ECMAScript

ECMAScript (/ˈɛkməskrɪpt/; **ES**)[1] is a standard for scripting languages, including JavaScript, JScript, and ActionScript. It is best known as a JavaScript standard intended to ensure the interoperability of web pages across different web browsers.[2] It is standardized by Ecma International in the document ECMA-262 (https://www.ecma-international.org/publications-and-standards/standards/ecma-262/).

ECMAScript is commonly used for client-side scripting on the World Wide Web, and it is increasingly being used for server-side applications and services using runtime environments such as Node.js,[3] Deno[4] and Bun.[5]

ECMAScript	
Paradigm	Multi-paradigm: prototype-based, functional, imperative
Designed by	Brendan Eich, Ecma International
First appeared	1997
Typing discipline	weak, dynamic
Website	www.ecma-international.org/publications-and-standards/standards/ecma-262/ (https://www.ecma-international.org/publications-and-standards/standards/ecma-262/)
Major implementations	
JavaScript, ActionScript, JScript, QtScript, InScript, Google Apps Script	
Influenced by	
Self, HyperTalk, AWK, C, CoffeeScript, Perl, Python, Java, Scheme	

ECMAScript (file format)	
Filename extensions	`.es`
Internet media type	`application/ecmascript`
Developed by	Sun Microsystems, Ecma International
Initial release	June 1997
Latest release	Edition 15 June 2024
Type of format	Scripting language
Website	Standards (https://www.ecma-international.org/publications-and-standards/standards/)

ECMAScript, ECMA-262, JavaScript

ECMA-262, or the *ECMAScript Language Specification*, defines the *ECMAScript Language*, or just **ECMAScript**.[6] ECMA-262 specifies only language syntax and the semantics of the core application programming interface (API), such as `Array`, `Function`, and `globalThis`, while valid implementations of JavaScript add their own functionality such as input/output and file system handling.

History

The ECMAScript specification is a standardized specification of a scripting language developed by Brendan Eich of Netscape; initially named Mocha, then LiveScript, and finally JavaScript.[7] In December 1995, Sun Microsystems and Netscape announced JavaScript in a press release.[8] In November 1996, Netscape announced a meeting of the Ecma International standards organization to advance the standardization of JavaScript.[9] The first edition of ECMA-262 was adopted by the Ecma General Assembly in June 1997. Several editions of the language standard have been published since then. The name "ECMAScript" was a compromise between the organizations involved in standardizing the language,

especially Netscape and Microsoft, whose disputes dominated the early standards sessions. Eich commented that "ECMAScript was always an unwanted trade name that sounds like a skin disease."[10] ECMAScript has been formalized through operational semantics by work at Stanford University and the Department of Computing, Imperial College London for security analysis and standardization.[11] "ECMA" stood for "European Computer Manufacturers Association" until 1994.

Evolution

Ecma's Technical Committee 39 (TC39) is responsible for the maintenance of ECMAScript.[12] New proposals to the language go through a staged process, with each stage representing the completeness of the proposal's specification. Consensus must be reached within the committee to advance a proposal to the next stage. Proposals that reach stage 4, the final stage, will be included into the next version of the standard.[13] Since the release of version 6 in June 2015, new major versions have been finalized and published every June.[14]

Features

The ECMAScript language includes structured, dynamic, functional, and prototype-based features.[15]

Imperative and structured

ECMAScript JavaScript supports C-style structured programming. Previously, JavaScript only supported function scoping using the keyword `var`, but ECMAScript 2015 added the keywords `let` and `const`, allowing JavaScript to support both block scoping and function scoping. JavaScript supports automatic semicolon insertion, meaning that semicolons that normally terminate a statement in C may be omitted in JavaScript.[16]

Like C-style languages, control flow is done with the `while`, `for`, `do` / `while`, `if` / `else`, and `switch` statements. Functions are weakly typed and may accept and return any type. Arguments not provided default to `undefined`.

Weakly typed

ECMAScript is weakly typed. This means that certain types are assigned implicitly based on the operation being performed. However, there are several quirks in JavaScript's implementation of the conversion of a variable from one type to another.

Dynamic

ECMAScript is dynamically typed. Thus, a type is associated with a value rather than an expression. ECMAScript supports various ways to test the type of objects, including duck typing.[17]

Transpiling

Since ES 2015, transpiling JavaScript has become very common. Transpilation is a source-to-source compilation in which newer versions of JavaScript are used, and a transpiler rewrites the source code so that it is supported by older browsers. Usually, transpilers transpile down to ES3 to maintain compatibility with all versions of browsers. The settings to transpile to a specific version can be configured according to need.

Transpiling adds an extra step to the build process and is sometimes done to avoid needing polyfills. Polyfills create new features for older environments that lack them. Polyfills do this at runtime in the interpreter, such as the user's browser or on the server. Instead, transpiling rewrites the ECMA code itself during the build phase of development before it reaches the interpreter.

Conformance

In 2010, Ecma International started developing a standards test for Ecma 262 ECMAScript.[18] Test262 is an ECMAScript conformance test suite that can be used to check how closely a JavaScript implementation follows the ECMAScript Specification. The test suite contains thousands of individual tests, each of which tests some specific requirement(s) of the ECMAScript specification. The development of Test262 is a project of the Ecma Technical Committee 39 (TC39). The testing framework and the individual tests are contributed to Ecma by member organizations of TC39.

Important contributions were made by Google (Sputnik test suite) and Microsoft, who both contributed thousands of tests. The Test262 test suite consisted of 38 014 tests as of January 2020.[19] ECMAScript specifications through ES7 are well-supported in major web browsers. The table below shows the conformance rate for current versions of software with respect to the most recent editions of ECMAScript.

Scripting engine conformance

Scripting engine	Reference application(s)	Conformance[20]			
		ES5[21]	ES6 (2015)[22]	ES2016+[23]	Next[24][25]
SpiderMonkey	Firefox 120	100%	98%	98%	5%
V8	Google Chrome 117, Microsoft Edge 113, Opera 98	100%	98%	98%	5%
JavaScriptCore	Safari 17	99%	100%	98%	11%

See also

- ECMAScript for XML (E4X)
- List of ECMAScript engines

References

1. Stefanov, Stoyan (2010). *JavaScript Patterns* (https://books.google.com/books?id=WTZqecc9olUC). O'Reilly Media, Inc. p. 5. ISBN 9781449396947. Archived (https://web.archive.org/web/20160610005241/https://books.google.com/books?id=WTZqecc9olUC) from the original on 2016-06-10. Retrieved 2016-01-12. "The core JavaScript programming language [...] is based on the *ECMAScript* standard, or ES for short."
2. Wirfs-Brock, Allen; Eich, Brendan (2020-05-02). "JavaScript: The First 20 Years" (https://doi.org/10.1145%2F3386327). *Proceedings of the ACM on Programming Languages*. **4** (HOPL): 1–189. doi:10.1145/3386327 (https://doi.org/10.1145%2F3386327). S2CID 219603695 (https://api.semanticscholar.org/CorpusID:219603695).
3. Wunder, C. "Node.js — ECMAScript 2015 (ES6) and beyond" (https://nodejs.org/en/docs/es6). *Node.js*.

4. "Deno joins JavaScript standards effort" (https://www.infoworld.com/article/3644460/deno-joins-javascript-standards-effort.html). 14 December 2021.
5. https://bun.sh/docs#:~:text=or%2C%20more%20formally%2C-,ECMAScript,-)%20is%20just%20a bun
6. Guo, Shu-yu (2022-02-14). "ECMAScriptÂŽ 2022 Language Specification" (https://tc39.es/ecma262/). *tc39.es*. Archived (https://web.archive.org/web/20200508053013/https://tc39.es/ecma262/) from the original on 2020-05-08. Retrieved 2022-02-14.
7. Krill, Paul (2008-06-23). "JavaScript creator ponders past, future" (http://www.infoworld.com/article/2653798/application-development/javascript-creator-ponders-past--future.html). *infoworld.com*. InfoWorld. Archived (https://web.archive.org/web/20140920141040/http://www.infoworld.com/article/2653798/application-development/javascript-creator-ponders-past--future.html) from the original on 2014-09-20. Retrieved 2013-10-31.
8. "Netscape and Sun announce JavaScript, the Open, Cross-platform Object Scripting Language for Enterprise Networks and the Internet" (https://web.archive.org/web/20020606002913/http://wp.netscape.com/newsref/pr/newsrelease67.html). *Netscape.com*. Netscape. 1995-12-04. Archived from the original (http://wp.netscape.com/newsref/pr/newsrelease67.html) on 2002-06-06. Retrieved 2019-11-04.
9. Press Release (November 15, 1996). "Industry Leaders to Advance Standardization of Netscape's JavaScript at Standards Body Meeting" (https://web.archive.org/web/19981203070212/http://cgi.netscape.com/newsref/pr/newsrelease289.html). *Netscape.com*. Netscape. Archived from the original (http://cgi.netscape.com/newsref/pr/newsrelease289.html) on 1998-12-03. Retrieved 2013-10-31.
10. Eich, Brendan (2006-10-03). "Will there be a suggested file suffix for es4?" (https://mail.mozilla.org/pipermail/es-discuss/2006-October/000133.html). *mozilla.org*. Mail.mozilla.org. Archived (https://web.archive.org/web/20200621202321/https://mail.mozilla.org/pipermail/es-discuss/2006-October/000133.html) from the original on 2020-06-21. Retrieved 2021-05-05.
11. Maffeis, Sergio; Mitchell, John C.; Taly, Ankur (2020-01-03). "An Operational Semantics for JavaScript" (http://theory.stanford.edu/people/jcm/papers/aplas08-camera-ready.pdf) (PDF). *stanford.edu*. Association for Computing Machinery. Archived (https://web.archive.org/web/20200103204704/http://theory.stanford.edu/people/jcm/papers/aplas08-camera-ready.pdf) (PDF) from the original on 2020-01-03. Retrieved 2020-01-03.
12. "TC39" (https://ecma-international.org/technical-committees/tc39), *Technical Committees*, Ecma International, retrieved 2024-08-11
13. "The TC39 Process" (https://tc39.es/process-document), *TC39*, Ecma International, retrieved 2024-08-11
14. "ECMAScript, TC39, and the History of JavaScript" (https://ui.dev/ecmascript), *ui.dev*, retrieved 2024-08-11
15. "About" (https://archive.today/20120802115457/http://www.ecmascript.org/about.php). ECMAScript. Archived from the original (http://www.ecmascript.org/about.php) on 2012-08-02. Retrieved 2009-12-17.
16. Flanagan, David (17 August 2006). *JavaScript: The Definitive Guide* (https://archive.org/details/javascriptdefini0000flan) (5th ed.). O'Reilly. p. 16. ISBN 978-0-596-10199-2.
17. "JavaScript data types and data structures – JavaScript | MDN" (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures). *Developer.mozilla.org*. 2017-02-16. Archived (https://web.archive.org/web/20170314230542/https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures) from the original on 2017-03-14. Retrieved 2017-02-24.
18. "ECMAScript Language – test262" (https://web.archive.org/web/20110514205704/http://test262.ecmascript.org/). Test262.ecmascript.org. Archived from the original (http://test262.ecmascript.org/) on 2011-05-14. Retrieved 2013-10-31.
19. "tc39/test262" (https://github.com/tc39/test262). *GitHub*. January 24, 2020. Archived (https://web.archive.org/web/20191001032806/https://github.com/tc39/test262) from the original on October 1, 2019. Retrieved January 29, 2020.
20. ES5 is the baseline for this test suite. The conformance rate for other editions reflects support for new features only, not a comprehensive score.

21. "ECMAScript 5 compatibility table" (https://compat-table.github.io/compat-table/es5/). *compat-table.github.io*. 2024-04-14. Archived (https://web.archive.org/web/20240114035054/https://compat-table.github.io/compat-table/es5/) from the original on 2024-01-14. Retrieved 2024-04-14.
22. "ECMAScript 6 compatibility table" (https://compat-table.github.io/compat-table/es6/). *compat-table.github.io*. 2024-04-14. Archived (https://web.archive.org/web/20240404212624/https://compat-table.github.io/compat-table/es6/) from the original on 2024-04-04. Retrieved 2024-04-14.
23. "ECMAScript 2016+ compatibility table" (https://compat-table.github.io/compat-table/es2016plus/). *compat-table.github.io*. 2024-04-14. Archived (https://web.archive.org/web/20240114034945/https://compat-table.github.io/compat-table/es2016plus/) from the original on 2024-01-14. Retrieved 2024-04-14.
24. "ECMAScript Next compatibility table" (https://compat-table.github.io/compat-table/esnext/). *compat-table.github.io*. 2024-04-14. Archived (https://web.archive.org/web/20240114035113/https://compat-table.github.io/compat-table/esnext/) from the original on 2024-01-14. Retrieved 2024-04-14.
25. Composite score that includes new features from ES7 through next edition drafts

---