Day - 3

JavaScript Functions

JavaScript functions are used to perform operations. We can call JavaScript functions many times to reuse the code.

Advantage of JavaScript function

Functions are useful in organizing the different parts of a script into the several tasks that must be completed. There are mainly two advantages of JavaScript functions.

1. Code reusability: We can call a function several times in a script to perform their tasks so it saves coding.
2. Less coding: It makes our program compact. We don't need to write many lines of code each time to perform a common task.

Rules for naming functions:

- It must be case sensitive.
- It must start with an alphabetical character (A-Z) or an underscore symbol.
- It cannot contain spaces.
- It cannot be used as a reserve word.

```
<script>
var add=new Function("num1","num2","return num1+num2");
document.writeln(add(2,5));
</script>
```

Closure

```
function greet(name) {
   function displayName() {
       console.log('Hi' + ' ' + name);
    }
    displayName();
}
greet('John');
```

JavaScript Closures

In JavaScript, closure provides access to the outer scope of a function from inside the inner function, even after the outer function has closed.

```
function calculate(x) {
   function multiply(y) {
      return x * y;
   }
   return multiply;
```

```
}


const multiply3 = calculate(3);

const multiply4 = calculate(4);

console.log(multiply3);

console.log(multiply3());

console.log(multiply3(6));

console.log(multiply4(2));
```

Task 1:- Build a Simple Arithmetic Calculator with JavaScript

Task: Build a Simple Arithmetic Calculator with JavaScript

Objective:
Create a simple arithmetic calculator that can perform addition, subtraction, multiplication, and division using JavaScript. The calculator should include functions, arrow functions, operators, and closures in its implementation.

Requirements:

1. **Arithmetic Operations:**
   - The calculator should perform basic arithmetic operations: addition, subtraction, multiplication, and division.
   - You must use JavaScript operators (`+`, `-`, `*`, `/`) for the calculations.
2. **Functions:**
   - Create separate functions for each arithmetic operation (addition, subtraction, multiplication, division).
   - Ensure that these functions accept two numbers as parameters and return the result.
3. **Arrow Functions:**
   - Use arrow functions to handle input validation. The validation function should check if the input is a valid number and prevent invalid operations.

4. **Closures:**
    - Implement a closure to store the result of the last calculation. The closure should allow users to add to or subtract from the last result.
    - The closure should provide a method to retrieve the last result.
5. **User Interface:**
    - Create a simple HTML interface for the calculator with buttons for digits (0-9), operators (+, -, *, /), and an equals (=) button.
    - The display should show the ongoing calculation, and the result should be shown after the equals button is clicked.
6. **Bonus:**
    - Implement a **clear** function to reset the calculator.

Deliverables:

- HTML file with the calculator layout.
- CSS file for styling the calculator.
- JavaScript file implementing the functions, arrow functions, and closures for the calculations.
- **Note:** Use an external script file.

Html file :-

```
<!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="style.css">



</head>
```

```
<body>


        <h1>Simple Calculator</h1>



    <div class="calculator-container">
        <div class="calculator">
        <input type="text" id="display" disabled>
        <div class="buttons">
            <button>7</button>
            <button>8</button>
            <button>9</button>
            <button>+</button>
            <button>4</button>
            <button>5</button>
            <button>6</button>
            <button>-</button>
            <button>1</button>
            <button>2</button>
            <button>3</button>
            <button>*</button>
            <button>0</button>
            <button>.</button>
            <button>=</button>
            <button>/</button>
```

```
            <button>C</button>
        </div></div>
    </div>
    <script src="script.js"></script>
</body>


</html>
```

Css file

```
body {
    display: flex;
    flex-direction: column; /* Stack elements vertically */
    justify-content: center; /* Center horizontally */
    align-items: center; /* Center vertically */
    height: 100vh;
    margin: 0;
    background-color: #c9c5c8;
    font-family: Arial, sans-serif;
}


h1 {
    text-align: center; /* Center the heading text */
    margin-bottom: 20px; /* Add some space below the heading */
}
```

```
.calculator-container { /* Style the container */
    display: flex;
    justify-content: center; /* Center the calculator */
    width: auto; /* Or a specific width if needed */
  }
.calculator {
    background-color: #0c0c0c;
    padding: 25px;
    border-radius: 20px;
    box-shadow: 0 0 10px rgba(26, 194, 76, 0.5);
}


#display {
    width: 100%;
    height: 80px;
    background-color: #3f3030;
    color: #ffffff;
    border: none;
    text-align: right;
    padding: 1px;
    margin-top:auto;
    font-size: 24px;
    border-radius: 5px;
    margin-bottom: 10px;
}
```

```
.buttons {
    display:grid;
    grid-template-columns: repeat(4, 1fr);
    gap: 10px;
}

button {
    background-color: #504141;
    color: #ffffff;
    border: none;
    padding: 20px;
    font-size: 18px;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s;
}

button:hover {
    background-color: #f7f2f2;
}

button:active {
    background-color: #ffecec;
}
```

Script js:-

```
// script.js

const createCalculator = () => {

    let lastResult = 0;



    const add = (a, b) => a + b;

    const subtract = (a, b) => a - b;

    const multiply = (a, b) => a * b;

    const divide = (a, b) => (b !== 0 ? a / b : "Error: Division by
zero");



    const validateInput = (input) => !isNaN(parseFloat(input)) && input
!== ""; // Parse float for decimals



    const updateLastResult = (result) => {

        lastResult = result;

    };



    const getLastResult = () => lastResult;



    return {

        add,

        subtract,

        multiply,

        divide,

        validateInput,
```

```
        updateLastResult,
        getLastResult,
    };
};

const calculator = createCalculator();

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

const appendToDisplay = (value) => {
    display.value += value;
};

const clearDisplay = () => {
    display.value = "";
};

const calculate = () => {
    const expression = display.value;

    try {
        // Use eval() for simpler calculation (with security considerations)
        const result = eval(expression);  // Be cautious using eval in production.
```

```
        if (calculator.validateInput(result)) { // Validate the
*result*
          display.value = result;
          calculator.updateLastResult(result);
        } else {
          display.value = "Error: Invalid Calculation";
        }

    } catch (error) {
        display.value = "Error: Invalid Expression";
        console.error("Calculation Error:", error); // Log the error
for debugging
    }
};



// Button event listeners (Improved)
const buttons = document.querySelectorAll('.buttons button'); // More
specific selector

buttons.forEach(button => {
    button.addEventListener('click', () => {
        const buttonValue = button.textContent;
        switch (buttonValue) {
            case '=':
                calculate();
```

```
                break;
            case 'C':
                clearDisplay();
                break;
            default:
                appendToDisplay(buttonValue);
        }
    });
});

function print(x){

}

const print1 = (x) => {

}
```

Output