# JavaScript Where To

**The <script> Tag**

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

## javaScript Functions and Events

A JavaScript **function** is a block of JavaScript code, that can be executed when "called" for.

For example, a function can be called when an **event** occurs, like when the user clicks a button.

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

### Example

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

# JavaScript Output

JavaScript can "display" data in different ways:

* Writing into an HTML element, using **innerHTML**.
* Writing into the HTML output using **document.write()**.
* Writing into an alert box, using **window.alert()**.
* Writing into the browser console, using **console.log()**.

## Using innerHTML

To access an HTML element, JavaScript can use the **document.getElementById(id)** method.

The **id** attribute defines the HTML element. The **innerHTML** property defines the HTML content:

### Example

<!DOCTYPE html>  
<html>  
<body>  
  
<h1>My First Web Page</h1>  
<p>My First Paragraph</p>  
  
<p id="demo"></p>  
  
<script>  
document.getElementById("demo").innerHTML = 5 + 6;  
</script>  
  
</body>  
</html>

## Using document.write()

For testing purposes, it is convenient to use **document.write()**:

<script>  
document.write(5 + 6);  
</script>

## Using window.alert()

You can use an alert box to display data:

### Example

<script>  
window.alert(5 + 6);  
</script>

## Using console.log()

For debugging purposes, you can use the **console.log()** method to display data.

<script>  
console.log(5 + 6);  
</script>

## JavaScript Comments

Not all JavaScript statements are "executed".

Code after double slashes **//** or between **/\*** and **\*/** is treated as a **comment**.

# JavaScript Variables

<script>

var x = 5;

var y = 6;

var z = x + y;

document.getElementById("demo").innerHTML = z;

</script>

## JavaScript String Operators

The + operator can also be used to add (concatenate) strings.

### Example

txt1 = "John";  
txt2 = "Doe";  
txt3 = txt1 + " " + txt2;

The result of txt3 will be:

John Doe

**JavaScript Data Types**

JavaScript variables can hold many **data types**: numbers, strings, objects and more:

var length = 16;                               // Number  
var lastName = "Johnson";                      // String  
var x = {firstName:"John", lastName:"Doe"};    // Object

## JavaScript Types are Dynamic.

JavaScript has dynamic types. This means that the same variable can be used to hold different data types:

### Example

var x;               // Now x is undefined  
var x = 5;           // Now x is a Number  
var x = "John";      // Now x is a String

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

**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  
}