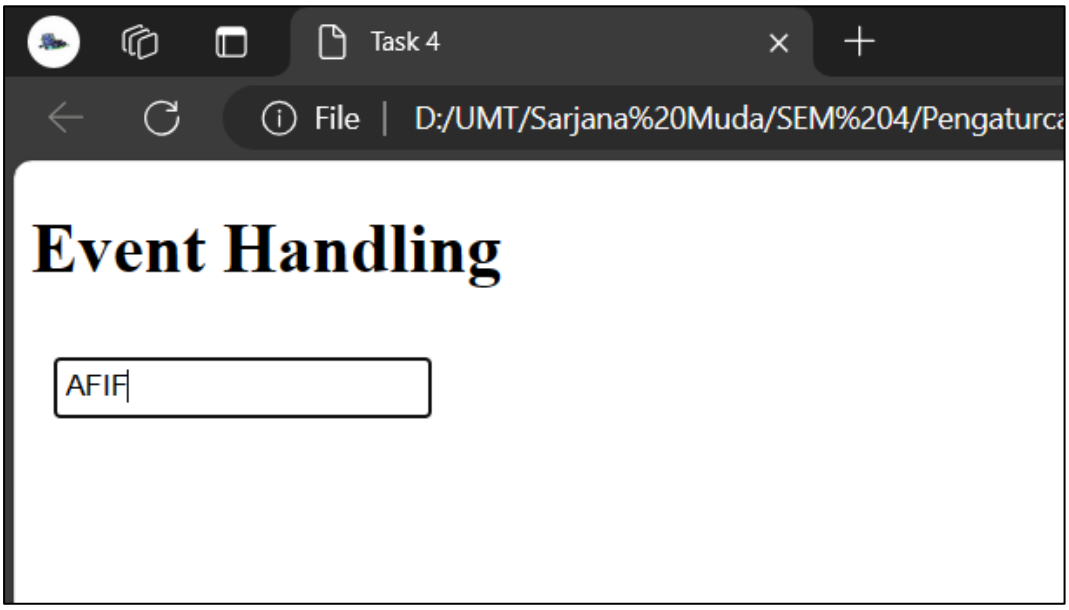
- Onmouseout



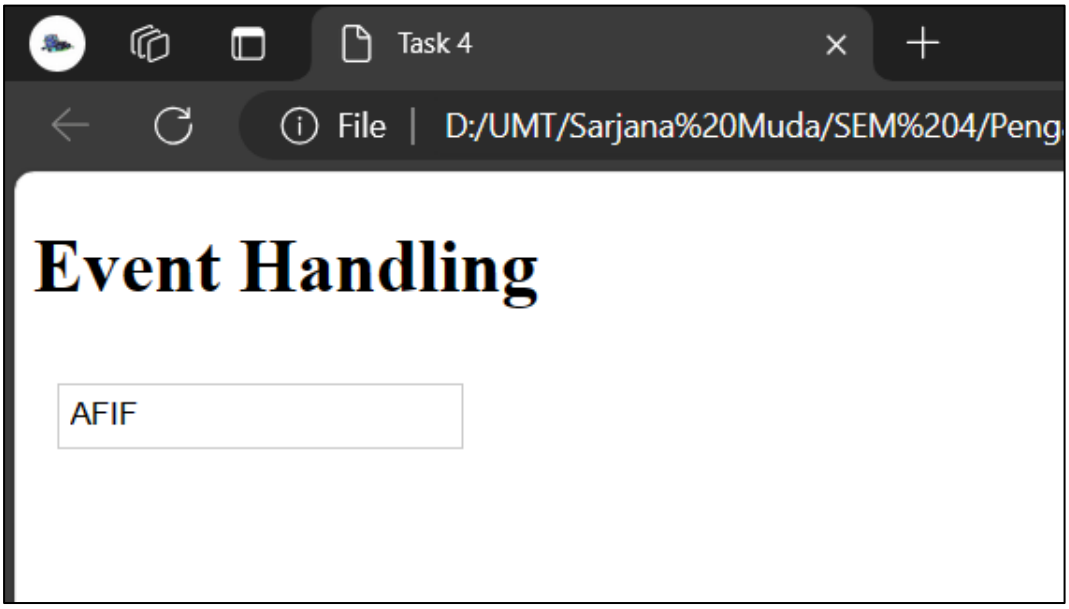
- Output (textfield events)
  - Onchange



- Onfocus



- Onblur



Task 5 : JavaScript with HTML Table

- Codes
    - task5.html

```
<!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>
    <style>
        table, th, td {
            border: 1px solid black;
        }

        table{
            border-collapse: collapse;
        }

        th, td {
            padding: 5px 10px;
        }

    </style>
</head>
<body>

    <h1>HTML Table with JavaScript</h1>

    <table id="myTable">
        <tbody>
            <tr>
                <td>1.</td>
                <td>Ahmad Faisal</td>
                <td>ahmadfaisal@gmail.com</td>
                <td>0199088888</td>
            </tr>
            <tr>
                <td>2.</td>
                <td>Ismail Sabri</td>
                <td>isabri@mail.com</td>
                <td>0199076760</td>
            </tr>
            <tr>
                <td>3.</td>
                <td>Fateh Yakin</td>
                <td>ffateh@hotmail.com</td>
                <td>0176067762</td>
            </tr>
        </tbody>
    </table>

    <script src="task5.js"></script>
```

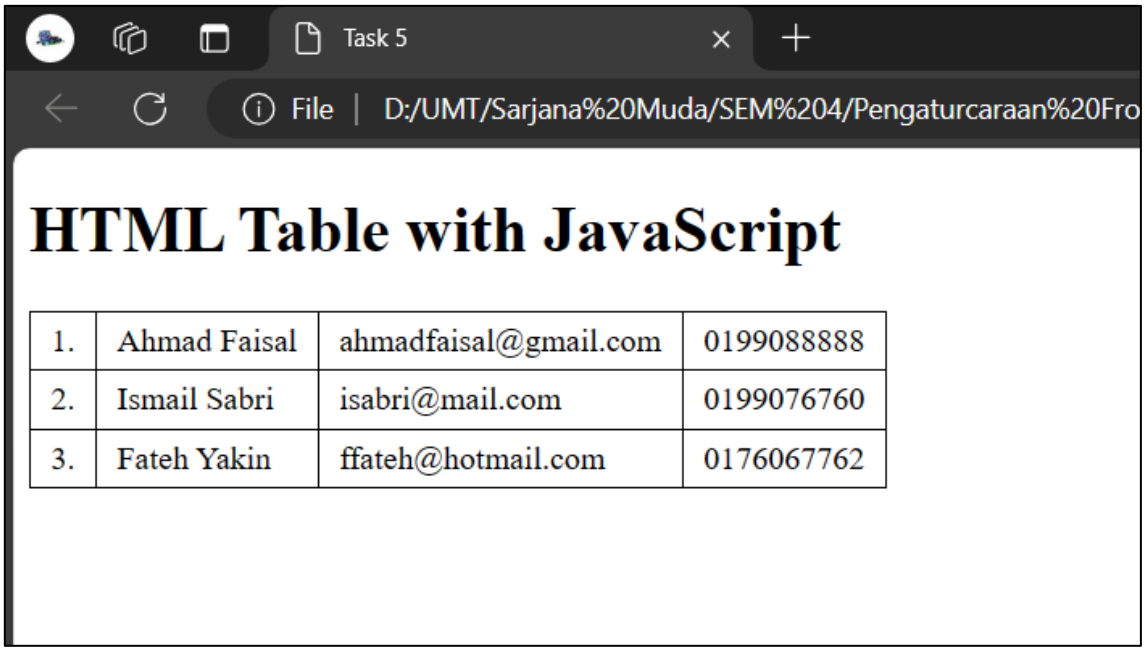
```
</body>
</html>
```

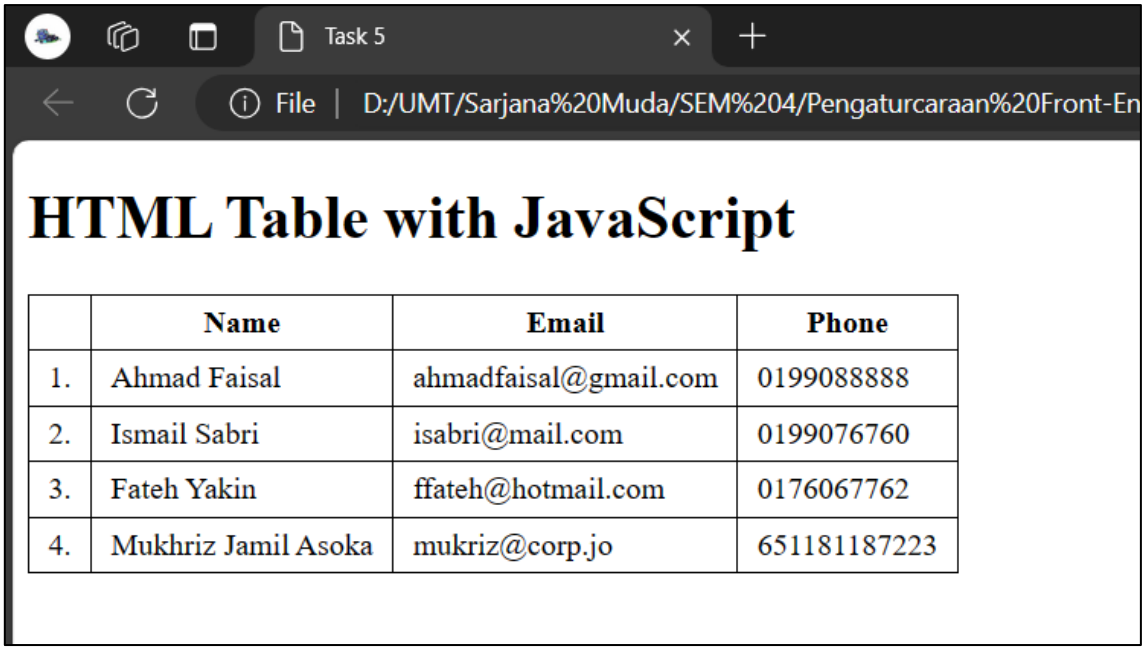
- task5.js

```
// Get the table element
const table = document.getElementById("myTable");

// Function to add a new record to the table
function addRecord(name, email, phone) {
    // Create a new row and cells for the record
    const newRow = document.createElement("tr");
    const cell1 = document.createElement("td");
    const cell2 = document.createElement("td");
    const cell3 = document.createElement("td");
    const cell4 = document.createElement("td");

    // Populate the cells with data
    cell1.textContent = table.rows.length + 1 + "."; //
Incremental number for each row
    cell2.textContent = name;
    cell3.textContent = email;
    cell4.textContent = phone;

    // Append cells to the new row
    newRow.appendChild(cell1);
    newRow.appendChild(cell2);
    newRow.appendChild(cell3);
    newRow.appendChild(cell4);

    // Append the new row to the table body
    table.getElementsByTagName("tbody")[0].appendChild(newRow);
}

// Function to add a header row to the table
function addTableHeader(headerData) {
    // Create the header row and cells
    const tableHeader = document.createElement("thead");
    const headerRow = document.createElement("tr");

    // Populate the header cells with data
    for (const headerText of headerData) {
        const headerCell = document.createElement("th");
        headerCell.textContent = headerText;
        headerRow.appendChild(headerCell);
    }

    // Append the header row to the table header
    tableHeader.appendChild(headerRow);

    // Insert the header before the table body
    const tableBody = table.getElementsByTagName("tbody")[0];
    table.insertBefore(tableHeader, tableBody);
}

// Add a sample record to the table
```

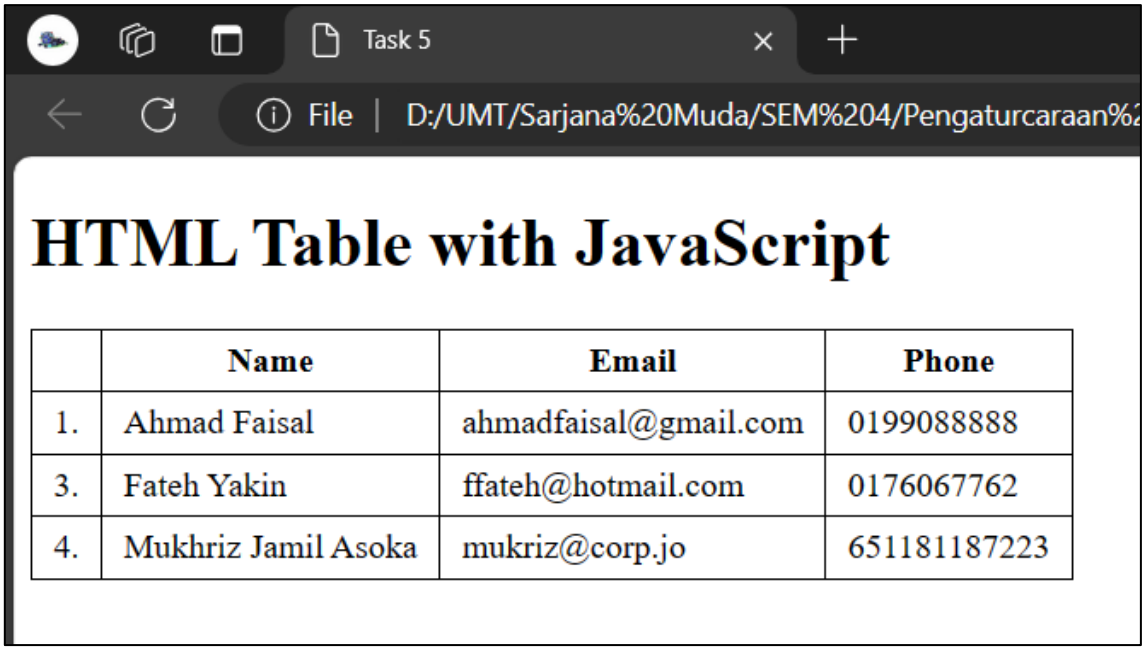
```
addRecord("Mukhriz Jamil Asoka", "mukriz@corp.jo",
"651181187223");

// Add a header to the table
addTableHeader(["", "Name", "Email", "Phone"]);

// Event listener for click events on table body
table.getElementsByTagName("tbody")[0].addEventListener("click",
event => {
    // Check if the clicked element is a table cell
    if (event.target.tagName === "TD") {
        // Remove the parent row if a cell is clicked
        event.target.parentNode.remove();
    }
});
```

- Output





	Name	Email	Phone
1.	Ahmad Faisal	ahmadfaisal@gmail.com	0199088888
2.	Ismail Sabri	isabri@mail.com	0199076760
3.	Fateh Yakin	ffateh@hotmail.com	0176067762
4.	Mukhriz Jamil Asoka	mukriz@corp.jo	651181187223



	Name	Email	Phone
1.	Ahmad Faisal	ahmadfaisal@gmail.com	0199088888
3.	Fateh Yakin	ffateh@hotmail.com	0176067762
4.	Mukhriz Jamil Asoka	mukriz@corp.jo	651181187223

Task 6 : JavaScript with HTML Table

- Codes
    - task6.html

```
<!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>
    <style>
        #big-square {
            width: 200px;
            height: 200px;
            border: 1px solid black;
            margin: 0 auto;
            position: relative;
        }

        .small-square {
            width: 50px;
            height: 50px;
            position: absolute;
            top: 0;
            left: 0;
        }

        #small-square1 {
            background-color: red;
            top: 50%;
            left: 10%;
        }

        #small-square2 {
            background-color: blue;
            top: 50%;
            left: 60%;
        }
    </style>
</head>
<body>

    <div id="big-square">
        <div class="small-square" id="small-
square1"></div>
        <div class="small-square" id="small-
square2"></div>
    </div>
    <br>
    <center>
        <button id="start-button">Start Animation</button>
        <button id="stop-button" disabled>Stop
Animation</button>
    </center>
```

```
        <script src="task6.js"></script>

    </body>
</html>
```

- task6.js

```
    const bigSquare = document.getElementById("big-square");
    const smallSquare1 = document.getElementById("small-
    square1");
    const smallSquare2 = document.getElementById("small-
    square2");
    const startButton = document.getElementById("start-
    button");
    const stopButton = document.getElementById("stop-button");

    let animationInterval; // Reference to the animation
    interval

    function getRandomPosition(max) {
        return Math.floor(Math.random() * max);
    }

    function moveSquares() {
        const bigSquareWidth = bigSquare.clientWidth -
    smallSquare1.clientWidth;
        const bigSquareHeight = bigSquare.clientHeight -
    smallSquare1.clientHeight;

        // Generate random positions within the boundaries of
    the big square
        const newTop1 = getRandomPosition(bigSquareHeight);
        const newLeft1 = getRandomPosition(bigSquareWidth);
        const newTop2 = getRandomPosition(bigSquareHeight);
        const newLeft2 = getRandomPosition(bigSquareWidth);

        // Update positions of the small squares
        smallSquare1.style.top = `${newTop1}px`;
        smallSquare1.style.left = `${newLeft1}px`;
        smallSquare2.style.top = `${newTop2}px`;
        smallSquare2.style.left = `${newLeft2}px`;
    }

    function startAnimation() {
        // Start animation by repeatedly calling moveSquares
    at a specific interval
        animationInterval = setInterval(moveSquares, 500); //
    Adjust interval for animation speed (50 milliseconds here)
        startButton.disabled = true;
        stopButton.disabled = false;
    }

    function stopAnimation() {
        clearInterval(animationInterval); // Clear the
    animation interval to stop movement
        startButton.disabled = false;
```

```
            stopButton.disabled = true;
        }

        startButton.addEventListener("click", startAnimation);
stopButton.addEventListener("click", stopAnimation);
```

- Output

Link GitHub: https://github.com/S65526AfifSyahmi/CSM3103-Lab-4