perplexity

who made javascript and in 10 days story, what was netscape first browser, conflict bw microsoft, netscape over browser and javascript and jscript, did netscape go to echma, why vs code doesnt understand js, why code doesnt run when we type node script.js in terminal, and what does node mean, how chrome understand, it has v8 engine smth, is it jit compiler , whats js engine, also give history how firefox came then with spider monke engine, netscape got close due to microsoft became open source, then chrome came, what lang are these engines written in, why vs code doesnt have predefined js engine, also how nodejs came, and how if we install nodejs we can run js anywhere, and how it came to be used as a backend language, give all answers in details and with additional info

Here is a detailed breakdown of all your questions, capturing the technical, historical, and industry-shaping aspects of JavaScript, browser engines, Node.js, and more:

Who Made JavaScript and the "10 Days" Story

Brendan Eich invented JavaScript in **May 1995** while at Netscape. Eich was hired to add a scripting language for web pages, and working under intense competition, he designed and implemented JavaScript's first version in **just 10 days**. The language was first called "Mocha," then "LiveScript," and finally "JavaScript," with the chosen name benefitting from the popularity of Java (though the languages are very different)[1] [2] [3] [4]. Eich's rapid design enabled a new era of web interactivity and kicked off massive changes in the web's direction.

Netscape's First Browser

Netscape's groundbreaking product was **Netscape Navigator**, released in late 1994. It quickly dominated the early web—making the Internet accessible to millions and setting the stage for JavaScript's inclusion[5] [6] [7].

Microsoft vs Netscape: Browser Wars, JavaScript, and JScript

With Navigator's popularity rising, Microsoft entered the arena with **Internet Explorer (IE)**, igniting the infamous "Browser Wars"[8] [9] [10]. Microsoft reverse-engineered JavaScript, releasing its own nonstandard variant called **JScript** for IE[11] [12] [13]. Differences between JavaScript and JScript, plus DOM incompatibility, forced web developers to write browser-specific code, causing "Best viewed in ..." website disclaimers and years of frustration for users and developers.

Microsoft's aggressive business tactics—including bundling IE with Windows, exclusive OEM agreements, and rapid development cycles—eventually forced Netscape out of dominance. By 2001, IE had overtaken the majority of the browser market, and Netscape's influence waned drastically[14] [15].

Netscape, ECMA, and the ECMAScript Standard

To resolve fragmentation between JavaScript and JScript, Netscape submitted its scripting language to **Ecma International**, resulting in the **ECMA-262 specification** and the creation of the **ECMAScript** standard. This ensured all browsers would eventually support a single, well-defined scripting language core[4] [16] [17] [18]. "ECMAScript" refers to the standard, while "JavaScript" and "JScript" are implementations.

Why VS Code Doesn't "Understand" JavaScript Out of the Box

VS Code is a flexible **text/code editor**—not a browser or a JavaScript engine[19] [20] [21]. JavaScript support in VS Code comes from built-in language services and extensions (like IntelliSense, syntax highlighting, and error checking), not from native JS execution. To "run" JS code, you need a runtime like Node.js, or a browser environment—VS Code itself cannot execute or "understand" JS at runtime without these tools[22] [23] [24].

Why `node script.js` Might Fail

- **Wrong Directory**: Be sure your terminal is in the folder with your script[25] [26].
- **Inside Node.js REPL**: If the prompt is `>`, you're inside the Node shell. Exit with `Ctrl+C` before running `node script.js`.
- **No Output**: If there's no `console.log`, the program can appear to do nothing.
- **Permissions/file issues**: In rare cases, system or config problems prevent execution.

Running JavaScript from the command line requires having **Node.js** installed; then you can execute:

```
node script.js
```

What Does "Node" Mean in Node.js?

Node.js is an open-source JavaScript runtime that allows you to run JS outside the browser—on servers, desktops, and more[27] [28] [29] [30]. It uses Chrome's V8 engine for execution and provides additional APIs to interact with files, networking, and OS resources.

How Chrome Runs JavaScript: V8 Engine, JIT Compilation, and JS Engines

Chrome runs JavaScript using the **V8 engine**, a high-performance JS engine developed by Google[31] [32] [33] [34] [35]. Here's how it works:

- **Parsing:** Converts code into an Abstract Syntax Tree (AST)[33] [34].
- **Ignition Interpreter:** Compiles the AST into lightweight bytecode.
- **JIT Compiler (TurboFan, Maglev):** Frequently executed code ("hot" functions) are translated Just-In-Time (JIT) into optimized machine code for maximum speed[32] [33] [36].
- **Hidden Classes & Inline Caching:** Advanced optimizations further speed up property accesses and calls.
- **Garbage Collection:** Manages memory automatically and efficiently.

Is V8 a JIT Compiler?
Yes: V8 interprets then compiles JavaScript to fast native code at runtime[36].

What Is a JS Engine?
A JavaScript engine parses, interprets (and usually compiles) JavaScript source code into machine-level instructions the computer can execute. Different browsers use different engines:

- **V8 (Chrome, Edge, Node.js)**
- **SpiderMonkey (Mozilla Firefox)**
- **JavaScriptCore/Nitro (Safari)**

Firefox: Its Arrival and the SpiderMonkey Engine

After Microsoft's dominance caused Netscape's decline, Netscape open-sourced its browser code, creating the **Mozilla project** and later **Mozilla Firefox** (2004)[5] [10]. Firefox uses the **SpiderMonkey JS engine** (also originally created by Brendan Eich in 1995; since evolved)—the world's first JavaScript engine.

Safari: JavaScriptCore (Nitro) and "Monkey" Engines

Apple's **Safari** arrived in 2003, using its own JS engine called **JavaScriptCore** or "Nitro." Apple introduced major JIT improvements and kept up with evolving standards. (The silly "monkey"-related names in browsers—SpiderMonkey, SquirrelFish—became a long-running tradition.)

Netscape: From Closure to Open Source

Losing the browser war, **Netscape open-sourced its code**, which became Mozilla. This move essentially created an open-source browser ecosystem, paving the way for Firefox's emergence and contributing to the culture that later enabled Chrome's swift adoption.

Chrome Launch and JavaScript's Reinvention

In **2008**, Google launched **Chrome** with the blazing-fast V8 engine. Chrome's performance forced others (Firefox, Safari, even IE/Edge) to radically update their engines. Chrome's architecture, V8, and multi-process design made the web faster, more secure, and more standards-compliant[33] [35].

What Languages Are JS Engines Written In?

- **V8:** Written in **C++**.
- **SpiderMonkey:** Primarily **C++**, with some **Rust**.
- **JavaScriptCore/Nitro: C++**.
  All are written in compiled languages for maximum speed and low-level memory control[33] [34].

Why VS Code Doesn't Ship With a Built-In JS Engine

VS Code is designed to be a **lightweight, flexible code editor**. Shipping a full JavaScript engine (like V8) inside VS Code would make the editor much heavier and duplicate what browsers/Node.js already provide. Instead, VS Code leverages Node.js for integrated terminals, debugging, and script running[19] [20].

How Node.js Came To Be and Turned JS Into a Backend Language

Node.js was created by **Ryan Dahl in 2009**, inspired by the need for fast, scalable, and non-blocking servers[30] [28]. Using Chrome's V8 engine, Node.js allowed developers to:

- Run JavaScript anywhere—not just in browsers.
- Build web servers, APIs, network apps, and command-line tools in JS.
- Achieve fast, scalable I/O via Node's event-driven, non-blocking architecture.

With npm (Node Package Manager), a massive ecosystem of reusable libraries flourished, helping JavaScript become one of the world's most popular backend languages.

If We Install Node.js, How Can We Run JS Anywhere?

Once Node.js is installed on your computer:

- You can run **any JavaScript file** using `node filename.js`—no browser needed.
- Node exposes important modules and APIs for file system, networking, databases, and more.

- Developers run build tools, scripts, and even desktop apps (via Electron) all powered by Node.js.

How JavaScript Rose as a Full-Stack Language

- **Front-end**: Originally just for making web pages interactive in browsers.
- **Back-end**: With Node.js, the same language could be used to build servers, APIs, and scalable network services.
- **Full Stack**: One language now powers both client and server, streamlining development and lowering the barrier for new programmers.

Timeline Table

Event/Term	Details & Significance
JavaScript	Brendan Eich, 10 days at Netscape, 1995
Netscape	Navigator, 1994; created web for masses, added JS
Browser Wars	Netscape vs Microsoft (IE), led to JScript, compatibility chaos
ECMAScript	Netscape to ECMA (1996), created the ECMAScript standard
VS Code	Editor only; needs Node.js or browser to run JS
Node.js	Ryan Dahl, 2009, uses V8 to run JS on servers, desktops, everywhere
JS Engines	V8 (C++), SpiderMonkey (C++/Rust), JavaScriptCore (C++); all compile/interp code
JIT	V8, Nitro, others use JIT: compile JS to fast machine code at runtime
Firefox	Mozilla from Netscape code, powered by SpiderMonkey
Safari	Apple’s browser, uses JavaScriptCore ("Nitro") engine
Chrome	Google, 2008, V8 engine; revolutionized JS performance, forced industry upgrades
Node install	Lets you run any JS file; JS became core backend/server language

In summary:
JavaScript began as a rushed project for browser scripting at Netscape, influenced by competitive pressure from Microsoft. After intense "Browser Wars" and the rise of IE, ECMA standardization unified the language. The demise of Netscape gave rise to an open-source revolution, leading to Firefox and Safari. Google Chrome and the V8 engine revolutionized speed and standards, while Node.js expanded JS to every device and the backend. VS Code is a great editor but leaves actual execution to dedicated engines—lightweight by design. All of this made JavaScript the world’s first truly universal programming language.

⁂

1. https://en.wikipedia.org/wiki/Brendan_Eich
2. https://cybercultural.com/p/1995-the-birth-of-javascript/
3. https://www.geeksforgeeks.org/javascript/history-of-javascript/

4. https://www.w3schools.com/js/js_history.asp
5. https://en.wikipedia.org/wiki/Netscape_Navigator
6. https://en.wikipedia.org/wiki/Netscape_(web_browser)
7. https://www.webdesignmuseum.org/software/netscape-navigator-1-0-in-1994
8. https://en.wikipedia.org/wiki/Browser_wars
9. https://www.investopedia.com/ask/answers/09/browser-wars-netscape-internet-explorer.asp
10. https://www.linkedin.com/pulse/netscape-microsoft-browser-wars-story-unethical-tactics-míra-podorský
11. https://code.zeba.academy/javascript-vs-jscript-ultimate-comparison/
12. https://csveda.com/jscript-versus-javascript-the-key-differences/
13. https://en.wikipedia.org/wiki/JScript
14. https://www.zdnet.com/home-and-office/networking/how-ie-really-beat-netscape/
15. https://www.reddit.com/r/todayilearned/comments/13i7hdv/til_about_the_first_browser_war_which_took_place/
16. https://en.wikipedia.org/wiki/ECMAScript
17. https://tc39.es/ecma262/
18. https://262.ecma-international.org
19. https://code.visualstudio.com/docs/nodejs/working-with-javascript
20. https://code.visualstudio.com/docs/languages/javascript
21. https://code.visualstudio.com/docs/editing/intellisense
22. https://stackoverflow.com/questions/36235242/visual-studio-code-intellisense-not-working-for-javascript
23. https://www.reddit.com/r/vscode/comments/114cv74/intellisense_not_working_for_javascripttypescript/
24. https://github.com/microsoft/vscode/issues/161570
25. https://stackoverflow.com/questions/70357454/why-node-js-cannot-run-my-javascript-code-in-the-terminal
26. https://github.com/nodejs/help/issues/2935
27. https://www.geeksforgeeks.org/node-js/what-is-node/
28. https://www.techtarget.com/whatis/definition/Nodejs
29. https://www.codecademy.com/article/what-is-node
30. https://en.wikipedia.org/wiki/Node.js
31. https://nodejs.org/en/learn/getting-started/the-v8-javascript-engine
32. https://www.w3schools.com/nodejs/nodejs_v8_engine.asp
33. https://en.wikipedia.org/wiki/V8_(JavaScript_engine)
34. https://www.geeksforgeeks.org/explain-v8-engine-in-node-js/
35. https://www.cloudflare.com/it-it/learning/serverless/glossary/what-is-chrome-v8/
36. https://v8.dev/blog/maglev