What is JavaScript

JavaScript is a programming language initially designed to interact with elements of web pages. In web browsers, JavaScript consists of three main parts:

* ECMAScript provides the core functionality.
* The [Document Object Model (DOM)](https://www.javascripttutorial.net/javascript-dom/) provides interfaces for interacting with elements on web pages
* The [Browser Object Model (BOM)](https://www.javascripttutorial.net/javascript-bom/) provides the browser API for interacting with the web browser.

JavaScript allows you to add interactivity to a web page. Typically, you use JavaScript with HTML and CSS to enhance a web page’s functionality, such as [validating forms](https://www.javascripttutorial.net/javascript-dom/javascript-form-validation/), creating interactive maps, and displaying animated charts.

When a web page is loaded, i.e., after HTML and CSS have been downloaded, the JavaScript engine in the web browser executes the JavaScript code. The JavaScript code then modifies the HTML and CSS to update the user interface dynamically.

The JavaScript engine is a program that executes JavaScript code. In the beginning, JavaScript engines were implemented as interpreters.

However, modern JavaScript engines are typically implemented as just-in-time compilers that compile JavaScript code to bytecode for improved performance.

Client-side vs. Server-side JavaScript

When JavaScript is used on a web page, it is executed in web browsers. In this case, JavaScript works as a client-side language.

JavaScript can run on both web browsers and servers. A popular JavaScript server-side environment is [Node.js](https://www.javascripttutorial.net/nodejs-tutorial/). Unlike client-side JavaScript, server-side JavaScript executes on the server that allows you to access databases, file systems, etc.

JavaScript History

In 1995, JavaScript was created by a Netscape developer named [Brendan Eich](https://en.wikipedia.org/wiki/Brendan_Eich). First, its name was Mocha. And then, its name was changed to LiveScript.

Netscape decided to change LiveScript to JavaScript to leverage Java’s fame, which was popular. The decision was made just before Netscape released its web browser product Netscape Navigator 2. As a result, JavaScript entered version 1.0.

Netscape released JavaScript 1.1 in Netscape Navigator 3. In the meantime, Microsoft introduced a web browser product called the [Internet Explorer](https://en.wikipedia.org/wiki/Internet_Explorer) 3 (IE 3), which competed with Netscape. However, IE came with its own JavaScript implementation called [JScript](https://en.wikipedia.org/wiki/JScript). Microsoft used the name JScript to avoid possible license issues with Netscape.

Hence, two different JavaScript versions were in the market:

* JavaScript in Netscape Navigator
* JScript in Internet Explorer.

JavaScript had no standards that governed its syntax and features. And the community decided that it was time to standardize the language.

In 1997, JavaScript 1.1 was submitted to the [European Computer Manufacturers Association](https://www.ecma-international.org/) (ECMA) as a proposal. [Technical Committee #39](https://www.ecma-international.org/memento/tc39-m.htm) (TC39) was assigned to standardize the language to make it a general-purpose, cross-platform, and vendor-neutral scripting language.

TC39 came up with ECMA-262, a standard for defining a new scripting language named ECMAScript (often pronounced Ek-ma-script).

After that, the International Organization for Standardization and International Electrotechnical Commissions (ISO/IEC) adopted ECMAScript (ISO/IEC-16262).

JavaScript overview

To [define a variable](https://www.javascripttutorial.net/javascript-variables/) in JavaScript, you use var keyword. For example:

var x = 10;

var y = 20;

Code language: JavaScript (javascript)

ES6 added a new way to declare a variable with the [let](https://www.javascripttutorial.net/es6/javascript-let/) keyword:

let x = 10;

let y = 20;

Code language: JavaScript (javascript)

There are differences between [var and let](https://www.javascripttutorial.net/es6/difference-between-var-and-let/). And it’s a good practice to use the let keyword to declare variables.

To declare a [function](https://www.javascripttutorial.net/javascript-function/), you use the function keyword. The following example defines a function that calculates the sum of two arguments:

function add( a, b ) {

return a + b;

}

Code language: JavaScript (javascript)

To call the add() function, you use the following syntax:

let result = add(x, y);

Code language: JavaScript (javascript)

To log the result into the console window of the web browser, you use the console.log() :

console.log(result);

Code language: JavaScript (javascript)

Now, you should see 30 in the console window.

JavaScript provides you with condition statements such as [if-else](https://www.javascripttutorial.net/javascript-if-else/) and [switch](https://www.javascripttutorial.net/javascript-switch-case/) statements. For example:

let a = 20,

b = 30;

function divide(a, b) {

if(b == 0) {

throw 'Division by zero';

}

return a / b;

}

Code language: JavaScript (javascript)

In the divide() function, we check whether the de-numerator (b) is zero. If yes, we [throw an exception](https://www.javascripttutorial.net/javascript-try-catch/). Otherwise, we return the result of a / b.

To declare an [array](https://www.javascripttutorial.net/javascript-array/), you use the following syntax:

let items = [];

Code language: JavaScript (javascript)

To declare an array with some initial elements, you specify the elements in the square brackets:

let items = [1, 2, 3];

Code language: JavaScript (javascript)

You can access the number of elements in the items array through its length property:

console.log(items.length); *// 3*

Code language: JavaScript (javascript)

To iterate over the elements of the items array, you use the [for](https://www.javascripttutorial.net/javascript-for-loop/) loop statement as follows:

for(let i = 0; i < items.length; i++) {

console.log(items[i]);

}

Code language: JavaScript (javascript)

Or use the [for...of](https://www.javascripttutorial.net/es6/javascript-for-of/) loop in [ES6](https://www.javascripttutorial.net/es6/):

for(let item of items) {

console.log(item);

}

JavaScript Hello World Example

**Summary**: This tutorial helps you get started with JavaScript by showing you how to embed JavaScript code into an HTML page.

To insert JavaScript into an HTML page, you use the <script> element. There are two ways to use the <script> element in an HTML page:

* Embed JavaScript code directly into the HTML page.
* Reference an external JavaScript code file.

Embed JavaScript code in an HTML page

Placing JavaScript code inside the <script> element directly is not recommended and should be used only for proof of concept or testing purposes.

The JavaScript code in the <script> element is interpreted from top to bottom. For example:

<!DOCTYPE html>

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<title>**JavaScript Hello World Example**</title>**

**<script>**

alert('Hello, World!');

**</script>**

**</head>**

**<body>**

**</body>**

**</html>**

Code language: HTML, XML (xml)

In the <script> element, we use the [alert()](https://www.javascripttutorial.net/javascript-bom/javascript-alert/) function to display the Hello, World! message.

Include an external JavaScript file

To include a JavaScript from an external file:

* First, create a file whose extension is .js e.g., app.js and place it in the js subfolder. Note that placing the JavaScript file in the js folder is not required however it is a good practice.
* Then, use the URL to the JavasScript source code file in the src attribute of the <script> element.

The following shows the contents of the app.js file:

alert('Hello, World!');

Code language: JavaScript (javascript)

And the following shows the helloworld.html file:

<!DOCTYPE html>

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<title>**JavaScript Hello World Example**</title>**

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

**</head>**

**<body>**

**</body>**

**</html>**

Code language: HTML, XML (xml)

If you launch the helloworld.html file in the web browser, you will see an alert that displays the Hello, World! message.

Note that you can load a JavaScript file from a remote server. This allows you to serve up JavaScript from various domains e.g., content delivery network (CDN) to speed up the page.

When you have multiple JavaScript files on a page, the JavaScript engine interprets the files in the order that they appear. For example:

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

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

Code language: HTML, XML (xml)

In this example, JavaScript engine will interpret the service.js and the app.js files in sequence. It completes interpreting the service.js file first before interpreting the app.js file.

For the page that includes many external JavaScript files, the blank page is shown during the page rendering phase.

To avoid this, you include the JavaScript file just before the </body> tag as shown in this example:

<!DOCTYPE html>

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<title>**JavaScript Hello World Example**</title>**

**</head>**

**<body>**

*<!-- end of page content here-->*

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

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

**</body>**

**</html>**

Code language: HTML, XML (xml)

The async and defer attributes

To change how the browser load and execute JavaScript files, you use one of two attributes of the <script> element async and defer.

These attributes take effect only on the external script files. The async attribute instructs the web browser to execute the JavaScript file asynchronously. The async attribute does not guarantee the script files to execute in the order that they appear. For example:

**<script async src="service.js"></script>**

**<script async src="app.js"></script>**

Code language: HTML, XML (xml)

The app.js file might execute before the service.js file. Therefore, you must ensure that there is no dependency between them.

The defer attribute requests the web browser to execute the script file after the HTML document has been parsed.

<!DOCTYPE html>

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<title>**JavaScript defer demonstration**</title>**

**<script defer src="defer-script.js"></script>**

**</head>**

**<body>**

**</body>**

**</html>**

Code language: HTML, XML (xml)

Even though we place the <script> element in the <head> section, the script will wait for the browser to receive the closing tag <html> to start executing.

Summary

* Use <script> element to include a JavaScript file in a HTML page.
* The async attribute of the <script> element instructs the web browser to fetch the JavaScript file in parallel and then parse and execute as soon as the JavaScript file is available.
* The defer attribute of the <script> element allows the web browser to execute the JavaScript file after the document has been parsed.