INTERNSHIP REPORT

WEEK 4 DAY 5

Submitted to:	**Ali Hyder**	Submission Date:	**18th July, 2025**
Internship Domain:	**Front Development**	Internship Name:	**ProSensia**
Student Name:	**Yasal Qamar**	Roll No.	**S25031**

JavaScript Functions (Regular, Arrow), Scope

Objective

To understand how JavaScript functions work, the difference between regular and arrow functions, and how variable scope impacts program behavior.

Topics Covered

Part 1: JavaScript Functions

Functions are reusable blocks of code designed to perform a specific task.

Regular Function (Function Declaration)

```
function greet(name) {

  return `Hello, ${name}!`;

}

console.log(greet("Yasal"));  // Output: Hello, Yasal!
```

Function Expression

```
const add = function(a, b) {

  return a + b;

};

console.log(add(2, 3));  // Output:
```

Part 2: Arrow Functions

Arrow functions are a shorter syntax introduced in ES6.

```
const greet = (name) => {
  return `Hello, ${name}!`;
};
console.log(greet("Qamar"));  // Output: Hello, Qamar!
```

Shorter Arrow Syntax

```
const square = x => x * x;
console.log(square(4));  // Output: 16
```

Part 3: Scope in JavaScript

The scope defines where variables can be accessed.

Type of Scope	Description
Global	Declared outside any function, accessible anywhere
Local	Declared inside a function, accessible only inside that function
Block	Declared inside {} with let or const, limited to that block

Example:

```
let globalVar = "I'm global";
function showScope() {
  let localVar = "I'm local";
  console.log(globalVar); // ✅
  console.log(localVar);  // ✅
}
showScope();
```

```
console.log(globalVar); // ✅

console.log(localVar);  // ❌ Error: localVar is not defined
```

Learning Outcome

1. Understood what functions are and how to use them.
2. Write reusable regular functions using the `function` keyword.
3. Used arrow functions for concise and modern syntax.
4. Differentiated between regular and arrow functions in behavior and style.
5. Learned about different scopes: global, local, and block.
6. Practiced scoping rules and avoided common variable errors.

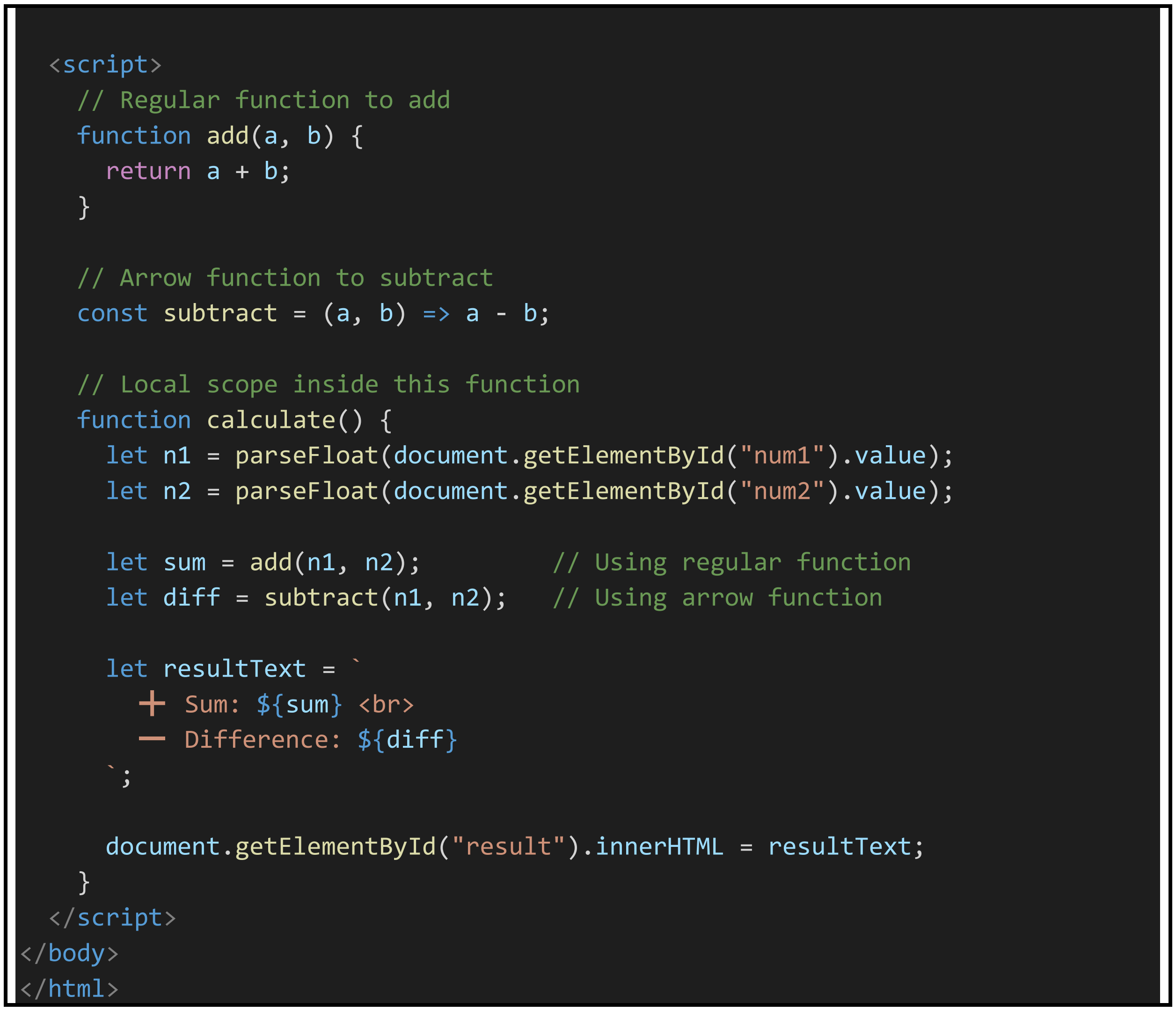
CODING

```html
<!DOCTYPE html>
<html>
<head>
  <title>Simple Calculator</title>
  <style>
    body {
      font-family: Arial;
      padding: 20px;
      max-width: 400px;
      margin: auto;
      background-color: #f0f0f0;
    }
    input, button {
      padding: 10px;
      margin: 5px 0;
      width: 100%;
    }
  </style>
</head>
<body>
  <h2>Simple Calculator</h2>

  <input type="number" id="num1" placeholder="Enter first number">
  <input type="number" id="num2" placeholder="Enter second number">

  <button onclick="calculate()">Calculate</button>

  <h3 id="result">Result will appear here</h3>
```



```
  <script>
    // Regular function to add
    function add(a, b) {
      return a + b;
    }

    // Arrow function to subtract
    const subtract = (a, b) => a - b;

    // Local scope inside this function
    function calculate() {
      let n1 = parseFloat(document.getElementById("num1").value);
      let n2 = parseFloat(document.getElementById("num2").value);

      let sum = add(n1, n2);          // Using regular function
      let diff = subtract(n1, n2);    // Using arrow function

      let resultText = `
        ➕ Sum: ${sum} <br>
        ➖ Difference: ${diff}
      `;

      document.getElementById("result").innerHTML = resultText;
    }
  </script>
</body>
</html>
```

```
// Regular function to add
function add(a, b) {
  return a + b;
}
```

- This is a **regular function** named add.
- It takes **two parameters** a and b.
- It returns their **sum** (a + b).
- Example: add(2, 3) → returns 5

```
// Arrow function to subtract
const subtract = (a, b) => a - b;
```

- This is an **arrow function**, assigned to a const variable called subtract.

- It also takes **two numbers**, and returns a - b (subtraction).
- Arrow functions are a shorter way to write functions in JavaScript.

```
function calculate() {
  let n1 = parseFloat(document.getElementById("num1").value);
  let n2 = parseFloat(document.getElementById("num2").value);
```

- calculate() is a function that runs **when the button is clicked**.
- document.getElementById("num1").value gets the **value entered in the input field** with id num1.
- parseFloat() converts that string value into a **number**.
- Same for num2.

```
  let sum = add(n1, n2);        // Using regular function
  let diff = subtract(n1, n2);  // Using arrow function
```

- Calls the add() function using the entered numbers → stores result in sum.
- Calls the subtract() arrow function → stores result in diff.

```
  let resultText = `
    ➕ Sum: ${sum} <br>
    ➖ Difference: ${diff}
  `;
```

- This is a **template string** (with backticks `).
- It shows the results with some emojis and line breaks (<br>).

```
  document.getElementById("result").innerHTML = resultText;}
```

- Sets the content of the <h3 id="result"> element to the result text.

Concept	**Used In Code**	**Meaning**
Regular Function	add(a, b)	Traditional way to declare functions
Arrow Function	const subtract = (a, b) => a - b;	Short, modern function syntax
DOM Manipulation	document.getElementById(...)	Accessing and changing HTML from JS
Scope	n1, n2, sum, diff	Variables inside calculate() are local

CONCUSLION:

I explored the core of JavaScript programming **functions and scope**. Functions allowed me to structure my code into reusable blocks, improving readability and efficiency. I compared **regular functions** with modern **arrow functions**, learning their syntax and use cases. Understanding **scope** helped me control where variables exist and how they affect each other. This knowledge is essential for writing clean, bug-free, and modular JavaScript applications.