JavaScript Function

Anthoniraj Amalanathan

1. **Function**

   In JavaScript, a function is a block of reusable code that performs a specific task. Functions can take parameters as input, perform operations, and return a value.

```javascript
// Define a function named 'add' that takes two parameters 'a' and 'b' and returns their sum
function add(a, b) {
    return a + b;
}

// Call the 'add' function
console.log(add(3, 5)); // Output: 8
```

2. **Anonymous Function**

   Anonymous functions in JavaScript are functions that are defined without a name.

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

// Define an anonymous function that takes two parameters 'a' and 'b' and returns their sum
const add = function(a, b) {
    return a + b;
};

// Call the anonymous function
console.log(add(3, 5)); // Output: 8
```

3. **Arrow Function**

   Arrow functions are a concise way to write anonymous functions in JavaScript. They have a shorter syntax compared to traditional function expressions.

   ```
   // Anonymous function
   const add = function(a, b) {
       return a + b;
   };

   // Arrow function to double a number
   const add = (a, b) => (a+b);

   // Call the 'add' arrow function
   console.log(add(4, 3)); // Output: 7
   ```

4. **Callback Function**

A callback function is a function passed as an argument to another function, which is then invoked inside the outer function to complete some action.

```
// Function that takes a callback to perform an operation
function greet(name, callback) {
   callback(name);
}

// Callback function to format a greeting message
function formatGreeting(name) {
    console.log(`Hello, ${name}!`);
}

// Call the 'greet' function with the 'formatGreeting' callback
greet("Anthoniraj", formatGreeting); // Output: Hello, Anthoniraj!
```

5. **Higher-Order Function (HOF)**

   A higher-order function is a function that takes one or more functions as arguments or returns a function as its result.

```javascript
// Higher-order function that takes a function and a value, and applies the function to the value
function applyOperation(operation, value) {
    return operation(value);
}

// Function to square a number
const square = (num) => num * num;

// Call the 'applyOperation' higher-order function with the 'square' function
console.log(applyOperation(square, 3)); // Output: 9
```

6. **Constructor function**

Constructor functions are often used to set up prototype chains for objects. Properties and methods defined on the constructor's prototype are shared among all instances created with that constructor.

```
function Person(name, age) {
    this.name = name;
    this.age = age;
}

Person.prototype.greet = function() {
    return `Hello, my name is ${this.name} and I am ${this.age} years old.`;
}

const kumar = new Person('Kumar', 30); // Create an object
console.log(kumar.greet()); // Outputs: "Hello, my name is Kumar and I am 30 years old."
```

7. **ES6 Class**

   ES6, or ECMAScript 2015, is the sixth version of the JavaScript specification, which is managed by the European Computer Manufacturers Association (ECMA).

```
class Person {
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }

    greet() {
        return `Hello, my name is ${this.name} and I am ${this.age} years old.`;
    }
}

const kumar = new Person('Kumar', 30); // Create an object
console.log(kumar.greet()); // Outputs: "Hello, my name is Kumar and I am 30 years old."
```