**[What is JavaScript?](https://javascript.info/intro" \l "what-is-javascript)**

*JavaScript* was initially created to “make web pages alive”.

The programs in this language are called *scripts*. They can be written right in a web page’s HTML and run automatically as the page loads.

Scripts are provided and executed as plain text. They don’t need special preparation or compilation to run.

In this aspect, JavaScript is very different from another language called [Java](https://en.wikipedia.org/wiki/Java_(programming_language)).

Why is it called JavaScript?

When JavaScript was created, it initially had another name: “LiveScript”. But Java was very popular at that time, so it was decided that positioning a new language as a “younger brother” of Java would help.

But as it evolved, JavaScript became a fully independent language with its own specification called [ECMAScript](http://en.wikipedia.org/wiki/ECMAScript), and now it has no relation to Java at all.

Today, JavaScript can execute not only in the browser, but also on the server, or actually on any device that has a special program called [the JavaScript engine](https://en.wikipedia.org/wiki/JavaScript_engine).

The browser has an embedded engine sometimes called a “JavaScript virtual machine”.

Different engines have different “codenames”. For example:

[V8](https://en.wikipedia.org/wiki/V8_(JavaScript_engine)) – in Chrome, Opera and Edge.

[SpiderMonkey](https://en.wikipedia.org/wiki/SpiderMonkey) – in Firefox.

…There are other codenames like “Chakra” for IE, “JavaScriptCore”, “Nitro” and “SquirrelFish” for Safari, etc.

How do engines work?

Engines are complicated. But the basics are easy.

The engine (embedded if it’s a browser) reads (“parses”) the script.

Then it converts (“compiles”) the script to the machine language.

And then the machine code runs

[What can in-browser JavaScript do?](https://javascript.info/intro" \l "what-can-in-browser-javascript-do)

Modern JavaScript is a “safe” programming language. It does not provide low-level access to memory or CPU, because it was initially created for browsers which do not require it

JavaScript’s capabilities greatly depend on the environment it’s running in. For instance, [Node.js](https://wikipedia.org/wiki/Node.js) supports functions that allow JavaScript to read/write arbitrary files, perform network requests, etc.

In-browser JavaScript can do everything related to webpage manipulation, interaction with the user, and the webserver.

For instance, in-browser JavaScript is able to:

Add new HTML to the page, change the existing content, modify styles.

React to user actions, run on mouse clicks, pointer movements, key presses.

Send requests over the network to remote servers, download and upload files (so-called [AJAX](https://en.wikipedia.org/wiki/Ajax_(programming)) and [COMET](https://en.wikipedia.org/wiki/Comet_(programming)) technologies).

Get and set cookies, ask questions to the visitor, show messages.

Remember the data on the client-side (“local storage”).

[What makes JavaScript unique?](https://javascript.info/intro" \l "what-makes-javascript-unique)

There are at least *three* great things about JavaScript:

Full integration with HTML/CSS.

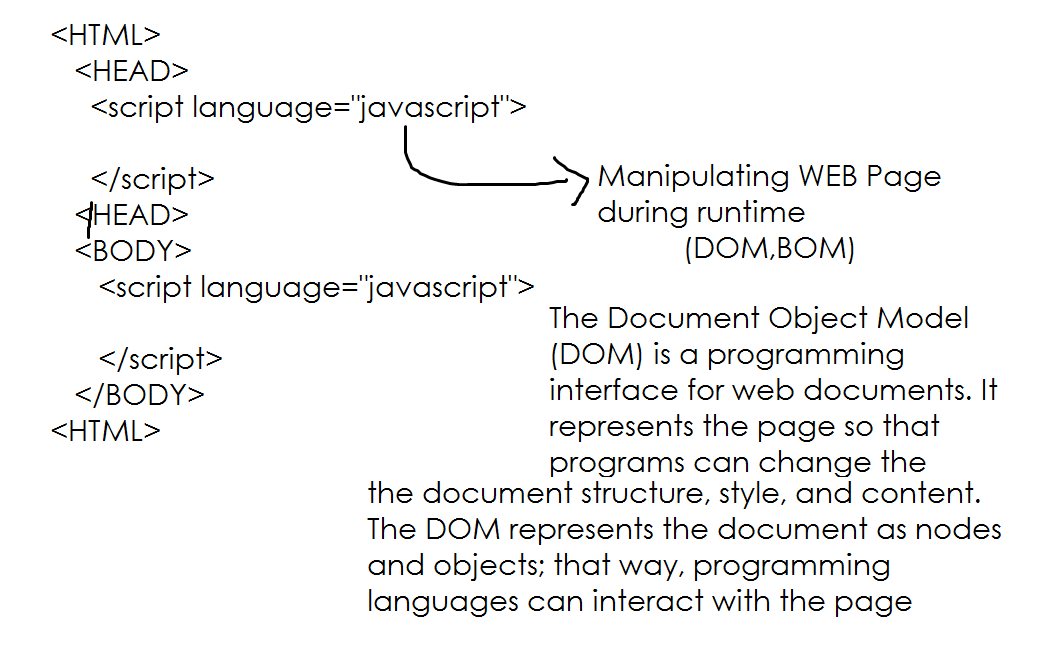
Simple things are done simply.

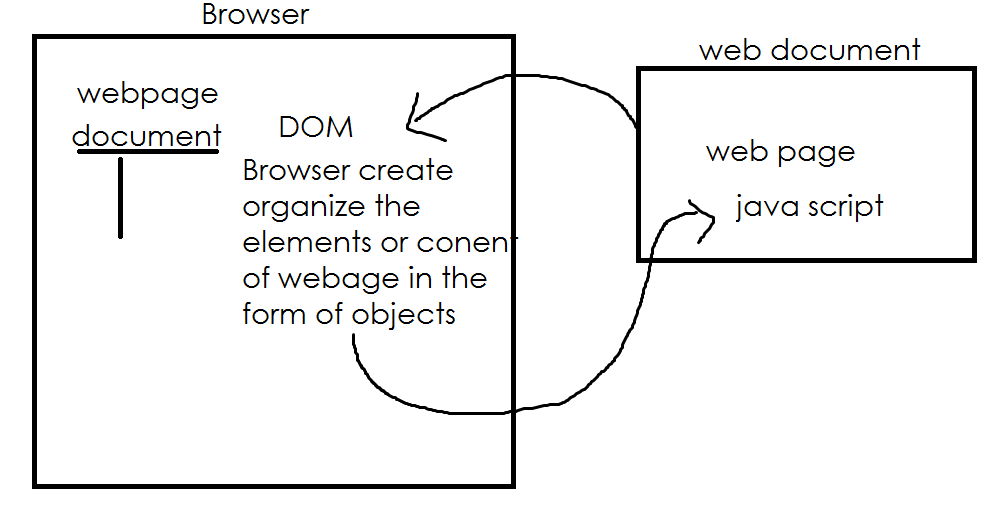
Supported by all major browsers and enabled by default.

JavaScript is the only browser technology that combines these three things.

That’s what makes JavaScript unique. That’s why it’s the most widespread tool for creating browser interfaces.

That said, JavaScript also allows to create servers, mobile applications, etc.





[The “script” tag](https://javascript.info/hello-world" \l "the-script-tag)

JavaScript programs can be inserted almost anywhere into an HTML document using the <script> tag.

<!DOCTYPE HTML>

<html>

<body>

<p>Before the script...</p>

<script>

alert( 'Hello, world!' );

</script>

<p>...After the script.</p>

</body>

</html>

[Modern markup](https://javascript.info/hello-world" \l "modern-markup)

**The <script> tag** has a few attributes that are rarely used nowadays but can still be found in old code:

The type attribute: <script type=…>

The old HTML standard, HTML4, required a script to have a type. Usually it was type="text/javascript". It’s not required anymore. Also, the modern HTML standard totally changed the meaning of this attribute. Now, it can be used for JavaScript modules. But that’s an advanced topic, we’ll talk about modules in another part of the tutorial.

The language attribute: <script language=…>

This attribute was meant to show the language of the script. This attribute no longer makes sense because JavaScript is the default language. There is no need to use it.

**[External scripts](https://javascript.info/hello-world" \l "external-scripts)**

If we have a lot of JavaScript code, we can put it into a separate file.

Script files are attached to HTML with the src attribute:

<script src="/path/to/script.js"></script>

Here, /path/to/script.js is an absolute path to the script from the site root. One can also provide a relative path from the current page. For instance, src="script.js", just like src="./script.js", would mean a file "script.js" in the current folder.

We can give a full URL as well. For instance:

<script src=<https://path>></script>

<script src="/js/script1.js"></script>

<script src="/js/script2.js"></script>

…

<https://www.greycampus.com/blog/programming/java-script-versions>

**Java Script Language Fundamentals**