

An Introduction to JavaScript

Let's see what's so special about JavaScript, what we can achieve with it, and which other technologies play well with it.

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 executed 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](#).

Why 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](#), 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](#).

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

Different engines have different “codenames”. For example:

- [V8](#) – in Chrome and Opera.
- [SpiderMonkey](#) – in Firefox.
- ...There are other codenames like “Trident” and “Chakra” for different versions of IE, “ChakraCore” for Microsoft Edge, “Nitro” and “SquirrelFish” for Safari, etc.

The terms above are good to remember because they are used in developer articles on the internet. We'll use them too. For instance, if “a feature X is supported by V8”, then it probably works in Chrome and Opera.

How do engines work?

Engines are complicated. But the basics are easy.

1. The engine (embedded if it's a browser) reads ("parses") the script.
2. Then it converts ("compiles") the script to the machine language.
3. And then the machine code runs, pretty fast.

The engine applies optimizations at each step of the process. It even watches the compiled script as it runs, analyzes the data that flows through it, and applies optimizations to the machine code based on that knowledge. When it's done, scripts run quite fast.

What makes JavaScript unique?

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

- Full integration with HTML/CSS.
- Simple things are done simply.
- Support 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.

While planning to learn a new technology, it's beneficial to check its perspectives. So let's move on to the modern trends affecting it, including new languages and browser abilities.

Languages "over" JavaScript

The syntax of JavaScript does not suit everyone's needs. Different people want different features.

That's to be expected, because projects and requirements are different for everyone.

So recently a plethora of new languages appeared, which are *transpiled* (converted) to JavaScript before they run in the browser.

Modern tools make the transpilation very fast and transparent, actually allowing developers to code in another language and auto-converting it “under the hood”.

Examples of such languages:

- [CoffeeScript](#) is a “syntactic sugar” for JavaScript. It introduces shorter syntax, allowing us to write clearer and more precise code. Usually, Ruby devs like it.
- [TypeScript](#) is concentrated on adding “strict data typing” to simplify the development and support of complex systems. It is developed by Microsoft.
- [Dart](#) is a standalone language that has its own engine that runs in non-browser environments (like mobile apps). It was initially offered by Google as a replacement for JavaScript, but as of now, browsers require it to be transpiled to JavaScript just like the ones above.

There are more. Of course, even if we use one of these languages, we should also know JavaScript to really understand what we’re doing.