JavaScript Functions

By Mohini Bharambe

Functions

- JavaScript functions are blocks of code that can be defined and executed whenever needed.
- They are a crucial part of JavaScript programming and are used to perform specific tasks or actions.
- **Why use functions?**
  - Code reusability
  - Improved readability
  - Easier debugging and testing
- Functions are often referred to as "first-class citizens" in JavaScript because they can be treated like any other value, such as a number or a string. This means that they can be assigned to variables, passed as arguments to other functions, and returned as values from functions.

- Here's the basic syntax for defining a function in JavaScript

```
function functionName(parameters) {
  // code to be executed
}
```

- The functionName is a unique identifier for the function, and the parameters are the variables that are passed to the function when it is called.
- These parameters act as placeholders for the actual values that are passed to the function when it is executed.
- **Declaring:**

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

- **Calling:**

```
greet(); // Outputs: Hello, world!
```

- Functions can also have multiple parameters, like this:

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

- In this case, the **add** function takes two parameters, x and y, and returns their sum.
- **Parameters:**
  - Variables listed in the function definition
  - **Example:** function add(a, b) { return a + b; }
- **Arguments:**
  - Values passed during the function call
  - **Example:** add(2, 3); // Returns 5

- **Default Parameters**
  - **Purpose:** Assign default values to parameters.
  - **Example:**

```
function multiply(a, b = 1) {
    return a * b;
}
console.log(multiply(5)); // Outputs: 5
```

- **Return Statement**
  - **Purpose:** Ends the function execution and specifies a value to return.
  - **Example:**

```
function square(num) {
    return num * num;
}
console.log(square(4)); // Outputs: 16
```

Arrow Function

- JavaScript also has a special type of function called an "arrow function," which uses a shorter syntax. Here's the same square a function defined using an arrow function:

```
const square = (x) => {
  return x * x;
};
```

- Arrow functions are often used when you want to create a small, one-line function that doesn't require a separate function keyword.

- Functions can be defined inside other functions, which is known as "nesting." This is useful for creating smaller, reusable blocks of code that can be called from within the larger function.

```
function outerFunction(x) {
  function innerFunction() {
    // code to be executed
  }
  // more code
}
```

- In this example, the innerFunction is defined inside the outerFunction and can only be called from within that function.

THANK YOU