Practical no-3

TITLE: Simple Calculator Application:
Task: Develop a basic calculator application using Java for web Requirements: Create an HTML page with input fields for numbers and buttons for operations (addition, subtraction, multiplication, division). Use JavaScript to handle user interactions and perform calculations based on the input. Implement error handling to prevent invalid operations (e.g., division by zero) Display the result dynamically on the webpage

Objective

The objective of this lab is to create a basic calculator web application using **HTML**, **CSS**, and **JavaScript**. The application will allow the user to input two numbers and select a mathematical operation (addition, subtraction, multiplication, division). The result will be displayed dynamically on the webpage, and error handling will be implemented for invalid operations (e.g., division by zero).

Tools & Technologies

1. **HTML** - To create the structure of the webpage.
2. **CSS** - For styling the application.
3. **JavaScript** - To handle user input and perform calculations.
4. **Browser** - To view and test the web application.

Prerequisites

1. Basic understanding of HTML, CSS, and JavaScript.
2. A text editor (e.g., VS Code, Sublime Text) for writing the code.

3. A web browser (e.g., Chrome, Firefox) to view the result.

Steps

Step 1: Create the HTML Structure

Start by creating the basic structure of the calculator page using HTML. This will include input fields for the numbers, buttons for each operation, and a space to display the result.

HTML Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Simple Calculator</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="calculator-container">
        <h1>Simple Calculator</h1>
        <form name="calculator">
            <input type="number" name="num1" id="num1"
placeholder="Enter first number" required>
            <input type="number" name="num2" id="num2"
placeholder="Enter second number" required>
            <br><br>
            <button type="button"
onclick="performOperation('+')">+</button>
```

```
            <button type="button"
onclick="performOperation('-')">-</button>
            <button type="button"
onclick="performOperation('*')">*</button>
            <button type="button"
onclick="performOperation('/')">/</button>
            <br><br>
            <h3>Result: <span id="result">0</span></h3>
        </form>
    </div>

    <script src="script.js"></script>
</body>
</html>
```

Explanation:

- The form contains two input fields (`num1` and `num2`) for entering the numbers.
- The buttons represent the four operations (addition, subtraction, multiplication, division). Each button calls the `performOperation()` function with the respective operation symbol as an argument.
- The result will be displayed inside the `<span>` tag with the id `result`.

Step 2: Add CSS for Styling (Optional)

To make the calculator visually appealing, you can add some basic CSS to style the inputs, buttons, and layout.

CSS Code (styles.css):

```
body {
    font-family: Arial, sans-serif;
```

```
    background-color: #f4f4f4;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
}

.calculator-container {
    background-color: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    text-align: center;
}

input[type="number"] {
    width: 200px;
    padding: 10px;
    margin: 5px;
    border-radius: 5px;
    border: 1px solid #ccc;
}

button {
    padding: 10px 20px;
    margin: 10px;
    border: none;
    background-color: #4CAF50;
    color: white;
    border-radius: 5px;
    cursor: pointer;
```

```
}

button:hover {
    background-color: #45a049;
}

h1 {
    font-size: 24px;
    color: #333;
}

h3 {
    font-size: 20px;
    color: #333;
}
```

Explanation:

- The page layout is centered using Flexbox.
- The calculator container has a white background with padding and rounded corners to give it a neat look.
- Input fields and buttons are styled to make them larger and more user-friendly.

Step 3: JavaScript to Handle Calculations

Now, let's write the JavaScript code that will perform the calculations when the user interacts with the calculator. We will use the `performOperation()` function to handle each of the four operations.

JavaScript Code (script.js):

```
// Function to perform the selected operation
```

```
function performOperation(operator) {
    // Get input values from the form
    let num1 = parseFloat(document.calculator.num1.value);
    let num2 = parseFloat(document.calculator.num2.value);
    let result = 0;

    // Check if the inputs are valid numbers
    if (isNaN(num1) || isNaN(num2)) {
        document.getElementById('result').innerText =
"Please enter valid numbers!";
        return;
    }

    // Perform the operation based on the operator
    switch (operator) {
        case '+':
            result = num1 + num2;
            break;
        case '-':
            result = num1 - num2;
            break;
        case '*':
            result = num1 * num2;
            break;
        case '/':
            // Handle division by zero error
            if (num2 === 0) {
                document.getElementById('result').innerText
= "Error: Division by zero!";
                return;
            } else {
                result = num1 / num2;
```

```
            }
            break;
        default:
            document.getElementById('result').innerText =
"Invalid operation!";
            return;
    }

    // Display the result dynamically on the webpage
    document.getElementById('result').innerText = result;
}
```

Explanation:

- The `performOperation()` function is triggered when any of the operation buttons is clicked.
- It retrieves the values entered by the user (`num1` and `num2`) and checks if they are valid numbers using `isNaN()`.
- It then performs the selected operation (addition, subtraction, multiplication, or division) using a `switch` statement.
- If the user tries to divide by zero, an error message is displayed.
- The result is dynamically updated in the HTML using `document.getElementById('result').innerText`.

Step 4: Testing the Calculator

1. Save all the files (`index.html`, `styles.css`, and `script.js`).
2. Open the `index.html` file in your web browser.
3. Enter two numbers in the input fields and click any of the operation buttons.
4. The result should be displayed below the buttons.

Error Handling

- **Invalid input**: If the user enters non-numeric values, the result section will show "Please enter valid numbers!"
- **Division by zero**: If the user attempts to divide by zero, the result will display "Error: Division by zero!"

Conclusion

You have successfully created a simple calculator web application that performs basic arithmetic operations using HTML, CSS, and JavaScript. The application includes error handling for invalid operations like division by zero and displays the result dynamically on the webpage.

Further Improvements

- **Clear Button**: You can add a button to clear the inputs and result.
- **Decimal Precision**: You can format the result to a fixed number of decimal places.
- **Advanced Operations**: Add more operations like square roots, exponents, or percentage calculations.