Lesson 5: Functions in JavaScript

Objective

By the end of this lesson, students will understand how to create and use functions in JavaScript, including function parameters, return values, and different ways to define functions.

1. Introduction to Functions

Functions are reusable blocks of code designed to perform a specific task. They help make code more modular, readable, and maintainable.

- **What is a Function?**
  - A function is a set of instructions that perform a specific task when called.
  - Functions help avoid code repetition by allowing you to write code once and reuse it multiple times.
- **Why Use Functions?**
  - **Code Reusability**: Write once, use multiple times.
  - **Modularity**: Break complex problems into smaller, manageable functions.
  - **Abstraction**: Hide the implementation details and expose only what is necessary.

2. Defining and Calling Functions

- **Function Declaration**
  - A function can be declared using the `function` keyword.

```
function greet() {
    console.log("Hello, World!");
}
```

  - **Calling a Function**
    - To execute the code inside the function, you need to call it.

```
greet();  // Output: Hello, World!
```

- **Function Parameters and Arguments**
  - **Parameters**: Variables listed as part of the function's definition.
  - **Arguments**: Values passed to the function when it is called.

```
function greet(name) {
    console.log("Hello, " + name + "!");
}

greet("Alice");  // Output: Hello, Alice!
```

- **Returning Values**

- A function can return a value using the `return` statement.

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

let sum = add(5, 10);  // sum is 15
console.log(sum);  // Output: 15
```

3. Function Expressions and Arrow Functions

- **Function Expression**
  - A function expression is when a function is stored in a variable.

```
const greet = function(name) {
    console.log("Hello, " + name + "!");
};

greet("Bob");  // Output: Hello, Bob!
```

- **Arrow Functions**
  - Arrow functions provide a shorter syntax for writing function expressions.

```
const add = (a, b) => {
    return a + b;
};

let sum = add(2, 3);
console.log(sum);  // Output: 5
```

  - **Implicit Return in Arrow Functions**
    - When the function body contains only a single expression, the `return` keyword can be omitted.

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

console.log(multiply(2, 4));  // Output: 8
```

4. Function Scope and Hoisting

- **Scope**
  - **Global Scope**: Variables declared outside any function are in the global scope and can be accessed anywhere.
  - **Local Scope**: Variables declared inside a function are in the local scope and can only be accessed within that function.

```
let globalVar = "I am global";

function testScope() {
    let localVar = "I am local";
    console.log(globalVar);  // Accessible
    console.log(localVar);   // Accessible
}

testScope();
console.log(globalVar);  // Accessible
console.log(localVar);   // Error: localVar is not defined
```

- **Hoisting**
  - **Function Hoisting**: Function declarations are hoisted to the top of their scope, meaning you can call a function before it is defined.

```
console.log(add(2, 3));  // Output: 5

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

  - **Variable Hoisting**: Variables declared with `var` are hoisted, but their values are not initialized. `let` and `const` are not hoisted.

```
console.log(x);  // Output: undefined
var x = 5;

console.log(y);  // Output: ReferenceError: y is not defined
let y = 10;
```

5. Hands-On Practice

Exercise 1: Simple Calculator

- Write a function that takes two numbers and a mathematical operator (`+` , `-` , `*` , `/`) and returns the result.

```
function calculator(a, b, operator) {
    if (operator === '+') {
        return a + b;
    } else if (operator === '-') {
        return a - b;
    } else if (operator === '*') {
        return a * b;
    } else if (operator === '/') {
        return a / b;
    } else {
        return "Invalid operator";
    }
}

console.log(calculator(10, 5, '+'));  // Output: 15
```

Exercise 2: Greeting Function

- Write a function that takes a person's name and age and returns a greeting message.

```
function greetPerson(name, age) {
    return `Hello, ${name}! You are ${age} years old.`;
}

console.log(greetPerson("Alice", 30));  // Output: Hello, Alice! You are 30
years old.
```

Exercise 3: Array Sum Function

- Write a function that takes an array of numbers and returns the sum of all the numbers.

```
function sumArray(numbers) {
    let sum = 0;
    for (let number of numbers) {
        sum += number;
    }
    return sum;
}

let numbers = [1, 2, 3, 4, 5];
console.log(sumArray(numbers));  // Output: 15
```

6. Homework/Assignment

Assignment 1: Maximum of Three Numbers

- Write a function that takes three numbers and returns the maximum of the three.

Assignment 2: Factorial Function

- Write a function that takes a number and returns its factorial. The factorial of a number is the product of all positive integers less than or equal to that number.

Assignment 3: Palindrome Checker

- Write a function that checks if a given string is a palindrome (a word that reads the same backward as forward).

7. Recommended Resources

Documentation and Tutorials

- MDN Web Docs: Functions
  - A detailed guide on functions, including how to define and call functions, and different types of functions in JavaScript.
- JavaScript.info: Functions
  - A comprehensive resource on functions in JavaScript, covering everything from basic syntax to advanced concepts.

Video Tutorials

- Programming with Mosh: JavaScript Functions
  - A video tutorial that explains functions in JavaScript with practical examples.
- Traversy Media: JavaScript Functions Tutorial
  - A thorough video guide on understanding and using functions in JavaScript, with clear examples and explanations.