# JAVASCRIPT

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

**3 Places to put JavaScript code:**

|  |  |  |
| --- | --- | --- |
| **Between the body tag of html** | **Between the head tag of html** | **In .js file (external javaScript)** |
| <body>  <script type="text/javascript">  document.write("JavaScript is a simple language for javatpoint learners");  </script>  </body> | <head>  <script type="text/javascript">  function msg(){  alert("Hello Javatpoint");  }  </script>  </head>  <body>  <p>Welcome to JavaScript</p>  <form>  <input type="button" value="click" onclick="msg()"/>  </form>  </body> | **message.js**  function msg(){   alert("Hello Javatpoint");  }  ****index.html****  <head>  <script type="text/javascript" src="message.js"></script>  </head>  <body>  <p>Welcome to JavaScript</p>  <form>  <input type="button" value="click" onclick="msg()"/>  </form>  </body> |

**JavaScript Variable and Datatype:**

**variables**

1. variables are used to store and manage data.
2. Var, let, const
3. The let keyword is a block-scoped variables.
4. The const keyword declares variables that cannot be reassigned. It’s block-scoped as well.

**Data -** primitive data types and Non primitive data types

1. primitive data types

- Number

- String

- Boolean

- Null

- Undefined

- Symbol

- BigInt

1. non-primitive data types

- Object

**JavaScript Operators**

* [Assignment operators](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_operators" \l "assignment_operators) =
* [Comparison operators](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_operators" \l "comparison_operators) == and ===
* [Arithmetic operators](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_operators" \l "arithmetic_operators) + - \*/
* [Bitwise operators](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_operators" \l "bitwise_operators)
* [Logical operators](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_operators" \l "logical_operators) AND(&&) OR (||) NOT(!)
* [BigInt operators](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_operators" \l "bigint_operators)
* [String operators](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_operators" \l "string_operators) console.log(“my” + “string”);
* [Conditional (ternary) operator](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_operators" \l "conditional_ternary_operator) condition? Value1 : value2
* [Comma operator](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_operators" \l "comma_operator)
* [Unary operators](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_operators" \l "unary_operators)
* [Relational operators](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_operators" \l "relational_operators)

**JAVASCRIPT LOOPS**

1. [for Loop](https://www.geeksforgeeks.org/loops-in-javascript/?ref=lbp" \l "javascript-for-loop)
2. [while Loop](https://www.geeksforgeeks.org/loops-in-javascript/?ref=lbp" \l "javascript-while-loop)
3. [do-while Loop](https://www.geeksforgeeks.org/loops-in-javascript/?ref=lbp" \l "javascript-dowhile-loop)
4. [for-in Loop](https://www.geeksforgeeks.org/loops-in-javascript/?ref=lbp" \l "javascript-forin-loop)
5. [for-of Loop](https://www.geeksforgeeks.org/loops-in-javascript/?ref=lbp" \l "javascript-forof-loop)
6. [Labeled Statement](https://www.geeksforgeeks.org/loops-in-javascript/?ref=lbp" \l "javascript-labeled-statement)
7. [Break Statement](https://www.geeksforgeeks.org/loops-in-javascript/?ref=lbp" \l "javascript-break-statement)
8. [Continue Statement](https://www.geeksforgeeks.org/loops-in-javascript/?ref=lbp" \l "javascript-continue-statement)
9. [Infinite Loop (Loop Error)](https://www.geeksforgeeks.org/loops-in-javascript/?ref=lbp" \l "javascript-infinite-loop-loop-error)

|  |  |  |
| --- | --- | --- |
| [for Loop](https://www.geeksforgeeks.org/loops-in-javascript/?ref=lbp" \l "javascript-for-loop) | for (initialization; testing condition; increment/decrement) {  statement(s)  } | for (x = 2; x <= 4; x++) { console.log("Value of x: " + x);} |
| [while Loop](https://www.geeksforgeeks.org/loops-in-javascript/?ref=lbp" \l "javascript-while-loop) | while (boolean condition) {  loop statements...  } | let val = 1;  while (val < 6) { console.log(val); val += 1;} |
| [do-while Loop](https://www.geeksforgeeks.org/loops-in-javascript/?ref=lbp" \l "javascript-dowhile-loop) | do {  Statements...  }  while (condition); | let test = 1;  do { console.log(test); test++;} while(test <= 5) |
| For- in loop | for(let variable\_name in object\_name) {  // Statement  } | let myObj = { x: 1, y: 2, z: 3 };for (let key in myObj) { console.log(key, myObj[key]);} |

**Type coercion in javascript:**

In JavaScript, type coercion is the automatic or implicit conversion of values from one data type to another (such as strings to numbers). JavaScript is a loosely typed or dynamically typed language, which means that you don't need to specify the type of a variable explicitly. This flexibility can lead to implicit type coercion in various situations. Here's a detailed look at type coercion in JavaScript:

**Types of Type Coercion**

1. **Implicit Coercion**: This occurs automatically during the evaluation of expressions.
2. **Explicit Coercion**: This is done manually by using functions and operators to convert data types.

**Implicit Type Coercion**

1. String and Number Addition: let result = 5 + '5'; // result is '55' (string)
2. Boolean to Number: let result = true + 2; // result is 3 (true is coerced to 1)

let result = false + 2; // result is 2 (false is coerced to 0)

1. Null and Undefined: let result = null + 5; // result is 5 (null is coerced to 0)

let result = undefined + 5; // result is NaN (undefined cannot be coerced to a number)

1. Comparison Operators: let result = '5' == 5; // result is true (string '5' is coerced to number 5)

let result = '5' === 5; // result is false (strict equality, no coercion)

**Explicit Type Coercion**

1. Number to String: let num = 5;

let str = String(num); // str is '5'

let str = num.toString(); // str is '5'

1. String to Number: let str = '123';

let num = Number(str); // num is 123

let num = parseInt(str);// num is 123

let num = parseFloat('123.45'); // num is 123.45

1. Boolean to Number: let bool = true;

let num = Number(bool); // num is 1

let num = +bool; // num is 1

1. Number to Boolean: let num = 0;

let bool = Boolean(num); // bool is false

let bool = !!num; // bool is false

**String Built-in Methods**

1. CharAt() - Returns the character at a specified index (position)

let text = "HELLO WORLD";  
 let letter = text.charAt(0); // output H

1. CharCodeAt(){Unicode}-Returns the Unicode of the character at a specified index

let text = "HELLO WORLD";  
 let code = text.charCodeAt(0); // output 72

1. Concat() - Returns two or more joined strings

let text1 = "sea";  
 let text2 = "food";  
 let result = text1.concat(text2);// output seafood

1. Repeat() - Returns a new string with a number of copies of a string

let text = "Hello world!";  
 let result = text.repeat(2); // output Hello world!Hello world!

1. Includes() - Returns if a string contains a specified value

let text = "Hello world, welcome to the universe.";  
 let result = text.includes("world"); // output true

1. Match() - Searches a string for a value, or a regular expression, and returns the matches

let text = "The rain in SPAIN stays mainly in the plain";  
 text.match("ain"); // output ain

1. Replace() - Searches a string for a pattern, and returns a string where the first match is replaced

let text = "Visit HELLO!";  
 let result =text.replace("HELLO", "Microsoft"); // output Visit Microsoft!

1. Search() - Searches a string for a value, or regular expression, and returns the index (position) of the match

let text = "Mr. Blue has a blue house";  
 let position = text.search("Blue"); // output 4

**JS Arrays & Operations**

\* Objects allow you to store keyed collections of values. That’s fine.But quite often we find that we need an ordered collection, where we have a 1st, a 2nd, a 3rd element and so on. For example, we need that to store a list of something: users, goods, HTML elements etc.

\* It is not convenient to use an object here, because it provides no methods to manage the order of elements. We can’t insert a new property “between” the existing ones. Objects are just not meant for such use.

\* There exists a special data structure named Array, to store ordered collections.

**[Declaration](https://javascript.info/array" \l "declaration)**

There are two syntax’s for creating an empty array:

let arr = new Array();

let arr = []; EX: let fruits = ["Apple", "Orange", "Plum"];

\* Array elements are numbered, starting with zero.

let fruits = ["Apple", "Orange", "Plum"];

alert( fruits[0] ); // Apple

alert( fruits[1] ); // Orange

alert( fruits[2] ); // Plum

\* We can replace an element: fruits[2] = 'Pear'; // now ["Apple", "Orange", "Pear"]

\* add a new one to the array:fruits[3] = 'Lemon'; // now ["Apple", "Orange", "Pear", "Lemon"]

\* The total count of the elements in the array is its length:

let fruits = ["Apple", "Orange", "Plum"]; alert( fruits.length ); // 3

\* We can also use alert to show the whole array.

let fruits = ["Apple", "Orange", "Plum"];

alert( fruits ); // Apple,Orange,Plum

**METHODS IN ARRAYS**

A [queue](https://en.wikipedia.org/wiki/Queue_(abstract_data_type)) is one of the most common uses of an array. In computer science, this means an ordered collection of elements which supports two operations:

* + 1. Shift and UnShift - shift() removes the first element from an array and returns it, while unshift() adds one or more elements to the beginning of an array and returns the new length.
    2. Push and Pop - Push () adds items to the end.

pop() removes the element from the end and returns it.

* + 1. Splice - The splice() method adds and/or removes array elements.

\*arr.splice(start[, deleteCount, elem1, ..., elemN])

\* array.splice(index, count, item1, ....., itemX)

const fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.splice(2, 0, "Lemon", "Kiwi");

Banana,Orange,Lemon,Kiwi,Apple,Mango

\*let arr = ["I", "study", "JavaScript"]; arr.splice(1, 1); // from index 1 remove 1 element alert( arr ); // ["I", "JavaScript"]

* + 1. Slice - arr.slice([start], [end])

It returns a new array copying to it all items from index start to end (not including end). Both start and end can be negative, in that case position from array end is assumed.

It’s similar to a string method str.slice, but instead of substrings, it makes subarrays

let arr = ["t", "e", "s", "t"];

alert( arr.slice(1, 3) ); // e,s (copy from 1 to 3)

alert( arr.slice(-2) ); // s,t (copy from -2 till the end)

* + 1. Concat - arr.concat(arg1, arg2...)

let arr = [1, 2];

// create an array from: arr and [3,4]

alert( arr.concat([3, 4]) ); // 1,2,3,4

// create an array from: arr and [3,4] and [5,6]

alert( arr.concat([3, 4], [5, 6]) ); // 1,2,3,4,5,6

// create an array from: arr and [3,4], then add values 5 and 6

alert( arr.concat([3, 4], 5, 6) ); // 1,2,3,4,5,6

**pop**

Extracts the last element of the array and returns it:

let fruits = ["Apple", "Orange", "Pear"];

alert( fruits.pop() ); // remove "Pear" and alert it

alert( fruits ); // Apple, Orange

**Push :** Append the element to the end of the array:

let fruits = ["Apple", "Orange"];

fruits.push("Pear");

alert( fruits ); // Apple, Orange, Pear

**Shift:** Extracts the first element of the array and returns it:

let fruits = ["Apple", "Orange", "Pear"];

alert( fruits.shift() ); // remove Apple and alert it

alert( fruits ); // Orange, Pear

**Unshift:** Add the element to the beginning of the array:

let fruits = ["Orange", "Pear"];

fruits.unshift('Apple');

alert( fruits ); // Apple, Orange, Pear

How to delete an element from the array?

The arrays are objects, so we can try to use delete:

let arr = ["I", "go", "home"];

delete arr[1]; // remove "go"

alert( arr[1] ); // undefined

// now arr = ["I", , "home"];alert( arr.length ); // 3

Array methods

Array sort

Array iteration