WIKIPEDIA

ECMAScript

ECMAScript (or **ES**)[1] is a scripting-language specification standardized by Ecma International in **ECMA-262** and ISO/IEC 16262. It was created to standardize JavaScript, so as to foster multiple independent implementations. JavaScript has remained the best-known implementation of ECMAScript since the standard was first published, with other well-known implementations including JScript and ActionScript.[2] ECMAScript is commonly used for client-side scripting on the World Wide Web, and it is increasingly being used for writing server applications and services using Node.js.

Contents

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 (http://www.ecma-international.org)
Major implementations	
JavaScript, SpiderMonkey, V8, ActionScript, JScript, QtScript, InScript, Google Apps Script	
Influenced by	
Self, HyperTalk, AWK, C, CoffeeScript, Perl, Python, Java, Scheme	

ECMAScript

History

The ECMAScript specification is a standardized specification of a scripting language developed by Brendan Eich of Netscape; initially it was named Mocha, later LiveScript, and finally JavaScript.[3] In December 1995, Sun Microsystems and Netscape announced JavaScript in a press release.[4] 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."[5]

While both JavaScript and JScript aim to be compatible with ECMAScript, they also provide additional features not described in the ECMA specifications.[6]

Filename extensions	`.es`
Internet media type	`application/ecmascript`
Developed by	Sun Microsystems, Ecma International
Initial release	June 1997
Latest release	Edition 9 (June 2018)
Type of format	Scripting language
Website	ECMA-262 (http://www.ecma-international.org/publications/standards/Ecma-262.htm), ECMA-290 (http://www.ecma-international.org/publications/standards/Ecma-290.htm), ECMA-327 (http://www.ecma-international.org/publications/standards/Ecma-327.htm), ECMA-357 (http://www.ecma-international.org/publications/standards/Ecma-357.htm), ECMA-402 (http://www.ecma-international.org/publications/standards/Ecma-402.htm)

Versions

There are nine editions of ECMA-262 published. Work on version 9 of the standard was finalized in June 2018.[7]

Edition	Date published	Name	Changes from prior edition	Editor
1	June 1997		First edition	Guy L. Steele Jr.
2	June 1998		Editorial changes to keep the specification fully aligned with ISO/IEC 16262 international standard	Mike Cowlishaw
3	December 1999		Added regular expressions, better string handling, new control statements, try/catch exception handling, tighter definition of errors, formatting for numeric output and other enhancements	Mike Cowlishaw
4	*Abandoned*		Fourth Edition was abandoned, due to political differences concerning language complexity. Many features proposed for the Fourth Edition have been completely dropped; some were incorporated into the sixth edition.	
5	December 2009		Adds "strict mode," a subset intended to provide more thorough error checking and avoid error-prone constructs. Clarifies many ambiguities in the 3rd edition specification, and accommodates behaviour of real-world implementations that differed consistently from that specification. Adds some new features, such as getters and setters, library support for JSON, and more complete reflection on object properties.[8]	Pratap Lakshman, Allen Wirfs-Brock
5.1	June 2011		This edition 5.1 of the ECMAScript standard is fully aligned with third edition of the international standard ISO/IEC 16262:2011.	Pratap Lakshman, Allen Wirfs-Brock
6	June 2015[9]	ECMAScript 2015 (ES2015)	See 6th Edition - ECMAScript 2015	Allen Wirfs-Brock
7	June 2016[10]	ECMAScript 2016 (ES2016)	See 7th Edition - ECMAScript 2016	Brian Terlson
8	June 2017[11]	ECMAScript 2017 (ES2017)	See 8th Edition - ECMAScript 2017	Brian Terlson
9	June 2018[7]	ECMAScript 2018 (ES2018)	See 9th Edition - ECMAScript 2018	Brian Terlson

In June 2004, Ecma International published ECMA-357 standard, defining an extension to ECMAScript, known as ECMAScript for XML (E4X). Ecma also defined a "Compact Profile" for ECMAScript – known as ES-CP, or ECMA 327 – that was designed for resource-constrained devices, which was withdrawn in 2015.[12]

4th Edition (abandoned)

The proposed fourth edition of ECMA-262 (**ECMAScript 4** or **ES4**) would have been the first major update to ECMAScript since the third edition was published in 1999. The specification (along with a reference implementation) was originally targeted for completion by October 2008.[13] An overview of the language was released by the working group on October 23, 2007.[14]

By August 2008, the ECMAScript 4th edition proposal had been scaled back into a project codenamed ECMAScript Harmony. Features under discussion for Harmony at the time included

- classes,
- a module system,
- optional type annotations and static typing, probably using a structural type system,
- generators and iterators,
- destructuring assignment, and
- algebraic data types.

Intent of these features was partly to better support *programming in the large*, and to allow sacrificing some of the script's ability to be dynamic to improve performance. For example, Tamarin – the virtual machine for ActionScript, developed and open sourced by Adobe – has just-in-time compilation (JIT) support for certain classes of scripts.

In addition to introducing new features, some ES3 bugs were proposed to be fixed in edition 4.[15][16] These fixes and others, and support for JSON encoding/decoding, have been folded into the ECMAScript, 5th Edition specification.[17]

Work started on Edition 4 after the ES-CP (Compact Profile) specification was completed, and continued for approximately 18 months where slow progress was made balancing the theory of Netscape's JavaScript 2 specification with the implementation experience of Microsoft's JScript .NET. After some time, the focus shifted to the ECMAScript for XML (E4X) standard. The update has not been without controversy. In late 2007, a debate between Eich, later the Mozilla Foundation's CTO, and Chris Wilson, Microsoft's platform architect for Internet Explorer, became public on a number of blogs. Wilson cautioned that because the proposed changes to ECMAScript made it backwards incompatible in some respects to earlier versions of the language, the update amounted to "breaking the Web,"[18] and that stakeholders who opposed the changes were being "hidden from view".[19] Eich responded by stating that Wilson seemed to be "repeating falsehoods in blogs" and denied that there was attempt to suppress dissent and challenged critics to give specific examples of incompatibility.[20] He pointed out that Microsoft Silverlight and Adobe AIR rely on C# and ActionScript 3 respectively, both of which are larger and more complex than ECMAScript Edition 3.[21]

5th Edition

Yahoo, Microsoft, Google, and other 4th edition dissenters formed their own subcommittee to design a less ambitious update of ECMAScript 3, tentatively named ECMAScript 3.1. This edition would focus on security and library updates with a large emphasis on compatibility. After the aforementioned public sparring, the ECMAScript 3.1 and ECMAScript 4 teams agreed on a compromise: the two editions would be worked on, in parallel, with coordination between the teams to ensure that ECMAScript 3.1 remains a strict subset of ECMAScript 4 in both semantics and syntax.

However, the differing philosophies in each team resulted in repeated breakages of the subset rule, and it remained doubtful that the ECMAScript 4 dissenters would ever support or implement ECMAScript 4 in the future. After over a year since the disagreement over the future of ECMAScript within the Ecma Technical Committee 39, the two teams reached a new compromise in July 2008: Brendan Eich announced that Ecma TC39 would focus work on the ECMAScript 3.1 (later renamed to ECMAScript, 5th Edition) project with full collaboration of all parties, and vendors would target at least two interoperable implementations by early 2009.[22][23] In April 2009, Ecma TC39 published the "final" draft of the 5th edition and announced that testing of interoperable implementations was expected to be completed by mid-July.[24] On December 3, 2009, ECMA-262 5th edition was published.[25]

6th Edition - ECMAScript 2015

The 6th edition, initially known as ECMAScript 6 (**ES6**) then and later renamed to ECMAScript 2015, was finalized in June 2015.[9][26] This update adds significant new syntax for writing complex applications, including class declarations (`class Foo { ... }`), ES6 modules like `import * as moduleName from "..."; export const Foo`, but defines them semantically in the same terms as ECMAScript 5 strict mode. Other new features include iterators and `for/of` loops, Python-style generators, arrow function expression (`() => {...}`), `let` keyword for local declarations, `const` keyword for constant variable declarations, binary data, typed arrays, new collections (maps, sets and WeakMap), promises, number and math enhancements, reflection, proxies (metaprogramming for virtual objects and wrappers) and template literals for strings.[27][28] The complete list is extensive.[29][30] As the first "ECMAScript Harmony" specification, it is also known as "ES6 Harmony."

7th Edition - ECMAScript 2016

The 7th edition, officially known as ECMAScript 2016, was finalized in June 2016.[10] The major standard language features include block-scoping of variables and functions, destructuring patterns (of variables), proper tail calls, exponentiation operator `**` for numbers, `await`, `async` keywords for asynchronous programming.[10][31]

8th Edition - ECMAScript 2017

The 8th edition, officially known as ECMAScript 2017, was finalized in June 2017.[11] Includes`async/await`constructions, which work using generators and promises. [32] ECMAScript 2017 (ES2017), the eighth edition, includes features for concurrency and atomics, syntactic integration with promises (async/await).[32][11]

9th Edition - ECMAScript 2018

The 9th edition, officially known as ECMAScript 2018, was finalized in June 2018.[7] New features include rest/spread operators for variables (three dots: `...identifier`), asynchronous iteration, `Promise.prototype.finally()` and additions to RegExp.[7]

ES.Next

ES.Next is a dynamic name that refers to whatever the next version is at time of writing. ES.Next features are more correctly called *proposals*, because, by definition, the specification has not been finalized yet.

Features

The ECMAScript language includes structured, dynamic, functional, and prototype-based features.[33]

Concise syntax

ES6 brought with it a new mode of defining functions, so called "arrow functions."[34] In ES5, a function would be defined as such:

```
var readWikiArticle = function(content) {
    // Read it!
};
```

Whereas in ES6, using the new concise arrow function syntax:

```
var readWikiArticle = (content) => {
    //Read article!
};
```

Arrow functions also improve variable binding between functions.

Transpiling

Since ES 2015, transpiling JavaScript has become very common. Transpilation is a source to source compilation in which the newer versions of JavaScript are used in the user's source code and the transpiler rewrites them so that they are compliant with the current specification.[35] Usually, transpilers transpile down to ES3 to maintain compatibility with all versions of browsers. The settings to transpiling to a specific version can be configured according to need. Transpiling adds an extra step to the build process and sometimes to avoid that polyfills can be used as well. Polyfills allow adding extra functionalities by including another JavaScript file which adds those specific functionalities.

Conformance

In 2010, Ecma International started developing a standards test for Ecma 262 ECMAScript.[36] Test262 is an ECMAScript conformance test suite that can be used to check how closely a JavaScript implementation follows the ECMAScript 5th Edition Specification. The test suite contains thousands of individual tests, each of which tests some specific requirements of the ECMAScript specification.

Development of test262 is a project of Ecma Technical Committee 39 (TC39). The testing framework and individual tests are created by member organizations of TC39 and contributed to Ecma for use in Test262.

Important contributions were made by Google (Sputnik testsuite) and Microsoft who both contributed thousands of tests. The Test262 testsuite already contains more than 11,000 tests and is being developed further as of 2013.

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	Reference application(s)	Conformance[37]			
		ES5[38]	ES6[39]	ES7[40]	Newer (2016+)[40][41]
Chakra	Microsoft Edge 18	100%	96%	100%	48%
SpiderMonkey	Firefox 67	100%	98%	100%	83%
Chrome V8	Google Chrome 75, Opera 62	100%	98%	100%	98%
JavaScriptCore (Nitro)	Safari 12.1	99%	99%	100%	87%

See also

- Comparison of layout engines (ECMAScript)
- ECMAScript for XML (E4X)
- JavaScript
- JScript
- 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. Retrieved 2016-01-12. "The core JavaScript programming language [...] is based on the *ECMAScript* standard, or ES for short."
2. "A Short History of JavaScript" (https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript). *W3C*. Retrieved 31 March 2017.
3. 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. Retrieved 2013-10-31.
4. "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. November 15, 1996. Archived from the original (http://cgi.netscape.com/newsref/pr/newsrelease289.html) on 1998-12-03. Retrieved 2013-10-31.

5. "Will there be a suggested file suffix for es4?" (https://mail.mozilla.org/pipermail/es4-discuss/2006-October/000133.html). Mail.mozilla.org. 2006-10-03. Retrieved 2013-10-31.
6. "JScript VS JavaScript" (http://javascript.about.com/od/reference/a/jscript.htm). About.com. 2015-11-25.
7. "ECMAScript 2018 Language Specification" (http://www.ecma-international.org/ecma-262/9.0/index.html). Ecma International. June 2018.
8. "Changes to JavaScript, Part 1: EcmaScript 5" (https://www.youtube.com/watch?v=Kq4FpMe6cRs). YouTube. 2009-05-18. Retrieved 2013-10-31.
9. "ECMAScript 2015 Language Specification" (http://www.ecma-international.org/ecma-262/6.0/index.html). Ecma International. June 2015.
10. "ECMAScript 2016 Language Specification" (http://www.ecma-international.org/ecma-262/7.0/index.html). Ecma International. June 2016.
11. "ECMAScript 2017 Language Specification" (http://www.ecma-international.org/ecma-262/8.0/index.html). Ecma International. June 2017.
12. 2015-03-24 Meeting Notes (https://esdiscuss.org/notes/2015-03-24). ESDiscuss. Also see Ecma withdrawn Standards (http://www.ecma-international.org/publications/standards/Standardwithdrawn.htm). ECMA.
13. "ES4 overview paper released" (https://mail.mozilla.org/pipermail/es-discuss/2007-October/001281.html). Mail.mozilla.org. Retrieved 2013-10-31.
14. "Proposed ECMAScript 4th Edition – Language Overview" (https://www.webcitation.org/5rBiWD4P6?url=http://www.ecmascript.org/es4/spec/overview.pdf) (PDF). *ecmascript.org*. 23 October 2007. Archived from the original (http://www.ecmascript.org/es4/spec/overview.pdf) (PDF) on 13 July 2010.
15. John Resig. "John Resig – Bug Fixes in JavaScript 2" (http://ejohn.org/blog/bug-fixes-in-javascript-2/). Ejohn.org. Retrieved 2013-10-31.
16. "Compatibility Between ES3 and Proposed ES4" (http://www.ecmascript.org/es4/spec/incompatibilities.pdf) (PDF). Ecmascript.org. Retrieved 2013-10-31.
17. "Wayback Machine" (https://web.archive.org/web/20090419044026/http://www.ecma-international.org/publications/files/drafts/tc39-2009-025.pdf) (PDF). 2009-04-19. Retrieved 2018-03-19.
18. "ECMAScript 3 and Beyond – IEBlog – Site Home – MSDN Blogs" (http://blogs.msdn.com/ie/archive/2007/10/30/ecmascript-3-and-beyond.aspx#5788577). Blogs.msdn.com. 2007-10-30. Retrieved 2013-10-31.
19. "What I think about ES4. - Albatross! - Site Home – MSDN Blogs" (http://blogs.msdn.com/cwilso/archive/2007/10/31/what-i-think-about-es4.aspx). Blogs.msdn.com. 2007-10-31. Retrieved 2013-10-31.
20. "Open letter to Chris Wilson" (http://brendaneich.com/2007/10/open-letter-to-chris-wilson/). Brendan Eich. 2007-10-31. Retrieved 2013-10-31.
21. "JavaScript 2 and the Open Web" (https://brendaneich.com/2007/11/my-media-ajax-keynote/). 2007-11-20. Retrieved 2014-01-20.
22. "ECMAScript Harmony" (https://mail.mozilla.org/pipermail/es-discuss/2008-August/003400.html). Mail.mozilla.org. Retrieved 2013-10-31.
23. "A Major Milestone in JavaScript Standardization – JScript Blog – Site Home – MSDN Blogs" (http://blogs.msdn.com/jscript/archive/2009/04/09/a-major-milestone-in-javascript-standardization.aspx). Blogs.msdn.com. 2009-04-09. Retrieved 2013-10-31.
24. "Ecma International finalises major revision of ECMAScript" (http://www.ecma-international.org/news/PressReleases/PR_Ecma_finalises_major_revision_of_ECMAScript.htm). Ecma International. 2009-04-09. Retrieved 2009-05-22.
25. "Ecma latest news" (http://www.ecma-international.org/news/index.html#Docs98thGA). Ecma-international.org. Retrieved 2013-10-31.
26. Krill, Paul. "It's official: ECMAScript 6 is approved" (http://www.infoworld.com/article/2937716/javascript/its-official-ecmascript-6-is-approved.html). *InfoWorld*. Retrieved 2018-03-19.
27. "5 Great Features in EcmaScript 6 (ES6 Harmony) - Wintellect" (http://www.wintellect.com/devcenter/nstieglitz/5-great-features-in-es6-harmony). *Wintellect*. 2014-03-24. Retrieved 2018-03-19.
28. "ECMAScript 6 (ES6): What's New In The Next Version Of JavaScript" (https://www.smashingmagazine.com/2015/10/es6-whats-new-next-version-javascript/). *Smashing Magazine*. 2015-10-28. Retrieved 2018-03-19.

29. "ECMAScript 6: New Features: Overview and Comparison" (http://es6-features.org/). *es6-features.org*. Retrieved 2018-03-19.
30. "Standard ECMA-262 6th Edition / June 2015 ECMAScript® 2015 Language Specification 14.2 Arrow Function Definitions" (https://www.ecma-international.org/ecma-262/6.0/index.html#sec-arrow-function-definitions). *www.ecma-international.org*. 2015.
31. Saboff, Michael (2016-05-23). "ECMAScript 6 Proper Tail Calls in WebKit" (https://webkit.org/blog/6240/ecmascript-6-proper-tail-calls-in-webkit/). *WebKit*. Retrieved 2019-04-11.
32. "ECMAScript 2017 (ES8): the final feature set" (http://2ality.com/2016/02/ecmascript-2017.html). *2ality*. Retrieved 2018-04-23.
33. "About" (https://archive.is/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.
34. "Arrow Functions: Fat and Concise Syntax in JavaScript" (https://www.sitepoint.com/es6-arrow-functions-new-fat-concise-syntax-javascript/). *SitePoint*. 2018-04-09. Retrieved 2018-11-21.
35. "Using ES2015 today" (http://wisdomgeek.com/web-development/using-es6-today-future-of-javascript/). *Wisdom Geek*. 2016-05-12. Retrieved 2018-08-29.
36. "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.
37. ES5 is the baseline for this test suite. The conformance rate for other editions reflects support for new features only, not a comprehensive score.
38. "ECMAScript 5 compatibility table" (https://kangax.github.io/compat-table/es5). *kangax.github.io*. Retrieved 2018-11-08.
39. "ECMAScript 6 compatibility table" (https://kangax.github.io/compat-table/es6). *kangax.github.io*. Retrieved 2018-11-08.
40. "ECMAScript 2016+ compatibility table" (https://kangax.github.io/compat-table/es2016plus). *kangax.github.io*. Retrieved 2018-11-08.
41. Composite score that includes new features from ES7 through next edition drafts

External links

- Official website (http://ecma-international.org/)

ISO Standard

- ISO 16262 (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=55755)

Ecma Standards

- ECMA-262 (http://www.ecma-international.org/publications/standards/Ecma-262.htm)
  - ECMA-262 ECMAScript Language Specification 3rd edition (December 1999) (http://www.ecma-international.org/publications/files/ECMA-ST-ARCH/ECMA-262,%203rd%20edition,%20December%201999.pdf)
  - ECMAScript Language Specification, Edition 3 Final, 24-Mar-00 (https://web.archive.org/web/20100201000000*/http://www.mozilla.org/js/language/E262-3.pdf)
  - ECMA-262 ECMAScript Language Specification 5th edition (December 2009) (http://www.ecma-international.org/publications/files/ECMA-ST-ARCH/ECMA-262%205th%20edition%20December%202009.pdf)
  - ECMA-262 ECMAScript Language Specification 5.1 edition (June 2011) (https://web.archive.org/web/20111103184035/http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-262%20edition%205.1%2C%20June%202011.pdf)

- ECMA-290 ECMAScript Components Specification (June 1999) (http://www.ecma-international.org/publications/standards/Ecma-290.htm)
- ECMA-327 ECMAScript 3rd Edition Compact Profile (June 2001) (http://www.ecma-international.org/publications/standards/Ecma-327.htm)
- ECMA-357 ECMAScript for XML (E4X) Specification (June 2004) (https://web.archive.org/web/20131104082608/http://www.ecma-international.org/publications/standards/Ecma-357.htm)
