Name : Patel Joy M.

Education : 4th year Electronics & communication engineering

College : GEC Gandhinagar

Project Name : JavaScript Concepts and Fundamentals

Task Name : Functions, Scope, and Closures

Table of content

Functions in JavaScript.

- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function is executed when it is called.
- Ex : funcName();
- Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
</head>

<body>
    <h3 id="show"></h3>
    <script>
        // Function to compute the product of p1 and p2

        function myFunction(p1, p2) {
            return p1 * p2;
        }
        // calling the  function :
        var result = myFunction(5, 4)
        document.getElementById("show").innerHTML = "OUTPUT : " + result;
    </script>
</body>

</html>
```

Patel Joy Manojkumar Task Allott

127.0.0.1:5500/index.html

127.0.0.1:5500/index.html

OUTPUT : 20

- The code to be executed, by the function, is placed inside curly brackets: {}

- Function parameters are listed inside the parentheses () in the function definition.

- Function arguments are the values received by the function when it is invoked.

- Inside the function, the arguments (the parameters) behave as local variables.

Why Functions?

- You can reuse code: Define the code once, and use it many times.

- You can use the same code many times with different arguments, to produce different results.

Scope in JavaScript

Scope determines the accessibility (visibility) of variables.

- JavaScript has 3 types of scope:

- Block scope
- Function scope
- Global scope

Block Scope

- Before ES6 (2015), JavaScript had only Global Scope and Function Scope.

- ES6 introduced two important new JavaScript keywords: let and const.

- These two keywords provide Block Scope in JavaScript.

- Variables declared inside a { } block cannot be accessed from outside the block:

- Scope of let keyword.

- example :

```
{
  let x = 2;
}
// x can NOT be used here (outside the
   curly brackets).
```

Scope of var keyword

Variables declared with the var keyword can NOT have block scope.

ex :

```
{
  var x = 2;
}
// x can be used here (outside the bracket).
```

Function Scope

- JavaScript has function scope: Each function creates a new scope.

- Variables defined inside a function are not accessible (visible) from outside the function.

- Variables declared with var, let and const are quite similar when declared inside a function.

```
<script>

    // var
    function myFunction() {
        var carName = "Volvo";    // Function Scope
    }

    // let

    function myFunction() {
        let carName = "Volvo";    // Function Scope
    }

    // const

    function myFunction() {
        const carName = "Volvo";    // Function Scope
    }

</script>
```

value of var, let and const can not be accessed out of the function.

Global Scope

- Variables declared Globally (outside any function) have Global Scope.

- All scripts and functions on a web page can access it.
- Global variables can be accessed from anywhere in a JavaScript program.

- Variables declared with var, let and const are quite similar when declared outside a block.

```
<script>

    //  They all have Global Scope:


    var x = 2;        // Global scope
    let x = 2;        // Global scope
    const x = 2;        // Global scope


    function myFunction() {
        var carName = "Volvo";   // Function Scope
    }


</script>
```

JavaScript Closures

- A closure can be defined as a JavaScript feature in which the inner function has access to the outer function variable. In JavaScript, every time a closure is created with the creation of a function.

- The closure has three scope chains listed as follows:

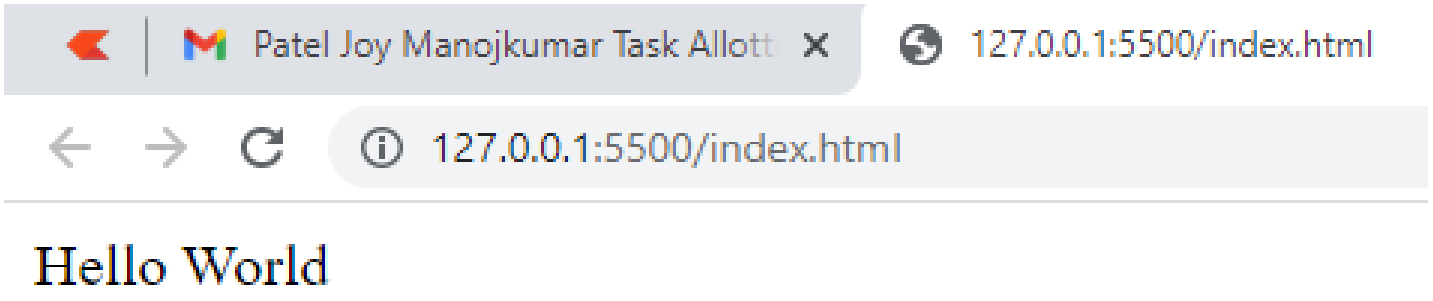
  - Access to its own scope.
  - Access to the variables of the outer function.
  - Access to the global variables.

```
<script>

function fun()
{
var msg1 = "Hello World"; // 'msg1' is the local variable, created by the fun()

function innerfun() // the innerfun() is the inner function, or a closure
{
return msg1;
}
return innerfun;
}
var output = fun();
document.write(output());

</script>
```



- In the above program we have two functions: fun() and innerfun(). The function fun() creates the local variable msg1 and the function innerfun(). The inner function innerfun() is only present in the body of fun(). The inner function can access the outer function's variable, so the function innerfun() can access the variable 'msg1', which is declared and defined in fun().

- This is the closure in action in which the inner function can have access to the global variables and outer function variables.

Thank You