**JavaScript:**

JavaScript (JS) is a lightweight interpreted (or just-in-time compiled) programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles.

Do not confuse JavaScript with the Java programming language — JavaScript is not "Interpreted Java". Both "Java" and "JavaScript" are trademarks or registered trademarks of Oracle in the U.S. and other countries. However, the two programming languages have very different syntax, semantics, and use.

**Why to learn JavaScript?**

JavaScript is an essential programming language for web developers to learn as it grows rapidly. JavaScript is responsible for behavior of the web pages. By learning it, you will develop skills in web development.

**What makes JavaScript unique?**

JavaScript has several unique features that set it apart from other programming languages:

* It can be fully integrated with HTML and CSS;
* It is supported by all major browsers;
* There are numerous frameworks written in Javascript (like React, Angular, Vue, React Native, etc.);
* JavaScript allows creating actions like button clicks, form submissions, and animations.

**Key Features of JavaScript:**

**Versatility**: JavaScript can be used to develop websites, games, mobile apps, and more.

**Client and Server-Side:** With frameworks like Node.js and Express.js, JavaScript is now used for building server-side applications.

**End-to-End Solutions:** JavaScript enables developers to create complete solutions for various problems.

**Constant Evolution:** JavaScript continually evolves with new features and standards.

**Vibrant Community:** A large community of users and mentors actively contributes to JavaScript’s growth.

**History and evolution of JavaScript**

The history of JavaScript started with Netscape Communications Corporation – an American independent computer services company. In 1994, the company dominated the browser market with its Netscape Navigator web browser. But in late 1995, Microsoft entered the arena with Internet Explorer. They tried to take control of the emerging technology from Netscape, so Microsoft became a threat to the company. Easy to guess, Netscape lost to it and other competitors in the first browser war, but the company succeeded in another field.

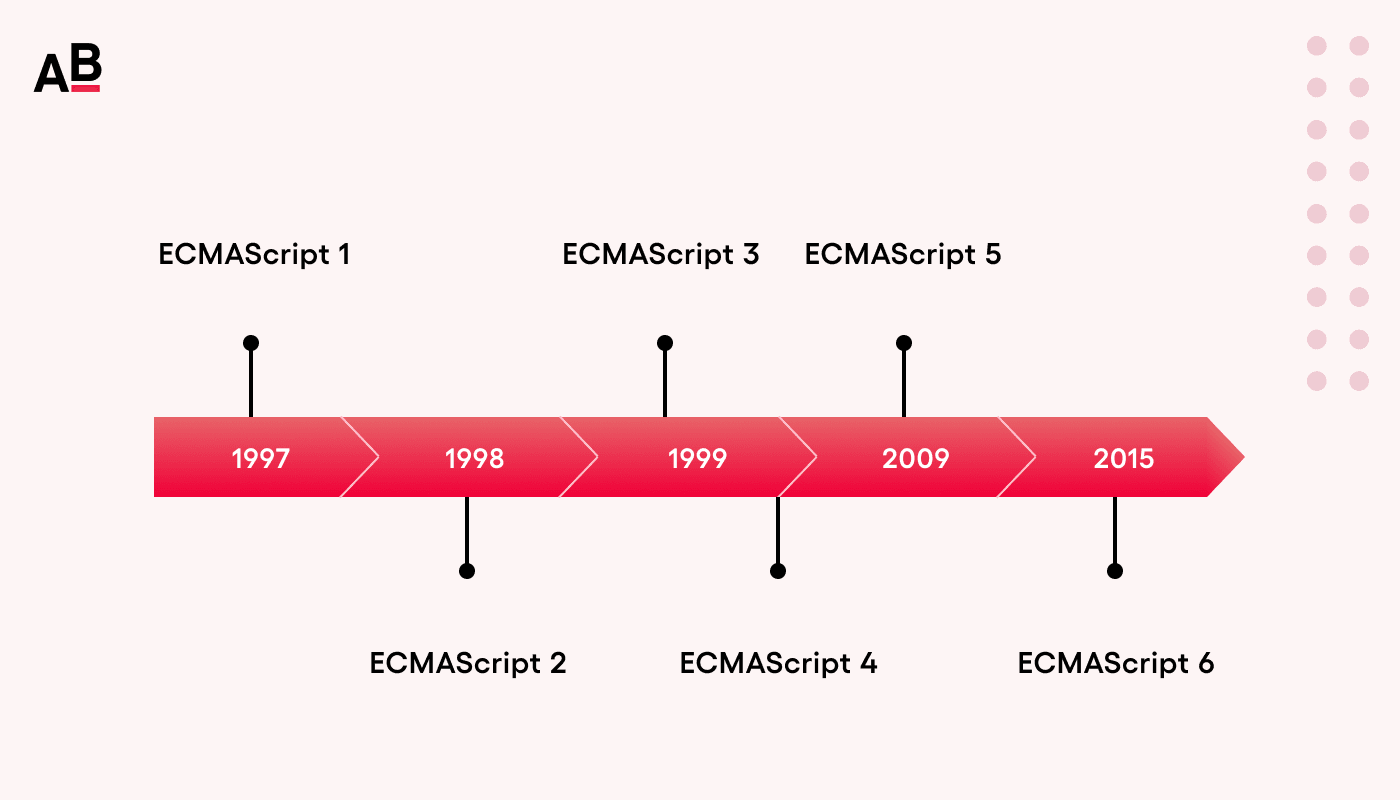
To remain competitive, Netscape sought to expand its audience by incorporating a scripting language that would enable interactions with the Document Object Model (DOM). Marc Andreessen, the founder of Netscape Communications, had the vision that the web needed a way to become more dynamic. At the time, the web was very limited and slow as people used dial-up connections through telephone lines and modems. When filling out forms on websites, validation for input fields had to be done through a full round trip to the server. This was a slow process and could result in data loss if the form wasn’t filled out correctly. To address this issue, Netscape Communications came up with a solution. They decided to enable validation of input fields on the client side to avoid the need for a full round trip to the server. This process required a small scripting language that would be available for both software engineers and people without extensive programming experience.

Netscape Communications then allied with Sun Microsystems. To outdo Microsoft’s Internet Explorer, their collaborative plan involved incorporating Sun’s Java programming language into the Netscape Navigator 2 browser. Thus, in April 1995, Netscape Communications hired Brendan Eich to develop and introduce a new scripting language for the browser. Eich had only 10 days to create a prototype yet he nailed the challenge.

The new language was initially called Mocha and it was designed to resemble Java, which was important to both Netscape and Sun. To achieve this, Eich utilized syntax influenced by C and implemented naming conventions similar to Java. He incorporated the prototype-based object model from the Self language, allowing objects to inherit directly from other objects. He also added first-class functions similar to those found in Lisp. This approach treated functions as variables, resulting in a language that combined the syntax of Java, the object orientation of Self, and the functional style of Scheme.

Near the release, the name was changed from Mocha to LiveScript, and later to JavaScript. It was mainly a marketing tactic to portray the new language as a supporting scripting language to Java. Thus, simple, dynamic, and accessible to non-developers, a new scripting language for the web was released in September 1995.

Initially, JavaScript was not well-received. Skilled Java developers believed that the language was too basic and not up to their standards, while designers and authors found it too challenging to work with. However, JavaScript changed the way the public saw the internet and gave web designers the possibility to make creative and interactive web page designs. And for a while, JavaScript helped Netscape Navigator maintain its position as a market leader.



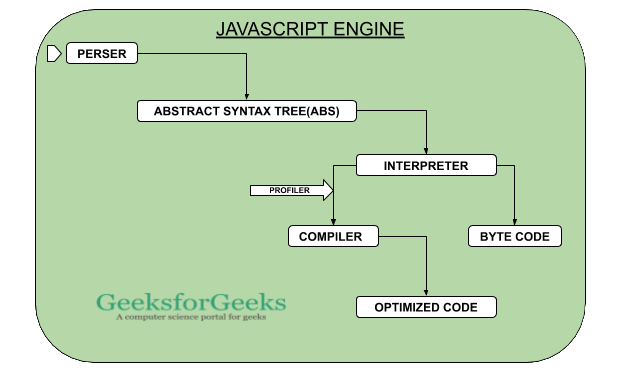
**JS Terms:** From currying to closures there are quite a number of special words used in JavaScript. These will not only help you increase your vocabulary but also better understand JavaScript. Special terms are normally found in documentation and technical articles. But some of them like closures are pretty standard things to know about. Knowing what the word itself means can help you know the concept it's named for better.

1. **Arity:** Arity (from Latin) is the term used to refer to the number of arguments or operands in a function or operation, respectively. You're most likely to come across this word in the realm of JavaScript when it's used to mention the “number of arguments expected by a JavaScript function”.
2. **Anonymous**: When something or someone is referred to as anonymous it means that thing's or person's name is unidentified. Likewise in JavaScript an anonymous function is the one that isn't identified by a name.
3. **Closure**: In JavaScript, closure is an inner function, that's accessible outside of its outer function's scope, with its connection to the outer function's variables still intact. They are able to take our parameters and bring us back variables (references to those variables, to be precise) from inside a function that we aren't allowed in.
4. **Currying**: The effect, named after Haskell Curry, refers to “using multiple functions with single arguments”, in place of a single function with multiple arguments.
5. **Hoisting**: Hoist means to raise something. Hoisting in JavaScript also means the same and what gets raised is the declaration (variable & function declarations). Declarations are where variables and functions are created with keywords var (not for global) and function.
6. **Mutation**: Mutation means change or modification. If you ever come across the word mutation in JavaScript it's probably referring to the changes that DOM elements went through. There's even an API called MutationObserver to keep an eye out for the DOM mutations like “addition of child elements or changes to the element's attributes.”
7. **Pragma**: Pragma is short for pragmatic information. In plain English, pragmatic is an adjective that means sensible and practical. In programming, pragma refers to the code that consist of useful information on “how a compiler or interpreter or assembler should process the program.”
8. **Sentinel**: Sentinels are soldiers who stand guard (Remember the ones from X-Men?). In programming, sentinels are values that are used to indicate the end of a loop or process. They can also be called "flags".
9. **Vanilla**: Vanilla is considered a traditional standard flavor. Vanilla JavaScript is referred to the standard JavaScript — no framework. Vanilla in fact isn't only used to describe the standard version of JavaScript but also other languages like CSS.
10. **Variadic**: Variadic is an adjective created by joining "variable" and "adicity". "Adicity" is from ancient Greek, with a meaning that's the same as the Latin word "arity" (Item 1 in this list). Thus, the term variadic is used to “express something that has variable number of arguments.”

**JavaScript Engines:**

JavaScript is a scripting language and is not directly understood by computer but the browsers have inbuilt JavaScript engine which help them to understand and interpret JavaScript codes. These engines help to convert our JavaScript program into computer-understandable language.

A JavaScript engine is a computer program that executes JavaScript code and converts it into computer understandable language.



**List of JavaScript Engines:**

|  |  |
| --- | --- |
| Browser | Name of Javascript Engine |
| Google Chrome | V8 |
| Edge (Internet Explorer) | Chakra |
| Mozilla Firefox | Spider Monkey |
| Safari | Javascript Core Webkit |

**JavaScript runtime environment:**

The JavaScript runtime environment provides access to built-in libraries and objects that are available to a program so that it can interact with the outside world and make the code work.

In the context of a browser this is comprised of the following elements:

The JavaScript engine (which in turn is made up of the heap and the call stack)

1. Web APIs
2. The callback queue
3. The event loop

It’s good to be mindful that the runtime environment can take on different forms based on the context, for example, the runtime environment in a browser is very different from that of Node.js. These differences are primarily at the implementation level, so most of the following concepts are still relevant.

**1. The JavaScript engine:** To start writing JavaScript we don’t need to install any specific software because each web browser has its own version of the JS engine that parses the code for us. Chrome uses the V8 JS engine which has been developed by the Chromium Project. Firefox uses SpiderMonkey which was first written by Brendan Eich at Netscape and is now maintained by the folks at Mozilla.

The purpose of the JavaScript engine is to translate source code that developers write into machine code that allows a computer to perform specific tasks.

**2. The heap:** The heap, also called the ‘memory heap’, is a section of unstructured memory that is used for the allocation of objects and variables.

**3. The call stack:** The call stack is a data structure that keeps track of where we are in the program and runs in a last-in, first-out way. Each entry in the stack is called a stack frame. This means that the frame at the top of the stack is the one the engine is focused on, and it will not move on to the next function unless the function above it has been removed from the stack.

As the JS engine steps into a function, it is pushed onto the stack. When a function returns a value or gets sent to the Web APIs, it is popped off the stack. If a function doesn’t explicitly return a value then the engine will return undefined and also pop the function off the stack. This is what is meant by the term “JavaScript runs synchronously”; it is single-threaded, so can only do one thing at a time.

**4. Web APIs:** The Web APIs are not a part of the JavaScript engine, but they are part of the runtime environment provided by the browser. There are a large number of APIs available in modern browsers that allow us to a wide variety of things. Some of the most common categories of browser APIs let us:

Manipulate documents: One of the most common Web APIs used is the DOM API, which allows developers to manipulate HTML and CSS, letting us create, change and even remove HTML and dynamically apply styles to our web pages.

Draw and manipulate graphics: Widely supported in browsers, the Canvas API and the Web Graphics Library API let us programmatically update pixel data contained in a <canvas> element.

Fetch data from a server: The Fetch API provides an interface for fetching resources across the network by using generic definitions of the Request and Response objects.

Features like event listeners, timing functions and AJAX requests all sit in the Web APIs container until an action gets triggered. A request finishes receiving its data, a timer reaches its set time or a click happens and this triggers a callback function to be sent to the callback queue.

**5. The callback queue:** The callback queue stores the callback functions sent from the Web APIs in the order in which they were added. This queue is a data structure that runs first in, first out. The queue uses the array push method to add a new callback function to the end of the queue and the array shift method to remove the first item in the queue.

Callback functions will sit in the queue until the call stack is empty, they are then moved into the stack by the event loop.

**6. The event loop:** The job of the event loop is to constantly monitor the state of the call stack and the callback queue. If the stack is empty it will grab a callback from the callback queue and put it onto the call stack, scheduling it for execution.

This is why JavaScript often gets described as being able to run asynchronously, even though it is a single-threaded language. JavaScript can only execute one function at a time, so this means it is synchronous, but as we can push callbacks from the Web APIs to the callback queue and in turn, the event loop can constantly add those callback to the call stack, we think of JavaScript as being able to run asynchronously.

