**JavaScript improvements**

In this reading, you will learn about the history of JavaScript and the importance of ECMA (European Computer Manufacturers Association) and ECMAScript.

JavaScript is a programming language that had humble beginnings.

It was built in only 10 days in 1995 by a single person, Brendan Eich, who was tasked with building a simple scripting language to be used in version 2 of the Netscape browser. It was initially called LiveScript, but since the Java language was so popular at the time, the name was changed to JavaScript - although Java and JavaScript are in no way related.

For the first few years, after it was built, JavaScript was a simple scripting language to add mouseover effects and other interactivity. Those effects were being added to webpages using the **<script>** HTML element.

Inside each of the script elements, there could be some JavaScript code. Due to the rule that HTML, CSS, and JavaScript must be backward compatible, even the most advanced code written in JavaScript today ends up being written between those script tags.

Over the years, JavaScript grew ever more powerful, and in recent times, it's continually touted as among the top three commonly used languages.

In 1996 Netscape made a deal with the organization known as ECMA (European Computer Manufacturers Association) to draft the specification of the JavaScript language, and in 1997 the first edition of the ECMAScript specification was published.

ECMA publishes this specification as the ECMA-262 standard.

You can think of a standard as an agreed-upon way of how things should work. Thus, ECMA-262 is a standard that specifies how the JavaScript language should work.

There have been 12 ECMA-262 updates - the first one was in 1997.

JavaScript as a language is not a completely separate, stand-alone entity. It only exists as an implementation. This implementation is known as a JavaScript engine.

Traditionally, the only environment in which it was possible to run a JavaScript engine, was the browser. More specifically, a JavaScript engine was just another building block of the browser. It was there to help a browser accomplish its users' goal of utilizing the internet for work, research, and play.

So, when developers write JavaScript code, they are using it to interact with a JavaScript engine. Put differently, developers write JavaScript code so that they can "talk to" a JavaScript engine.

Additionally, the JavaScript engine itself comes with different ways to interact with various other parts of the browser. These are known as Browser APIs.

Thus, the code that you write in the JavaScript programming language allows you to: 1. Interact with the JavaScript engine inside of the browser 2. Interact with other browser functionality that exists outside of the JavaScript engine, but is still inside the browser.

Although traditionally it was possible to interact with the JavaScript engine only inside of the browser, this all changed in 2009, when Node.js was built by Ryan Dahl.

He came up with a way to use a JavaScript engine as a stand-alone entity. Suddenly, it was possible to use JavaScript outside of the browser, as a separate program on the command line, or as a server-side environment.

Today, JavaScript is ubiquitous and is running in browsers, on servers, actually, on any device that can run a JavaScript engine.