**Name – Kuldeep Kumar**

**Email id- 2019082@iiitdmj.ac.in**

**Task name- Function , scope , closure**

**JavaScript Functions**

A JavaScript function is a block of code designed to perform a particular task.

## A JavaScript function is defined with the function keyword, followed by a **name**, followed by parentheses **()**.

## 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.

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.

function name(parameter1, parameter2, parameter3) {  
  // code to be executed  
}

# JavaScript Scope

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:

## Local Scope

Variables declared within a JavaScript function, become **LOCAL** to the function.

# // code here can NOT use carName function myFunction() {

# let carName = "Volvo";   // code here CAN use carName } // code here can NOT use carName

## 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.

They all have **Function Scope**:

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

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

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

## Global Scope

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

**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.

They all have **Global Scope**:

# JavaScript Closures

JavaScript variables can belong to the **local** or **global** scope.

Global variables can be made local (private) with **closures**.

function myFunction() {  
  let a = 4;  
  return a \* a;  
}

But a function can also access variables defined **outside** the function, like this:

let a = 4;  
function myFunction() {  
  return a \* a;  
}

In the last example, **a** is a **global** variable.

In a web page, global variables belong to the page.

Global variables can be used (and changed) by all other scripts in the page.

In the first example, **a** is a **local** variable.

A local variable can only be used inside the function where it is defined. It is hidden from other functions and other scripting code.

Global and local variables with the same name are different variables. Modifying one, does not modify the other.