Lesson 2: Control Structures and Functions in JavaScript

Objective

In this lesson, you'll learn how to use control structures to make decisions and repeat actions in your JavaScript code. You'll also be introduced to functions, which allow you to write reusable blocks of code. By the end of this lesson, you'll be able to control the flow of your code and create functions.

1. Control Structures

What are Control Structures?

- Control structures are constructs in JavaScript that allow you to dictate how and when certain blocks of code are executed. They are crucial for creating interactive and responsive web applications.

1.1 Conditional Statements

If Statement

- The `if` statement lets you execute a block of code only if a specified condition is true.

```
let age = 18;
if (age >= 18) {
  console.log("You are an adult.");
}
```

If-Else Statement

- The `if-else` statement provides an alternative block of code that executes if the condition is false.

```
let age = 16;
if (age >= 18) {
  console.log("You are an adult.");
} else {
  console.log("You are a minor.");
}
```

Else If Statement

- The `else if` statement allows you to test multiple conditions sequentially.

```
let age = 20;
if (age < 13) {
  console.log("You are a child.");
} else if (age < 18) {
  console.log("You are a teenager.");
} else {
  console.log("You are an adult.");
}
```

Switch Statement

- The `switch` statement is an alternative to using multiple `if-else` statements. It allows you to perform different actions based on different conditions.

```
let day = "Monday";
switch (day) {
  case "Monday":
    console.log("Start of the week!");
    break;
  case "Wednesday":
    console.log("Midweek");
    break;
  case "Friday":
    console.log("Weekend is near!");
    break;
  default:
    console.log("Just another day.");
}
```

1.2 Loops

For Loop

- The `for` loop is used to execute a block of code a certain number of times.

```
for (let i = 0; i < 5; i++) {
  console.log("Iteration number " + i);
}
```

While Loop

- The `while` loop continues to execute a block of code as long as a specified condition is true.

```
let i = 0;
while (i < 5) {
  console.log("Iteration number " + i);
  i++;
}
```

Do-While Loop

- The `do-while` loop is similar to the `while` loop, but it guarantees that the block of code runs at least once, even if the condition is false initially.

```
let i = 0;
do {
  console.log("Iteration number " + i);
  i++;
} while (i < 5);
```

2. Functions

What are Functions?

- Functions are reusable blocks of code that perform a specific task. They help you avoid repeating code by allowing you to call the same block of code from multiple places in your program.

2.1 Defining Functions

Function Declaration

- You can define a function using the `function` keyword followed by a name, parameters (optional), and a block of code.

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

Function Expression

- Functions can also be defined as expressions and assigned to variables.

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

Arrow Functions

- Arrow functions provide a concise syntax for writing functions.

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

2.2 Calling Functions

Calling a Function

- Once a function is defined, you can call it by its name followed by parentheses. If the function expects arguments, pass them inside the parentheses.

```
function add(a, b) {
  return a + b;
}
let result = add(3, 4);  // result is 7
console.log(result);  // Output: 7
```

Function Parameters and Arguments

- Functions can take parameters, which are variables that the function expects. When calling the function, you provide arguments, which are the actual values passed to the function.

```
function multiply(x, y) {
  return x * y;
}
console.log(multiply(5, 6));  // Output: 30
```

Return Statement

- The `return` statement is used to return a value from a function. If a function doesn't have a `return` statement, it returns `undefined` by default.

```
function square(num) {
  return num * num;
}
let squaredNumber = square(5);  // squaredNumber is 25
console.log(squaredNumber);  // Output: 25
```

3. Hands-On Practice

Exercise 1: Conditional Statements

- Write a function that takes a number as input and returns whether the number is positive, negative, or zero.

Exercise 2: Loops

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

Exercise 3: Functions

- Create a function that converts Celsius to Fahrenheit. Test your function by calling it with different values.

4. Homework/Assignment

- **Assignment 1: Write a Function to Calculate Factorial**
  - Write a function that takes a number as input and returns the factorial of that number. Use a `for` loop within the function to calculate the factorial.
- **Assignment 2: Simple Calculator**
  - Create a simple calculator function that takes two numbers and an operator (`+` , `-` , `*` , `/`) and returns the result. Use `if-else` or `switch` statements to handle different operations.

5. Recommended Resources

Documentation and Tutorials

- MDN Web Docs: Control Flow and Error Handling
  - This section of the MDN Web Docs provides a comprehensive overview of control structures in JavaScript, including conditionals and loops.

- JavaScript.info: Functions
    - A detailed guide on JavaScript functions, from basic concepts to advanced topics, with examples and exercises.
- W3Schools: JavaScript Functions
    - An easy-to-understand tutorial on JavaScript functions, covering different ways to define and call functions.

Video Tutorials

- freeCodeCamp: JavaScript Functions
    - A beginner-friendly video that explains JavaScript functions and how to use them in your code.
- The Net Ninja: JavaScript Control Flow
    - A YouTube video that covers control flow in JavaScript, including conditionals and loops.

Books

- **"Eloquent JavaScript" by Marijn Haverbeke**
    - Chapters 2 and 3 focus on control flow and functions, providing both explanations and exercises.

Practice Platforms

- Codecademy: Learn JavaScript - Control Flow
    - A hands-on module that focuses on control flow in JavaScript, complete with interactive exercises.
- HackerRank: JavaScript Practice
    - Offers challenges on various JavaScript topics, including control structures and functions, to test your understanding.