JavaScript Functions Review

JavaScript Functions

- Functions are reusable blocks of code that perform a specific task.
- Functions can be defined using the function keyword followed by a name, a list of parameters, and a block of code that performs the task.

Example Code

```
function addNumbers(x, y, z) {
  return x + y + z;
}


console.log(addNumbers(5, 3, 8)); // Output: 16
```

- Arguments are values passed to a function when it is called.
- A function call is the process of executing a function in a program by specifying the function's name followed by parentheses, optionally including arguments inside the parentheses.
- When a function finishes its execution, it will always return a value.
- By default, the return value of a function is undefined.
- The return keyword is used to specify the value to be returned from the function and ends the function execution.
- Default parameters allow functions to have predefined values that will be used if an argument is not provided when the function is called. This makes functions more flexible and prevents errors in cases where certain arguments might be omitted.

Example Code

```
const calculateTotal = (amount, taxRate = 0.05) => {
  return amount + (amount * taxRate);
};
```

```
console.log(calculateTotal(100)); // Output: 105
```

- Function Expressions are functions that you assign to variables. By doing this, you can use the function in any part of your code where the variable is accessible.

Example Code

```
const multiplyNumbers = function(firstNumber, secondNumber) {
 return firstNumber * secondNumber;
};

console.log(multiplyNumbers(4, 5)); // Output: 20
```

Arrow Functions

- Arrow functions are a more concise way to write functions in JavaScript.

Example Code

```
const calculateArea = (length, width) => {
 const area = length * width;
 return `The area of the rectangle is ${area} square units.`;
};

console.log(calculateArea(5, 10)); // Output: "The area of the rectangle is 50 square units."
```

- When defining an arrow function, you do not need the function keyword.
- If you are using a single parameter, you can omit the parentheses around the parameter list.

Example Code

```
const cube = x => {
 return x * x * x;
};

console.log(cube(3)); // Output: 27
```

- If the function body consists of a single expression, you can omit the curly braces and the return keyword.

Example Code

```
const square = number => number * number;


console.log(square(5)); // Output: 25
```

Scope in Programming

- **Global scope**: This is the outermost scope in JavaScript. Variables declared in the global scope are accessible from anywhere in the code and are called global variables.
- **Local scope**: This refers to variables declared within a function. These variables are only accessible within the function where they are declared and are called local variables.
- **Block scope**: A block is a set of statements enclosed in curly braces {} such as in if statements, or loops.
- Block scoping with let and const provides even finer control over variable accessibility, helping to prevent errors and make your code more predictable.