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).

Example

function myFunction(p1, p2) {  
    return p1 \* p2;              // The function returns the product of p1 and p2  
}

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

Function **parameters** are the **names** listed in the function definition.

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

Inside the function, the arguments behave as local variables.

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

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)

You will learn a lot more about function invocation later in this tutorial.

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:

var 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.

Example

Convert Fahrenheit to Celsius:

function toCelsius(fahrenheit) {  
    return (5/9) \* (fahrenheit-32);  
}  
document.getElementById("demo").innerHTML = toCelsius(77);

The () Operator Invokes the Function

Using the example above, toCelsius refers to the function object, and toCelsius() refers to the function result.

Example

Accessing a function without () will return the function definition:

function toCelsius(fahrenheit) {  
    return (5/9) \* (fahrenheit-32);  
}  
document.getElementById("demo").innerHTML = toCelsius;

Functions Used as Variables

In JavaScript, you can use functions the same way as you use variables.

Example

You can use:

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

Instead of:

var x = toCelsius(32);  
var text = "The temperature is " + x + " Celsius";