CHAPTER 3 – FUNCTIONS

JS Functions

A JS Function is a block of designed to perform a particular task.

Syntax → *function name(parameter1, parameter2){*

//code to be executed

}

Function Invocation

A code inside the function will execute when "something" invokes (calls) the function:

- When an event occurs (when a user clicks a button).
- When it is invoked (called) from JS code.
- Automatically (self invoked)

Function Return

When JS reaches a return statement, the function will stop executing.

Functions often compute a return value. The return value is "returned" back to the "caller":

e.g.- *let x = myFunction(4,3);*

//Function is called, return value will end up in x

function myFunction(a,b){

*return a*b;*

//Function returns the product of a and b

}

Function Invocation

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.

e.g.- *function toCelsius(fahrenheit){*

*return(5/9) * (fahrenheit-32);*

}

document.getElementById("demo").innerHTML

=toCelsius(77);

document.getElementById("demo").innerHTML

=toCelsius(46);

Note : The () Operator invokes the function. Accessing a function without() will return the function object instead of the function result.

e.g.- *toCelsius* → *refers to the function object*

toCelsius() → *refers to the function result*

Local Variables

Variables declared within a JS Function, become LOCAL to the function.

Local variables can be only be accessed from within the function.

e.g.-
```
function myFunction(){
  let carName = "Volvo";
//scope of carName is valid only here
}
```

Global Variables

Variables declared outside the JS Function & anywhere in the program are known as the GLOBAL Variables.

e.g.-
```
let x=1;   //here x is the global variable
  function Name(){ }
```