**Javascript**

JavaScript is a High level programming language that is primarily used to enhance the interactivity and dynamic behavior of websites.

JavaScript is a lightweight, cross-platform, single-threaded, and High level interpreted compiled programming language. It is also known as the scripting language for webpages.

### .**Lightweight**

* **Meaning:** JavaScript is considered lightweight because it has a simple syntax and doesn't require a large amount of system resources to run.

### **Cross-Platform**

* **Meaning:** JavaScript is cross-platform, meaning it can run on various operating systems and devices. Whether it's a Windows PC, a Mac, a Linux system, or a mobile device, JavaScript code can be executed as long as the environment has a JavaScript engine, such as web browsers (Chrome, Firefox, Safari)

### **Single-Threaded**

* **Meaning:** JavaScript operates on a single thread, meaning it can perform one operation at a time per thread. This is often referred to as the "JavaScript event loop," where JavaScript handles operations sequentially. While JavaScript is single-threaded, it can handle asynchronous tasks (like network requests, timers, etc.) using callbacks, promises, and the async/await syntax, allowing it to perform tasks without blocking the main thread.

### **High-Level and Easy to Write**

* Scripting languages are usually high-level, meaning they are closer to human language and easier to read, write, and understand. They are often used to write code quickly and efficiently without needing complex syntax or understanding the underlying hardware.

### **Interpreted Compiled**

* **Meaning:** JavaScript is often referred to as an interpreted language because traditionally, it was executed directly by the browser without prior compilation. modern JavaScript engines (like V8 in Chrome) use Just-In-Time (JIT) compilation, which compiles JavaScript code to machine code at runtime, improving performance. Thus, it's both interpreted (in a traditional sense) and compiled (due to JIT).

### **Scripting Language for Webpages**

* **Meaning:** JavaScript was initially created as a scripting language for the web. Its primary use was to add interactivity to HTML pages. Over time, it has evolved significantly but remains the core technology for client-side web development.

**The History of JavaScript**

**1. Origins and Creation (1995)**

* **Who Introduced JavaScript:** JavaScript was created by Brendan Eich, a programmer working at Netscape Communications Corporation.
* **Development Timeline:** Brendan Eich developed the first version of JavaScript in just 10 days in May 1995.
* **Company:** Netscape, the company that hired Eich, was a pioneer in the early web browser market. At that time, Netscape was in direct competition with Microsoft to dominate the browser space.

**2. Early Naming and Initial Implementation**

* **Initial Name - Mocha:** The language was initially called *Mocha*. This name was chosen by Marc Andreessen, one of the co-founders of Netscape.
* **Renaming to LiveScript:** Before its official launch, the language was renamed *LiveScript*. This name was intended to emphasize its dynamic capabilities in web pages, distinguishing it from static HTML.
* **Final Renaming to JavaScript:** In December 1995, Netscape changed the language’s name to *JavaScript*. This was part of a marketing collaboration with Sun Microsystems. The name JavaScript was chosen to leverage the popularity of Java at the time, even though JavaScript and Java are fundamentally different languages.

**3. Relationship with Java and the Role of Sun Microsystems**

* **Java vs. JavaScript:** Despite the name similarity, Java and JavaScript serve different purposes and have distinct syntaxes. Java is a statically typed, compiled language, whereas JavaScript is a dynamically typed, interpreted language.
* **Sun Microsystems’ Involvement:** Sun Microsystems, the creators of Java, and Netscape agreed to rename LiveScript to JavaScript to associate the new scripting language with Java. This move helped market JavaScript as a complementary technology to Java, which was popular among developers at that time.

**4. Microsoft's Response: Introduction of JScript**

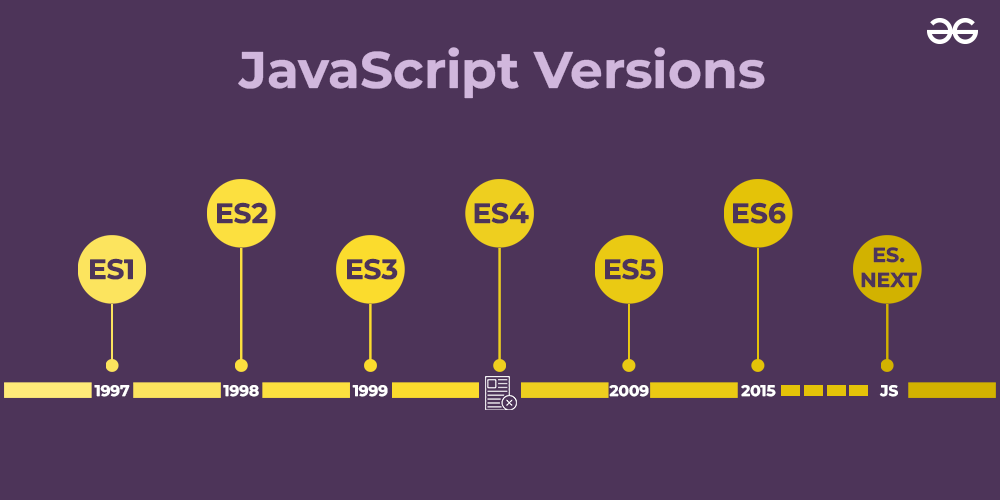
* **Launch of JScript:** In response to JavaScript’s growing popularity, Microsoft developed its own implementation of JavaScript called *JScript* in 1996.
* **Compatibility:** JScript was designed to be compatible with JavaScript but included certain proprietary features and enhancements. Microsoft integrated JScript into Internet Explorer 3.0, their web browser, to compete with Netscape Navigator.
* **Browser Wars:** This period marked the beginning of the first browser wars, with Netscape and Microsoft vying for dominance in the web browser market.

**5. Standardization: ECMAScript**

* **Why Standardization was Necessary:** The existence of different implementations (JavaScript by Netscape and JScript by Microsoft) created inconsistencies and compatibility issues across browsers.
* **Submission to ECMA:** In November 1996, Netscape submitted JavaScript to the European Computer Manufacturers Association (ECMA) for standardization.
* **Creation of ECMAScript:** In June 1997, ECMA published the first edition of ECMAScript, a standardized specification for JavaScript. ECMAScript provided a standard that all scripting languages like JavaScript and JScript could conform to, ensuring greater compatibility and consistency across different web browsers.
* **Role of ECMA-262:** ECMAScript is often referred to as ECMA-262, which is the name of the standard specification document. All major browsers began aligning their JavaScript engines to follow the ECMAScript standard.

**6. Evolution of JavaScript and ECMAScript Versions**

* **ECMAScript 2 and 3:** After the first edition, ECMAScript 2 (June 1998) and ECMAScript 3 (December 1999) were released, with ES3 becoming widely implemented and serving as the base for JavaScript development for many years.
* **ECMAScript 4:** There was an attempt to create ECMAScript 4 with major enhancements, but it was abandoned due to disagreements within the industry.
* **ECMAScript 5 (ES5):** Released in 2009, ES5 included important features such as strict mode, JSON support, and more. ES5 was a significant improvement over ES3 and was widely adopted.
* **ECMAScript 6 (ES6/ECMAScript 2015):** Released in 2015, ES6 introduced major enhancements, including arrow functions, classes, modules, template literals, let and const keywords, promises, and many more. ES6 marked a major evolution in JavaScript, making it more powerful and easier to work with.



**7. Modern JavaScript: ECMAScript Evolution and Community Involvement**

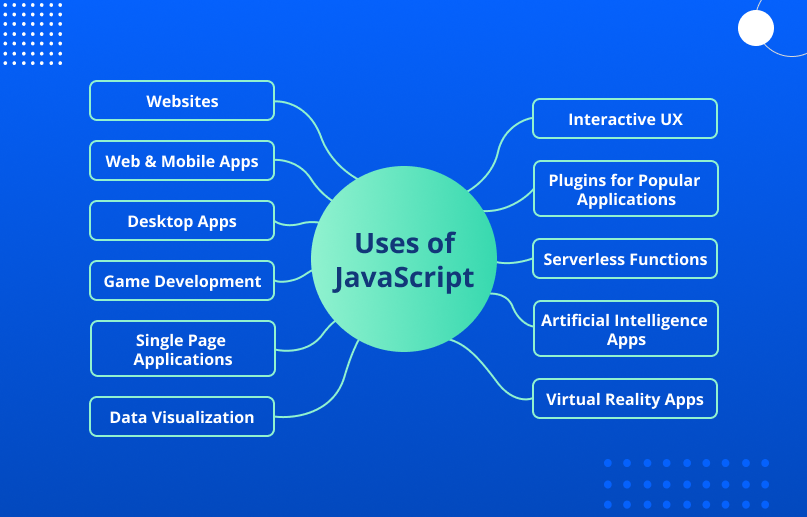
* **Annual Release Cycle:** Since ECMAScript 2015, the TC39 committee, which oversees the evolution of ECMAScript, adopted an annual release cycle. Each year, a new version of ECMAScript is released with incremental updates and new features.
* **ES7 (ES2016), ES8 (ES2017), ES9 (ES2018), etc.:** These versions introduced features like the exponentiation operator, async/await, Object.values/Object.entries, and more.
* **Community and Open-Source Influence:** The JavaScript community, including developers and companies, actively contributes to the evolution of JavaScript through proposals, feedback, and implementation in various environments (browsers, Node.js, etc.).

**8. JavaScript's Role Today**

* **Ubiquity:** JavaScript is a core technology of the World Wide Web, alongside HTML and CSS. It runs on all modern web browsers and is essential for creating interactive web applications.
* **Beyond the Browser:** JavaScript’s usage has extended beyond the browser, thanks to environments like Node.js. It is used for server-side programming, mobile app development (e.g., React Native), desktop apps (e.g., Electron), and more.
* **Frameworks and Libraries:** JavaScript has a rich ecosystem of frameworks and libraries like React, Angular, Vue, and jQuery, which make it easier to build complex and responsive web applications.

Some of the common uses of JavaScript are:

1. **Web Development:** JavaScript is primarily used for creating interactive and dynamic web pages and web applications. It can be used for tasks such as form validation, animations, user interface enhancements, and more.
2. **Mobile App Development:** JavaScript can be used to create mobile applications that can run on both iOS and Android devices using frameworks such as React Native and Ionic.
3. **Game Development:** JavaScript is used to create web-based games, such as HTML5 games, that can be played on any device with a web browser.
4. **Server-Side Development:** JavaScript can be used to create server-side applications using Node.js, which is a platform that allows developers to run JavaScript on the server side.
5. **Desktop Application Development:** JavaScript can be used to create desktop applications using frameworks such as Electron, which allows developers to build cross-platform desktop applications using web technologies.
6. **Internet of Things (IoT):** JavaScript can be used to program IoT devices, such as sensors and smart home devices, using frameworks such as Johnny-Five and Cylon.js.



## JavaScript: the Swiss army knife trap | by Tiago Sansão | Medium