DWA_02.8 Knowledge Check_DWA2

---

1. What do ES5, ES6 and ES2015 mean - and what are the differences between them?

ES5, ES6, and ES2015 are all terms used to refer to different versions of the ECMAScript standard, which is a specification that defines the scripting language used by JavaScript.

ES5 introduced significant enhancements to JavaScript. Some notable features added in ES5 include strict mode, which enforces stricter rules for writing JavaScript code, as well as new array manipulation methods, JSON support, and improved error handling with try-catch statements. ES5 is widely supported by modern web browsers.

ES6 is the next major update to the ECMAScript standard and was released in 2015. ES6 introduced many new features and syntax enhancements to JavaScript, making it a more powerful and expressive language. Some notable features introduced in ES6 include block-scoped variables with “let” and “const”, arrow functions, template literals, classes, modules, and enhanced object literals. ES6 also introduced new built-in methods for working with arrays and strings, as well as Promises for handling asynchronous operations.

ES2015 is essentially another name for ES6. The term "ES2015" reflects the year in which the standard was finalized. After ES6, the ECMAScript committee decided to adopt a new naming convention based on the year of the release, so ES2015 is synonymous with ES6.

---

2. What are JScript, ActionScript and ECMAScript - and how do they relate to JavaScript?

JScript: JScript is a dialect of ECMAScript, the standardized scripting language specification. JScript was developed by Microsoft as a scripting language for their Internet Explorer web browser. JScript is very similar to JavaScript and shares most of the core syntax and features. However, there may be slight differences in the implementation and available APIs between JScript and JavaScript.

ActionScript: ActionScript is a scripting language developed by Adobe Systems, primarily used in the Adobe Flash platform. ActionScript was originally based on ECMAScript 4, which was a proposed version of the ECMAScript standard. However, ECMAScript 4 was never officially released, and Adobe continued to evolve ActionScript independently. ActionScript 3.0, which is the most widely used version, shares many similarities with ECMAScript 3 (the version that preceded ECMAScript 5).

ECMAScript: ECMAScript is a standardized scripting language specification that defines the syntax, semantics, and behavior of scripting languages like JavaScript, JScript, and ActionScript. It provides a common base for implementing these languages, ensuring compatibility and interoperability. The ECMAScript standard is maintained by the Ecma International organization and goes through a regular revision process. JavaScript is the most well-known and widely used implementation of ECMAScript, and the terms "JavaScript" and "ECMAScript" are often used interchangeably. However, other implementations like JScript and ActionScript also adhere to the ECMAScript specification.

---

3. What is an example of a JavaScript specification - and where can you find it?

An example of a JavaScript specification is the ECMAScript specification. It defines the syntax, semantics, and behavior of the JavaScript language.

The ECMAScript specification can be found on the Ecma International website, which is the organization responsible for maintaining the standard. The latest ECMAScript specification, as of my knowledge cutoff in September 2021, is ECMAScript 2022 (also known as ES12). You can access the ECMAScript specification by visiting the following URL:
https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

On that page, you will find links to download the ECMAScript specification in PDF format. Additionally, you can explore other related resources and publications provided by Ecma International.

It's worth noting that the ECMAScript specification is a detailed technical document intended for language implementers and developers who want an in-depth understanding of JavaScript's inner workings. For most practical purposes, developers can rely on JavaScript language resources and documentation, which provide practical guidance and examples for working with JavaScript.

---

4. What are v8, SpiderMonkey, Chakra and Tamarin? Do they run JavaScript differently?

V8: V8 is the JavaScript engine developed by Google, primarily used in the Google Chrome web browser. V8 is written in C++ and provides high-performance execution of JavaScript code. It utilizes techniques such as just-in-time (JIT) compilation to optimize JavaScript execution speed. V8 has gained popularity due to its fast performance and is also used in other environments like Node.js.

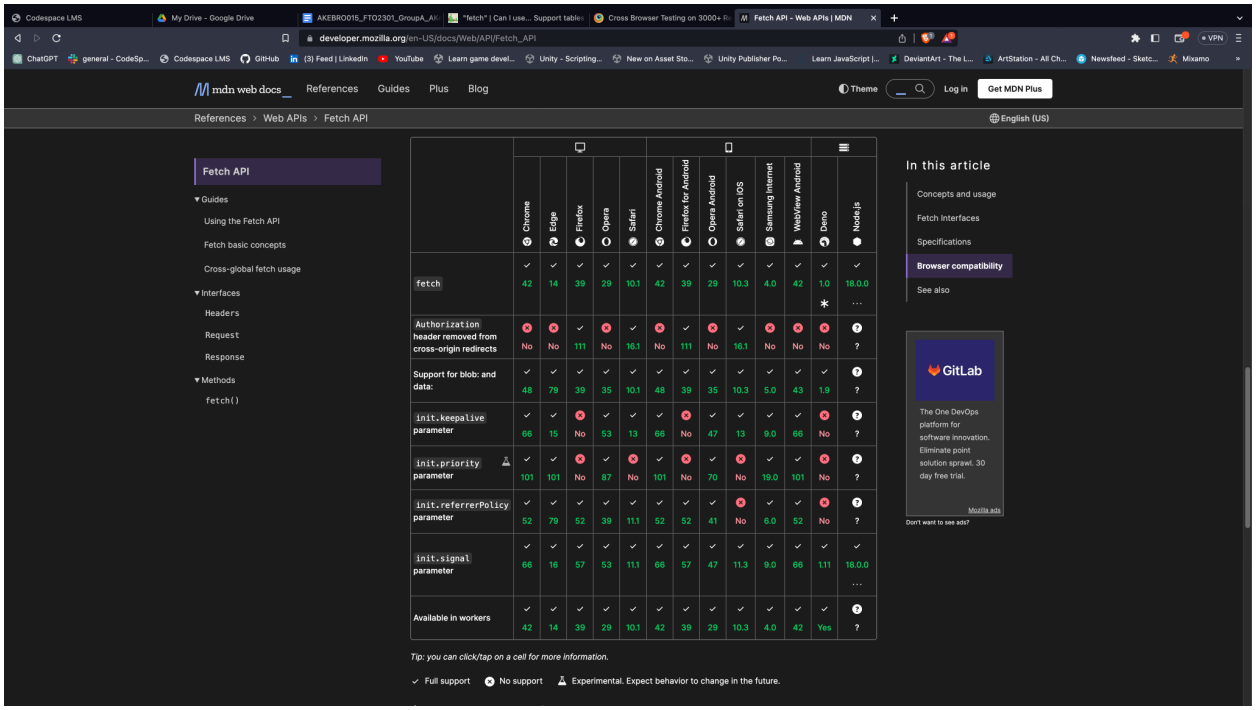
SpiderMonkey: SpiderMonkey is the JavaScript engine developed by Mozilla, the organization behind the Firefox web browser. It was one of the first JavaScript engines ever created. SpiderMonkey is implemented in C++ and features a bytecode interpreter as well as a JIT compiler. It has a long history of development and has been optimized for performance and compatibility.

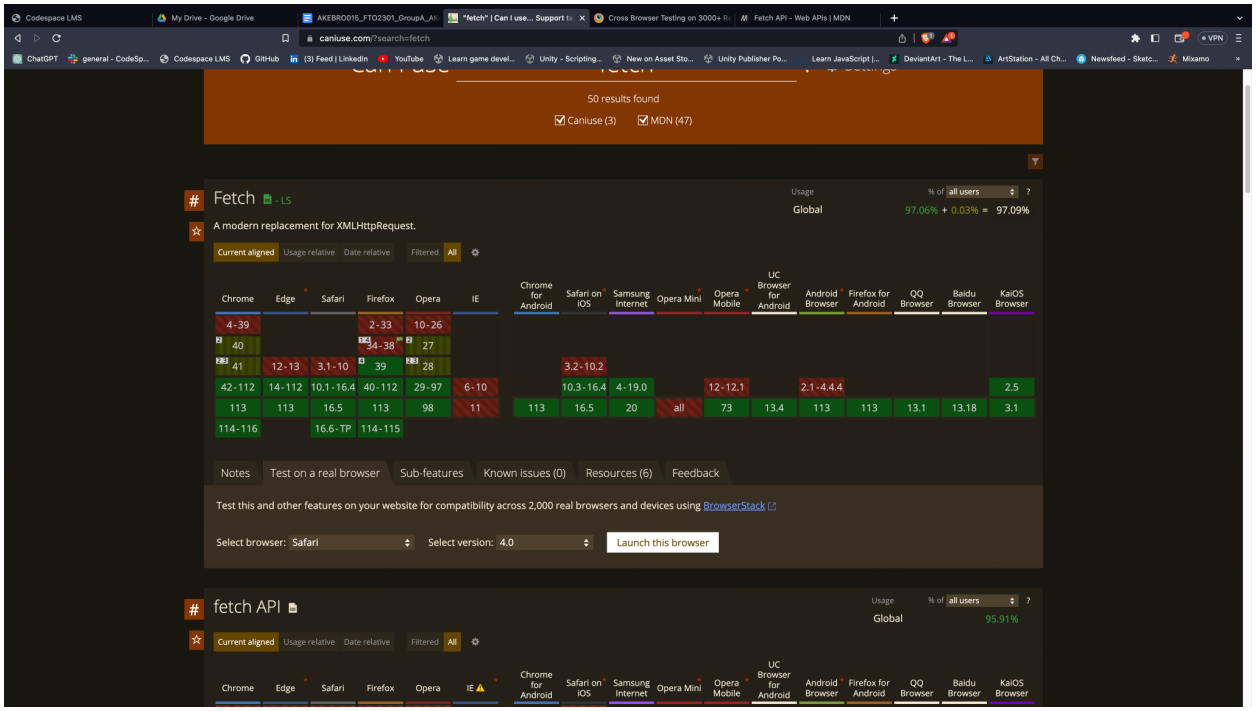
Chakra: Chakra, also known as ChakraCore, was the JavaScript engine developed by Microsoft for their Edge and Internet Explorer web browsers. ChakraCore is an open-source version of the engine that can be embedded in other applications. It was written in C++ and offered features like Just-in-Time (JIT) compilation and support for concurrent and parallel execution. However, Microsoft has transitioned to using the Blink rendering engine, which utilizes V8 as its JavaScript engine, in their newer Chromium-based Edge browser.

Tamarin: Tamarin is a JavaScript engine developed by Adobe Systems. It was specifically designed to execute ActionScript and ECMAScript bytecode used in Adobe Flash Player. Tamarin incorporated a JIT compiler and advanced garbage collection techniques to optimize performance. However, Adobe officially discontinued the development of Flash Player in 2020.

---

5. Show a practical example using **caniuse.com** and the MDN compatibility table.


mdn web docs
References
Guides
Plus
Blog
Theme
Log in
Get MDN Plus
References > Web APIs > Fetch API
English (US)
Fetch API
Guides
Using the Fetch API
Fetch basic concepts
Cross-global fetch usage
Interfaces
Headers
Request
Response
Methods
fetch()
Chrome
Edge
Firefox
Opera
Safari
Chrome Android
Firefox for Android
Opera Android
Safari on iOS
Samsung Internet
WebView Android
Deno
Node.js
fetch
Authorization header removed from cross-origin redirects
Support for blob: and data:
init.keepalive parameter
init.priority parameter
init.referrerPolicy parameter
init.signal parameter
Available in workers
Tip: you can click/tap on a cell for more information.
Full support
No support
Experimental. Expect behavior to change in the future.
In this article
Concepts and usage
Fetch Interfaces
Specifications
Browser compatibility
See also
GitLab
The One DevOps platform for software innovation. Eliminate point solution sprawl. 30 day free trial.
Mozilla ads
Don't want to see ads?


caniuse.com/?search=fetch
50 results found
Caniuse (3)
MDN (47)
Fetch
A modern replacement for XMLHttpRequest.
Usage
Global
97.06% + 0.03% = 97.09%
Notes
Test on a real browser
Sub-features
Known issues (0)
Resources (6)
Feedback
Test this and other features on your website for compatibility across 2,000 real browsers and devices using BrowserStack
Select browser: Safari
Select version: 4.0
Launch this browser
fetch API
Global
95.91%