JavaScript Functions

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

A JavaScript function is executed when "something" invokes it (calls it).

## **JavaScript Function Syntax**

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

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

The parentheses may include parameter names separated by commas:  
**(parameter1, parameter2, ...)**

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

## **Function Declarations**

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

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.

Semicolons are used to separate executable JavaScript statements.  
Since a function **declaration** is not an executable statement, it is not common to end it with a semicolon.

A Function is much the same as a Procedure or a Subroutine, in other programming languages.

Declared functions are not executed immediately. They are "saved for later use", and will be executed later, when they are invoked (called upon).

### **Example**

function myFunction(a, b)

{  
  return a \* b;  
}

## **Function Invocation**

The 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 JavaScript code
* Automatically (self invoked)

## **Function Return**

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

If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.

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

### **Example**

Calculate the product of two numbers, and return the result:

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  
}

The result in x will be:

12

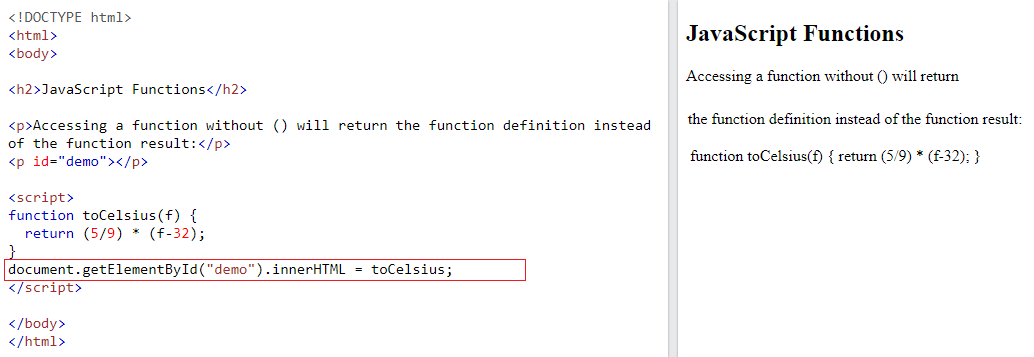
## **Why Functions?**

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.

## **The () Operator Invokes the Function**

Accessing a function without () will return the function object instead of the function result.



## **Functions Used as Variable Values**

Functions can be used the same way as you use variables, in all types of formulas, assignments, and calculations.

### **Example**

Instead of using a variable to store the return value of a function:

let x = toCelsius(77);  
let text = "The temperature is " + x + " Celsius";

You can use the function directly, as a variable value:

let text = "The temperature is " + toCelsius(77) + " Celsius";

## **Local Variables**

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

Local variables can only be accessed from within the function.

### **Example**

// code here can NOT use carName  
  
function myFunction() {  
  let carName = "Volvo";  
  // code here CAN use carName  
}  
  
// code here can NOT use carName

Since local variables are only recognized inside their functions, variables with the same name can be used in different functions.

Local variables are created when a function starts, and deleted when the function is completed.