JavaScript Introduction

## **JavaScript Can Change HTML Content**

One of many JavaScript HTML methods is getElementById().

The example below "finds" an HTML element (with id="demo"), and changes the element content (innerHTML) to "Hello JavaScript":

### **Example**

document.getElementById("demo").innerHTML = "Hello JavaScript";

JavaScript accepts both double and single quotes:

### **Example**

document.getElementById('demo').innerHTML = 'Hello JavaScript';

## **JavaScript Can Change HTML Attribute Values**

In this example JavaScript changes the value of the src (source) attribute of an <img> tag:

## **JavaScript Can Change HTML Styles (CSS)**

Changing the style of an HTML element, is a variant of changing an HTML attribute:

### **Example**

document.getElementById("demo").style.fontSize = "35px";

## **JavaScript Can Hide HTML Elements**

Hiding HTML elements can be done by changing the display style:

### **Example**

document.getElementById("demo").style.display = "none";

## **JavaScript Can Show HTML Elements**

Showing hidden HTML elements can also be done by changing the display style:

### **Example**

document.getElementById("demo").style.display = "block";

## **The <script> Tag**

In HTML, JavaScript code is inserted between <script> and </script> tags.

### **Example**

<script>  
document.getElementById("demo").innerHTML = "My First JavaScript";  
</script>

## **JavaScript in <head> or <body>**

You can place any number of scripts in an HTML document.

Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

## **JavaScript in <head>**

In this example, a JavaScript function is placed in the <head> section of an HTML page.

The function is invoked (called) when a button is clicked:

### **Example**

<!DOCTYPE html>  
<html>

<head>  
<script>  
function myFunction() {  
  document.getElementById("demo").innerHTML = "Paragraph changed.";  
}  
</script>  
</head>  
<body>

<h1>A Web Page</h1>  
<p id="demo">A Paragraph</p>  
<button type="button" onclick="myFunction()">Try it</button>

</body>  
</html>

## **JavaScript in <body>**

In this example, a JavaScript function is placed in the <body> section of an HTML page.

The function is invoked (called) when a button is clicked:

### **Example**

<!DOCTYPE html>  
<html>  
<body>  
  
<h1>A Web Page</h1>  
<p id="demo">A Paragraph</p>  
<button type="button" onclick="myFunction()">Try it</button>  
  
<script>  
function myFunction() {  
  document.getElementById("demo").innerHTML = "Paragraph changed.";  
}  
</script>  
  
</body>  
</html>

## **External JavaScript**

Scripts can also be placed in external files:

### **External file: myScript.js**

function myFunction() {  
  document.getElementById("demo").innerHTML = "Paragraph changed.";  
}

External scripts are practical when the same code is used in many different web pages.

JavaScript files have the file extension **.js**.

To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

### **Example**

<script src="myScript.js"></script>

## **External JavaScript Advantages**

Placing scripts in external files has some advantages:

* It separates HTML and code
* It makes HTML and JavaScript easier to read and maintain
* Cached JavaScript files can speed up page loads

To add several script files to one page  - use several script tags:

### **Example**

<script src="myScript1.js"></script>  
<script src="myScript2.js"></script>

## **External References**

External scripts can be referenced with a full URL or with a path relative to the current web page.

This example uses a full URL to link to a script:

### **Example**

<script src="https://www.w3schools.com/js/myScript1.js"></script>

This example uses a script located in a specified folder on the current web site:

### **Example**

<script src="/js/myScript1.js"></script>

This example links to a script located in the same folder as the current page:

### **Example**

<script src="myScript1.js"></script>

You can read more about file paths in the chapter [HTML File Paths](https://www.w3schools.com/html/html_filepaths.asp).

# 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).

### **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 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.

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)

## **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.

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

### **Example**

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

## **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:

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

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

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