ADDING CSS CLASSES

Using **ClassList**, CSS classes can be added or removed. See the code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <style>
        .main { color: red; }
    </style>
</head>
<body>
    <h1 id="item">Black-Red Text</h1>
</body>
</html>
```

```
// Gets the <h1>
const header = document.querySelector( `h1` )
header.classList.add(`main`)
// header.classList.remove (`main`)
```

Or you can use the following JS:

```
// Gets the <h1>
const header = document.querySelector( `h1` )
header.classList +='main'
// header.classList.remove (`main`)
```

You can also use **className**, but pay attention that using "**classList**", you can add or remove a class without affecting any others the element may have. But if you assign "**className**", it will

wipe out any existing classes while adding the new one (or if you assign an empty string it will wipe out all of them).

Using **classList** is the recommended technique to add, remove, or toggle classes. To completely replace the class string, you would use **className**.

Now that you know much about manipulating both contents and style of the DOM, let's do the following exercises:

1- Can you create a collapsible:

ClassName vs ClassList

When clicked, it will open the content:

ClassName vs ClassList

Using "classList", you can add or remove a class without affecting any others the element may have. But if you assign "className", it will wipe out any existing classes while adding the new one (or if you assign an empty string it will wipe out all of them).

https://www.w3schools.com/howto/howto_js_collapsible.asp

2- Now Create a Sticky Header!
   a. Use the following for Header:
      "History of JavaScript"
   b. Use the following long content:

From Wikipedia, the free encyclopedia
Not to be confused with Java (programming language), Javanese script, or ECMAScript.
".js" redirects here. For the Microsoft dialect used in Internet Explorer, see JScript.
For the uses of JavaScript on Wikipedia, see Wikipedia:WikiProject JavaScript.
JavaScript

Screenshot of JavaScript source code
Paradigm Multi-paradigm: event-driven, functional, imperative, procedural, object-oriented

Designed by Brendan Eich of Netscape initially; others have also contributed to the ECMAScript standard
First appeared December 4, 1995; 28 years ago[1]
Stable release
ECMAScript 2023[2] Edit this on Wikidata / June 2023; 10 months ago
Preview release
ECMAScript 2025[3] Edit this on Wikidata / 28 February 2024; 35 days ago
Typing discipline Dynamic, weak, duck
Filename extensions
.js.cjs.mjs[4]
Website ecma-international.org/publications-and-standards/standards/ecma-262/
Major implementations
V8, JavaScriptCore, SpiderMonkey, Chakra
Influenced by
Java,[5][6] Scheme,[6] Self,[7] AWK,[8] HyperTalk[9]
Influenced
ActionScript, ArkTS, AssemblyScript, CoffeeScript, Dart, Haxe, JS++, Opa, TypeScript
JavaScript at Wikibooks

JavaScript (/ˈdʒɑːvəskrɪpt/), often abbreviated as JS, is a programming language and core technology of the Web, alongside HTML and CSS. 99% of websites use JavaScript on the client side for webpage behavior.[10]

Web browsers have a dedicated JavaScript engine that executes the client code. These engines are also utilized in some servers and a variety of apps. The most popular runtime system for non-browser usage is Node.js.

JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard.[11] It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).

The ECMAScript standard does not include any input/output (I/O), such as networking, storage, or graphics facilities. In practice, the web browser or other runtime system provides JavaScript APIs for I/O.

Although Java and JavaScript are similar in name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design.

History
Creation at Netscape

The first popular web browser with a graphical user interface, Mosaic, was released in 1993. Accessible to non-technical people, it played a prominent role in the rapid growth of the early World Wide Web.[12] The lead developers of Mosaic then founded the Netscape corporation, which released a more polished browser, Netscape Navigator, in 1994. This quickly became the most-used.[13]

During these formative years of the Web, web pages could only be static, lacking the capability for dynamic behavior after the page was loaded in the browser. There was a desire in the flourishing web development scene to remove this limitation, so in 1995, Netscape decided to add a programming language to Navigator. They pursued two routes to achieve this: collaborating with Sun Microsystems to embed the Java language, while also hiring Brendan Eich to embed the Scheme language.[6]

The goal was a "language for the masses",[14] "to help nonprogrammers create dynamic, interactive Web sites".[15] Netscape management soon decided that the best option was for Eich to devise a new language, with syntax similar to Java and less like Scheme or other extant scripting languages.[5][6] Although the new language and its interpreter implementation were called LiveScript when first shipped as part of a Navigator beta in September 1995, the name was changed to JavaScript for the official release in December.[6][1][16]

The choice of the JavaScript name has caused confusion, implying that it is directly related to Java. At the time, the dot-com boom had begun and Java was a popular new language, so Eich considered the JavaScript name a marketing ploy by Netscape.[14]

Adoption by Microsoft

Microsoft debuted Internet Explorer in 1995, leading to a browser war with Netscape. On the JavaScript front, Microsoft created its own interpreter called JScript.[17]

Microsoft first released JScript in 1996, alongside initial support for CSS and extensions to HTML. Each of these implementations was noticeably different from their counterparts in Netscape Navigator.[18][19] These differences made it difficult for developers to make their websites work well in both browsers, leading to widespread use of "best viewed in Netscape" and "best viewed in Internet Explorer" logos for several years.[18][20]

The rise of JScript

Brendan Eich later said of this period: "It's still kind of a sidekick language. It's considered slow or annoying. People do pop-ups or those scrolling messages in the old status bar at the bottom of your old browser."[14]

In November 1996, Netscape submitted JavaScript to Ecma International, as the starting point for a standard specification that all browser vendors could conform to. This led to the official release of the first ECMAScript language specification in June 1997.

The standards process continued for a few years, with the release of ECMAScript 2 in June 1998 and ECMAScript 3 in December 1999. Work on ECMAScript 4 began in 2000.[17]

However, the effort to fully standardize the language was undermined by Microsoft gaining an increasingly dominant position in the browser market. By the early 2000s, Internet Explorer's market share reached 95%.[21] This meant that JScript became the de facto standard for client-side scripting on the Web.

Microsoft initially participated in the standards process and implemented some proposals in its JScript language, but eventually it stopped collaborating on ECMA work. Thus ECMAScript 4 was mothballed.

Growth and standardization
During the period of Internet Explorer dominance in the early 2000s, client-side scripting was stagnant. This started to change in 2004, when the successor of Netscape, Mozilla, released the Firefox browser. Firefox was well received by many, taking significant market share from Internet Explorer.[22]

In 2005, Mozilla joined ECMA International, and work started on the ECMAScript for XML (E4X) standard. This led to Mozilla working jointly with Macromedia (later acquired by Adobe Systems), who were implementing E4X in their ActionScript 3 language, which was based on an ECMAScript 4 draft. The goal became standardizing ActionScript 3 as the new ECMAScript 4. To this end, Adobe Systems released the Tamarin implementation as an open source project. However, Tamarin and ActionScript 3 were too different from established client-side scripting, and without cooperation from Microsoft, ECMAScript 4 never reached fruition.

Meanwhile, very important developments were occurring in open-source communities not affiliated with ECMA work. In 2005, Jesse James Garrett released a white paper in which he coined the term Ajax and described a set of technologies, of which JavaScript was the backbone, to create web applications where data can be loaded in the background, avoiding the need for full page reloads. This sparked a renaissance period of JavaScript, spearheaded by open-source libraries and the communities that formed around them. Many new libraries were created, including jQuery, Prototype, Dojo Toolkit, and MooTools.

Google debuted its Chrome browser in 2008, with the V8 JavaScript engine that was faster than its competition.[23][24] The key innovation was just-in-time compilation (JIT),[25] so other browser vendors needed to overhaul their engines for JIT.[26]

In July 2008, these disparate parties came together for a conference in Oslo. This led to the eventual agreement in early 2009 to combine all relevant work and drive the language forward. The result was the ECMAScript 5 standard, released in December 2009.

Reaching maturity
Ambitious work on the language continued for several years, culminating in an extensive collection of additions and refinements being formalized with the publication of ECMAScript 6 in 2015.[27]

The creation of Node.js in 2009 by Ryan Dahl sparked a significant increase in the usage of JavaScript outside of web browsers. Node combines the V8 engine, an event loop, and I/O APIs, thereby providing a stand-alone JavaScript runtime system.[28][29] As of 2018, Node had been used by millions of developers,[30] and npm had the most modules of any package manager in the world.[31]

The ECMAScript draft specification is currently maintained openly on GitHub,[32] and editions are produced via regular annual snapshots.[32] Potential revisions to the language are vetted through a comprehensive proposal process.[33][34] Now, instead of edition numbers, developers check the status of upcoming features individually.[32]

The current JavaScript ecosystem has many libraries and frameworks, established programming practices, and substantial usage of JavaScript outside of web browsers. Plus, with the rise of single-page applications and other JavaScript-heavy websites, several transpilers have been created to aid the development process.[35]

https://www.w3schools.com/howto/howto_js_sticky_header.asp