**JS Examples:**

**Change HTML Attribute Values:** changes the value of the src (source) attribute of an <img> tag.

Ex.

<button onclick="document.getElementById('myImage').src='pic\_bulbon.gif'">Turn on the light</button>

<img id="myImage" src="pic\_bulboff.gif" style="width:100px">

<button onclick="document.getElementById('myImage').src='pic\_bulboff.gif'">Turn off the light</button>

**Change HTML Styles (CSS):**

Ex.

<p id="demo">JavaScript can change the style of an HTML element.</p>

<button type="button" onclick="document.getElementById('demo').style.fontSize='35px'">

Click Me!

</button>

**Hide HTML Elements:** Can be done by changing the display style.

Ex.

<p id="demo">JavaScript can hide HTML elements.</p>

<button type="button" onclick="document.getElementById('demo').style.display='none'">Click Me!</button>

**Show HTML Elements:** Can also be done by changing the display style.

Ex.

<p id="demo" style="display:none">Hello JavaScript!</p>

<button type="button" onclick="document.getElementById('demo').style.display='block'">Click Me!</button>

**JavaScript Display Possibilities**

**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

Ex.

<p id="demo"></p> <script> document.getElementById("demo").innerHTML = 5 + 6; </script>

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

*Using document.write() after an HTML document is loaded, will delete all existing HTML*

Ex.

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

*---This below overwrites the <p>---*

<p>My first paragraph.</p>

<button type="button" onclick="document.write(5 + 6)">Try it</button>

**window.alert():** An alert box to display data.

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

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

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

**JavaScript Statements**

JavaScript statements are composed of:

Values, Operators, Expressions, Keywords, and Comments.

This statement tells the browser to write "Hello Dolly." inside an HTML element with id="demo":

Ex. document.getElementById("demo").innerHTML = "Hello Dolly.";

**JavaScript Programs**

A computer program is a list of "instructions" to be "executed" by a computer.

In a programming language, these programming instructions are called statements.

A JavaScript program is a list of programming statements.

*Add a semicolon at the end of each executable statement*

[**JavaScript Code Blocks**](https://www.w3schools.com/js/js_reserved.asp)

JavaScript statements can be grouped together in code blocks, inside curly brackets {...}. The purpose of code blocks is to define statements to be executed together. One place you will find statements grouped together in blocks, is in JavaScript functions.

function myFunction() {  
  document.getElementById("demo1").innerHTML = "Hello Dolly!";  
  document.getElementById("demo2").innerHTML = "How are you?";  
}

**JavaScript Keywords:** JavaScript statements often start with a keyword to identify the JavaScript action to be performed.

|  |  |
| --- | --- |
| **Keyword** | **Description** |
| break | Terminates a switch or a loop |
| continue | Jumps out of a loop and starts at the top |
| debugger | Stops the execution of JavaScript, and calls (if available) the debugging function |
| do ... while | Executes a block of statements, and repeats the block, while a condition is true |
| for | Marks a block of statements to be executed, as long as a condition is true |
| function | Declares a function |
| if ... else | Marks a block of statements to be executed, depending on a condition |
| return | Exits a function |
| switch | Marks a block of statements to be executed, depending on different cases |
| try ... catch | Implements error handling to a block of statements |
| var | Declares a variable |

[**JavaScript Arithmetic Operators**](https://www.w3schools.com/js/js_arithmetic.asp)**:** Arithmetic operators are used to perform arithmetic on numbers

|  |  |
| --- | --- |
| **Operator** | **Description** |
| + | Addition |
| - | Subtraction |
| \* | Multiplication |
| \*\* | Exponentiation ([ES2016](https://www.w3schools.com/js/js_es6.asp)) |
| / | Division |
| % | Modulus (Division Remainder) |
| ++ | Increment |
| -- | Decrement |

**JavaScript Assignment Operators**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Example** | **Same As** |
| = | x = y | x = y |
| += | x += y | x = x + y |
| -= | x -= y | x = x - y |
| \*= | x \*= y | x = x \* y |
| /= | x /= y | x = x / y |
| %= | x %= y | x = x % y |
| <<= | x <<= y | x = x << y |
| >>= | x >>= y | x = x >> y |
| >>>= | x >>>= y | x = x >>> y |
| &= | x &= y | x = x & y |
| ^= | x ^= y | x = x ^ y |
| |= | x |= y | x = x | y |
| \*\*= | x \*\*= y | x = x \*\* y |

**JavaScript Arrays**

JavaScript arrays are written with square brackets “[]”. Array items are separated by commas “,”. The following code declares (creates) an array called cars, containing three items (car names):

var cars = ["Saab", "Volvo", "BMW"];

**JavaScript Objects**

JavaScript objects are written with curly braces “{}”.

Object properties are written as name:value pairs, separated by commas “,”.

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

**The typeof Operator**

You can use the JavaScript typeof operator to find the type of a JavaScript variable. (number, string, boolean, undefined / function, object).

**JavaScript Functions**

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

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

**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":

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  
}