Hema Coding School

YouTube Link: https://www.youtube.com/@HemaCodingSchool

Function

How function overcome repeated logic (avoiding code duplication)?

```
var num1 = 5;
var num2 = 10;
var sum = num1 + num2;
console.log("Sum:", sum); // Output: Sum: 15
var num3 = 8;
var num4 = 3;
var sum2 = num3 + num4;
console.log("Sum:", sum2); // Output: Sum: 11
// Function to calculate the sum of two numbers
function calculateSum(a, b) {
  console.log(a + b);
  // return a + b;
}
// Calculate the sum using the function
calculateSum(5, 10);
calculateSum(8, 3);
```

Functions are fundamental building blocks in programming and are used to organize code, improve readability, and promote code reusability.

1. **Input (Parameters):** Functions can take input values, known as parameters or arguments, which are variables or values passed to the function when it is called. These inputs allow functions to work with different data.
2. **Logic (Operations):** Inside a function, you write the logic or instructions that define what the function should do. This can include calculations, conditional statements, loops, and other operations.
3. **Output (Return Value):** Functions can optionally return a value as a result of their execution. This return value is the output of the function and can be used in the rest of the program.
4. **Reusability:** One of the main benefits of functions is their reusability. Once you define a function, you can call it multiple times from different parts of your code, avoiding code duplication and promoting code modularity.

Here's how **return** works in functions:

1. **Returning a Value:**
    - When a function executes a **return** statement, it exits the function immediately and returns the specified value to the code that called the function.
    - This returned value can be used by the calling code.

Juice Mixture:

Traditional Function:

```
//--------------------Juice Mixture----------------------
// Function definition: Mixes fruits and vegetables to make a juice
function makeJuice(fruit, vegetable) {
    let juice = `Juice made with ${fruit} and ${vegetable}`;
    return juice;
}
```

Function Expression:

```
// Calling the function with input (ingredients)
let myJuice = makeJuice('apple', 'carrot');

// Output (result) of the function
console.log(myJuice); // Output: Juice made with apple and carrot
```

Object constructor function: constructor functions are used to create objects with shared properties and methods.

```
var conStructFunction = function(name, age, role)
{
    this.name = name,
    this.age = age,
    this.role = role
}

console.log(student)
console.log(student.name)
```

Scope:

Scope defines the accessibility of variables and functions at various parts of your code. In JavaScript, there are primarily two types of scope: **global scope** and **function scope**.

Global Scope

A variable or function has global scope if it is declared outside of any function. These variables and functions are accessible from anywhere in the code.

```
var globalVariable = "I am a global variable";
function showGlobalVariable() {
    console.log(globalVariable); // Accessible here
}
showGlobalVariable(); // Output: "I am a global variable"
console.log(globalVariable); // Accessible here too
```

Hema Coding School

YouTube Link: https://www.youtube.com/@HemaCodingSchool

Function Scope

A variable or function has function scope if it is declared within a function. These variables are only accessible within that function and cannot be accessed from outside.

```
function showLocalVariable() {
    var localVariable = "I am a local variable";
    console.log(localVariable); // Accessible here
}
showLocalVariable(); // Output: "I am a local variable"
console.log(localVariable); // Error: localVariable is not defined
```

IIFE ➔ Immediate Invoke function

Normal Function:

- A normal function is defined with a name and can be called at any point in your code by referencing that name.
- It's typically used when you want to reuse the function multiple times or call it in response to an event or user action.

IIFE (Immediately Invoked Function Expression):

- An IIFE is defined as an anonymous function (a function without a name) that is immediately executed after it's defined.
- It's often used to create a private scope for variables and functions, preventing pollution of the global scope.

```
// Normal function
function greet(name) {
    console.log('Hello, ' + name + '!');
}

// Calling the normal function
greet('John'); // Outputs: Hello, John!

// IIFE example
(function(name) {
    console.log('Hello, ' + name + ' from an IIFE!');
})('Jane'); // Immediately invoked with 'Jane'
// Outputs: Hello, Jane from an IIFE!
```

Tasks:

Task 1: Calculate Average Write a function called **calculateAverage** that takes an array of numbers as a parameter and returns the average of those numbers.

```
//1
function calculateAverage(numbers) {
```

```
    var sum = 0;
    for (var num of numbers) {
        sum += num;
    }
    return sum / numbers.length;
}

var numbers = [5, 10, 15, 20];
console.log(calculateAverage(numbers)); // Output: 12.5
```

Task 2: Check Prime Number Write a function called **isPrime** that takes a number as a parameter and returns true if the number is prime, otherwise false.

```
//2
function isPrime(number) {
    if (number <= 1) {
        return false;
    }
    for (var i = 2; i <= Math.sqrt(number); i++) {
        if (number % i === 0) {
            return false;
        }
    }
    return true;
}

console.log(isPrime(7)); // Output: true
console.log(isPrime(12)); // Output: false
```

Task 3: Reverse String Write a function called **reverseString** that takes a string as a parameter and returns the reversed version of that string.

```
//3
function reverseString(str) {
    return str.split('').reverse().join('');
}

var inputString = 'hello';
console.log(reverseString(inputString)); // Output: 'olleh'
```

Task 4: Palindrome Check Write a function called **isPalindrome** that takes a string as a parameter and returns true if the string is a palindrome (reads the same backward as forward), otherwise false.

```
//4
function isPalindrome(str) {
```

```
    var reversed = str.split('').reverse().join('');
    return str === reversed;
}

console.log(isPalindrome('racecar')); // Output: true
console.log(isPalindrome('hello')); // Output: false
```

Task 5: Fibonacci Sequence Write a function called **fibonacci** that takes a number **n** as a parameter and returns an array containing the first **n** numbers in the Fibonacci sequence.

```
//5
function fibonacci(n) {
    var fib = [0, 1];
    for (var i = 2; i < n; i++) {
        fib[i] = fib[i - 1] + fib[i - 2];
    }
    return fib.slice(0, n);
}

console.log(fibonacci(5)); // Output: [0, 1, 1, 2, 3]
console.log(fibonacci(8)); // Output: [0, 1, 1, 2, 3, 5, 8, 13]
```

Interview Questions:

What is a function in JavaScript?

Answer: A function in JavaScript is a block of code designed to perform a particular task. It is executed when "called" (invoked). Functions can accept inputs (parameters) and return an output (result).

What is the difference between console.log() and return in a function?

Answer:

- console.log() prints the output to the console and is mainly used for debugging.
- return exits the function and returns a value to the caller, which can be used in further calculations or logic.

What are anonymous functions in JavaScript? Provide an example.

- **Answer**: Anonymous functions are functions without a name. They are often used as arguments to other functions or assigned to variables

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

Hema Coding School

YouTube Link: https://www.youtube.com/@HemaCodingSchool

Explain the difference between function declarations and function expressions.

- **Answer**: Function declarations define a function with a specified name and are hoisted to the top of their scope. Function expressions define a function as part of an expression and are not hoisted.

What is an Immediately Invoked Function Expression (IIFE)? Provide an example.

- **Answer**: An IIFE is a function that is executed immediately after it is defined.

```
(function(name){
   console.log(name)
})("Mahesh")
```

What is function currying in JavaScript? Provide an example.

- **Answer**: Currying is a technique of evaluating a function with multiple arguments, into a sequence of functions with a single argument.

```
function multiply(a) {
   return function(b) {
      return a * b;
   };
}

const multiplyBy2 = multiply(2);
console.log(multiplyBy2(3)); // Output: 6
```