JavaScript: Functions & Scope

AN INTRODUCTION TO JAVASCRIPT FUNCTIONS AND SCOPE

What are Functions?

- • Functions are reusable blocks of code that perform a task.
- • Functions help in modularizing code and improving maintainability.
- Example:

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

Function Declaration vs Function Expression

- 1. Function Declaration: Named functions that can be hoisted.
- Example:
- `function add(a, b) { return a + b; }`
- 2. Function Expression: Assigned to a variable, not hoisted.
- Example:
- `const multiply = function(a, b) { return a * b; };`

Arrow Functions (ES6)

- • Introduced in ES6 for concise syntax.
- • Automatically binds `this`.
- Example:
- const square = (x) => x * x;
- console.log(square(5)); // Output: 25

Function Parameters and Default Values

- • Functions can take parameters and return values.
- • Default values can be set for parameters.
- Example:

```
  function greet(name = 'Guest') {
      return 'Hello, ' + name;
  }
  console.log(greet()); // Output: Hello, Guest
```

Understanding Scope in JavaScript

- • Scope determines where variables can be accessed.
- • JavaScript has Global Scope, Function Scope, and Block Scope.
- Example:

```
  let globalVar = 'I am global';
  function testScope() {
      let localVar = 'I am local';
      console.log(globalVar); // Accessible
      console.log(localVar);  // Accessible
  }
  console.log(localVar); // Error: localVar is not defined
```

Closures in JavaScript

- • A function that remembers the scope where it was created.
- Example:

```
  function outer() {
      let count = 0;
      return function() { count++; return count; };
  }
  const counter = outer();
  console.log(counter()); // Output: 1
```

Common Mistakes

- • Forgetting to use `return` in a function.
- • Misusing `this` inside functions.
- • Not understanding the difference between `var`, `let`, and `const`.

Best Practices

- • Use `const` for functions that should not be redefined.
- • Prefer arrow functions for short, single-line functions.
- • Avoid using global variables.

Summary & Key Takeaways

- • Functions make code reusable and modular.
- • JavaScript supports function declarations, expressions, and arrow functions.
- • Understanding scope is essential to avoid variable conflicts.