

# Welcome to Developer News.

This is a free, open source, no-ads place to cross-post your blog articles. Read about it here.

28 OCTOBER 2017 / [#JAVASCRIPT](#) [#WEB DEVELOPMENT](#) [#PROGRAMMING](#)

## What's the difference between JavaScript and ECMAScript?





by Michael Aranda

I've tried googling "the difference between JavaScript and ECMAScript."

I ended up having to wade through a sea of ambiguous and seemingly conflicting results:

“JavaScript is a standard.”

“ECMAScript is a specification.”

“JavaScript is an implementation of the ECMAScript standard.”

“ECMAScript is standardized JavaScript.”

“ECMAScript is a language.”

“JavaScript is a dialect of ECMAScript.”

“ECMAScript is JavaScript.”

Holding back the urge to cry, I bucked up and decided to commit to some painful yet productive research.

This article represents my current understanding of the differences between JavaScript and ECMAScript. It is geared towards people who are familiar with JavaScript but would like a clearer understanding of its relationship with ECMAScript, web browsers, [Babel](#), and more. You will also learn about scripting languages, JavaScript engines, and JavaScript runtimes for good measure.

## A JavaScript/ECMAScript glossary

Below is a list of definitions, designed with a focus on consistency and clarity. The definitions are not 100% complete. They are constructed in a way that provides a high-level understanding of the connection and relationship between JavaScript and ECMAScript.

Without further ado, let's get started.

### Ecma International

An organization that creates standards for technologies.



To illustrate an example of “standard” (though not one created by Ecma), think of all the keyboards you have ever used. Did the vast majority have letters in the same order, and a space bar, an Enter key, arrow keys, with numbers displayed in a row at the top? This is

ECMAScript  
standard.

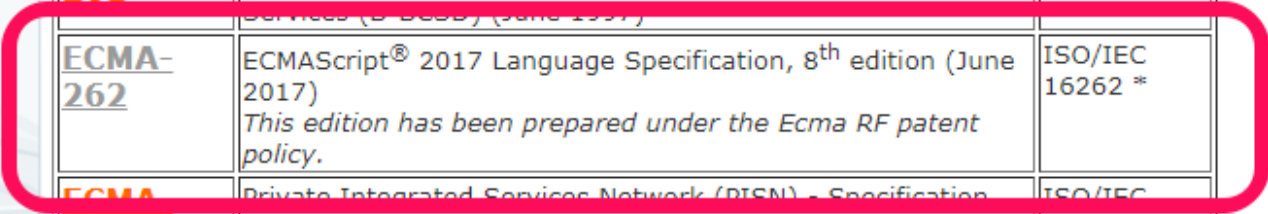
## ECMA-262

This is a standard published by Ecma International. It contains the specification for a general purpose scripting language.



specification, it represents a scripting language specification called ECMAScript.

Think of ECMA-262 as ECMAScript's reference number.



<a href="#">ECMA-259</a>	Data Interchange on 12,7 mm 208-Track Magnetic Tape Cartridges - DLT 5 Format (June 1997)	ISO/IEC 15896 *
<a href="#">ECMA-260</a>	Data Interchange on 356 mm Optical Disk Cartridges - WORM, using Phase Change Technology Capacity: 14,8 and 25 Gbytes per Cartridge (June 1997)	ISO/IEC 15898
<a href="#">ECMA-261</a>	Broadband Private Integrated Services Network (B-PISN) - Service Description - Broadband Connection Oriented Bearer Services (B-COS) (June 1997)	ISO/IEC 15899
<a href="#">ECMA-262</a>	ECMAScript® 2017 Language Specification, 8 <sup>th</sup> edition (June 2017) <i>This edition has been prepared under the Ecma RF patent policy.</i>	ISO/IEC 16262 *
<a href="#">ECMA-263</a>	Private Integrated Services Network (PISN) - Specification Functional Model and Information Flows - Call Priority Interruption and Call Priority Interruption Protection Supplementary Services (CPI(P)SD), 3 <sup>rd</sup> edition (December 2001)	ISO/IEC 15991 ETSI EN 301 655
<a href="#">ECMA-264</a>	Private Integrated Services Network (PISN) - Inter-Exchange Signalling Protocol - Call Priority Interruption and Call Priority Interruption Protection Supplementary Services (QSIG-CPI(P)), 3 <sup>rd</sup> edition (December 2001)	ISO/IEC 15992 ETSI EN 301 656

ECMA-260, ECMA-261, ECMA-262. There's ECMAScript.

## A scripting language

**A programming language designed specifically for acting on an existing entity or system**

For a general idea of what makes a programming language a scripting language, consider the commands “walk”, “run”, and “jump.” These actions require something to carry them out, perhaps a person, a dog, or a video game character. Without an actor to perform these commands, “walk”, “run”, and “jump” wouldn’t make sense. This set of actions is analogous to a scripting language that focuses on manipulating an external entity.

## ECMAScript

**The specification defined in ECMA-262 for creating a general purpose scripting language.**

**Synonym:** ECMAScript specification



While ECMA-262 is the name of the standard, it represents the scripting language specification ECMAScript.

Photo credit: [code.tutsplus.com](https://code.tutsplus.com)

ECMAScript provides the rules, details, and guidelines that a scripting language must observe to be considered ECMAScript compliant.



## 2 Conformance

A conforming implementation of ECMAScript must provide and support all the types, values, objects, properties, functions, and program syntax and semantics described in this specification.

A conforming implementation of ECMAScript must interpret source text input in conformance with the latest version of the Unicode Standard and ISO/IEC 10646.

A conforming implementation of ECMAScript that provides an application programming interface that supports programs that need to adapt to the linguistic and cultural conventions used by different human languages and countries must implement the interface defined by the most recent edition of ECMA-402 that is compatible with this specification.

A conforming implementation of ECMAScript may provide additional types, values, objects, properties, and functions beyond those described in this specification. In particular, a conforming implementation of ECMAScript may provide properties not described in this specification, and values for those properties, for objects that are described in this specification.

A conforming implementation of ECMAScript may support program and regular expression syntax not described in this specification. In particular, a conforming implementation of ECMAScript may support program syntax that makes use of the “future reserved words” listed in subclause 11.6.2.2 of this specification.

## JavaScript

**A general purpose scripting language that conforms to the ECMAScript specification.**

A conforming implementation of ECMAScript must not implement any extension that is listed as a Forbidden Extension in subclause 16.2.

An excerpt from the [ECMAScript 2017 Language Specification](#). The document is only about 900 pages, if you are looking for a light read.



Photo credit: [Udemy](#).

JavaScript is the coffee-flavored language with which I love to program. ECMAScript is the specification it's based on. By reading the [ECMAScript specification](#), you learn how to create

scripting language.

When people call JavaScript a “dialect of the ECMAScript language,” they mean it in the same sense as when talking about English, French, or Chinese dialects. A dialect derives most of its lexicon and syntax from its parent language, but deviates enough to deserve distinction.

JavaScript mostly implements the ECMAScript specification as described in ECMA-262, but a handful of differences do exist. Mozilla outlines JavaScript’s non-ECMAScript language features [here](#):

The following features are already implemented, but only available in the [Firefox Nightly channel](#) and not yet included in a draft edition of an ECMAScript specification.

### Additions to the ArrayBuffer object

- `ArrayBuffer.transfer()` ([spec](#))

### New TypedObject objects

- [Typed Objects draft](#)

### New SIMD objects

- [SIMD specification draft and polyfill](#)

### New Shared Memory objects

- `SharedArrayBuffer`
- `Atomics`

A screenshot from September 3, 2017. It is a list of JavaScript's experimental features that are not a part of ECMAScript (at least not yet).

## A JavaScript engine

A program or interpreter that understands and executes JavaScript code.

**Synonyms:** JavaScript interpreter, JavaScript implementation



JavaScript engines are commonly found in web browsers, including V8 in Chrome, SpiderMonkey in Firefox, and Chakra in Edge. Each engine is like a language module for its application, allowing it to support a certain subset of the JavaScript language.

A JavaScript engine to a browser is like language comprehension to a person. If we re-visit our example of the actions “walk”, “run”, “jump”, a JavaScript engine is the part of an “entity” that actually understands what these actions mean.

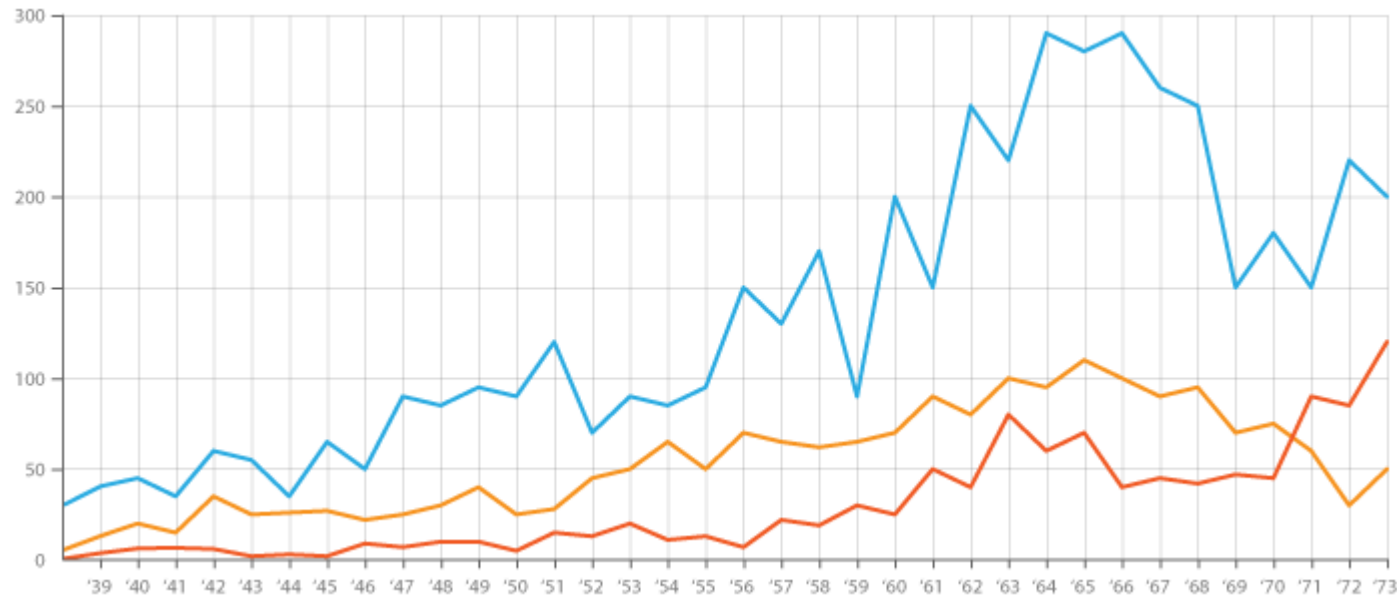


Photo credit: [datavizcatalogue.com](http://datavizcatalogue.com)

## Differences in browser performance

Two people may recognize the command “jump”, but one may react to the command faster because the person can understand and process the command faster than the other person. Similarly, two browsers can understand JavaScript code, but one runs it faster because its JavaScript engine is implemented more efficiently.





Photo credit: [vcsolutions.com](http://vcsolutions.com)

## Differences in browser support

Consider the differences that exist between people who speak the same language. Even if many people speak English, some may know some words, expressions, and syntax rules that others don't, and vice versa. Browsers are the same way. Even though the JavaScript engines of browsers all understand JavaScript, some browsers have a greater understanding of the language than others. There are differences in the way browsers support the language.

With regards to browser support, people usually talk about “ECMAScript compatibility” rather than “JavaScript compatibility,” even though JavaScript engines parse and execute... well, JavaScript. This can be a little confusing, but there is an explanation.



Implementation ↕	Applications ↕	ECMAScript edition ↕
SpiderMonkey	Firefox, the Gecko layout engine, Adobe Acrobat <sup>[d 1]</sup>	2017 <sup>[37]</sup> <sup>[d 2]</sup>
V8	Google Chrome, Node.js, Opera, MarkLogic. <sup>[38]</sup>	2016 <sup>[d 3]</sup> and features from 2017 <sup>[37]</sup>
JavaScriptCore (Nitro)	WebKit, Safari, Qt 5	2017 <sup>[39]</sup>
Chakra	Microsoft Edge	5.1 and features from 2015, <sup>[d 4]</sup> 2016 and 2017 <sup>[37]</sup>

This table is part of a browser support table in the [ECMAScript Wikipedia](#) page. JavaScript versions are not mentioned here.

If you will recall, ECMAScript is a specification for what a scripting language **could** look like. Releasing a new edition of ECMAScript does not mean that all JavaScript engines in existence suddenly have those new features. It is up to the groups or organizations who are responsible for JavaScript engines to be up-to-date about the latest ECMAScript specification, and to adopt its changes.

Therefore, developers tend to ask questions like, “What version of ECMAScript does this browser support?” or “Which ECMAScript features does this browser support?” They want to know if Google, Mozilla, and Microsoft have gotten around to updating their browsers’ JavaScript engines — for example [V8](#), [SpiderMonkey](#), and [Chakra](#), respectively — with the features described in the latest ECMAScript.

The [ECMAScript compatibility table](#) is a good resource for answering those questions.

update at one go. They incorporate the new ECMAScript features incrementally, as seen in this excerpt from Firefox's JavaScript changelog:

## Firefox 50

- The ES2015 `Symbol.hasInstance` property has been implemented ([bug 1054906](#)).
- The ES2017 `Object.getOwnPropertyDescriptors()` method has been implemented ([bug 1245024](#)).
- The behavior of `\W` in `RegExp` with `unicode` and `ignoreCase` flags is changed to match recent draft spec. Now it doesn't match to `K`, `S`, `k`, `s`, and `KELVIN SIGN (U+212A)`, and `LATIN SMALL LETTER LONG S (U+017F)` ([bug 1281739](#)).

In Firefox 50, pieces of ES2015 and ES2017 were both implemented in Firefox's JavaScript engine, SpiderMonkey. Other pieces of ES2015 and ES2017 were implemented before, and will continue to be implemented in the future.

## A JavaScript runtime

The environment in which the JavaScript code runs and is interpreted by a JavaScript engine. The runtime provides the host objects that JavaScript can operate on and work with.

**Synonyms:** Host environment



Photo credit: [Emuparadise](#)

The JavaScript runtime is the “existing entity or system” mentioned in the scripting language definition. Code passes through the JavaScript engine, and once parsed and understood, an entity or system performs the interpreted actions. A dog walks, a person runs, a video game character jumps (or in the case of the above image, wrecks).

Applications make themselves available to JavaScript scripting by providing “host objects” at runtime. For the client side, the JavaScript runtime would be the web browser, where host objects like windows and HTML documents are made available for manipulation.

document objects are not actually a part of the core JavaScript language. They are Web APIs, objects provided by a browser acting as JavaScript's host environment. For the server side, the JavaScript runtime is Node.js. Server-related host objects such as the file system, processes, and requests are provided in Node.js.

An interesting point: different JavaScript runtimes can share the same JavaScript engine. V8, for example, is the JavaScript engine used in both Google Chrome and Node.js — two very different environments.

## ECMAScript 6

It is the sixth edition of the ECMA-262 standard, and features major changes and improvements to the ECMAScript specification.

**Synonyms:** ES6, ES2015, and ECMAScript 2015



This edition of ECMAScript changed its name from ES6 to ES2015 because in 2015 Ecma International decided to switch to annual releases of ECMAScript. Accordingly, Ecma International also started to name new editions of the ECMAScript specification based on the year they are released. In short, ES6 and ES2015 are two different names for the same thing.

## **Babel**

A transpiler that can convert ES6 code to ES5 code.



Developers can use the shiny new features that come with ES6, but may be concerned with cross-browser compatibility for their web apps. At the time of the writing of this article, Edge and Internet Explorer do not fully support features from the ES6 specification.

Concerned developers can use Babel to convert their ES6 code to a functionally equivalent version that only use ES5 features. All of the major browsers fully support ES5, so they can run the code without any issues.

## One more interesting tidbit

I hope you found this information about JavaScript and ECMAScript useful. Before we wrap up things here, I'd like to share one more piece of information that needs to be clarified for fledgling web developers like me.

## Chicken or the egg

A confusing bit of history is that JavaScript was created in 1996. It was then submitted to

time, because JavaScript conformed to the ECMAScript specification, JavaScript is an example of an ECMAScript implementation.

That leaves us with this fun fact: ECMAScript is based on JavaScript, and JavaScript is based on ECMAScript.

I know.

It sounds exactly like the time-travel trope of people being their own parent — a little wonky, but kind of fun to think about.

## All good things

I know we've all had fun here, but that was a lot of information to digest. I'll take this opportunity to say farewell.

Please feel free to leave any questions, comments, suggestions, or concerns below.

Thank you very much for reading!



[Show comments](#)

Continue reading about

## JavaScript

The Myth of Inaccessible React

---

A Visual Reference For D3

---

How to Kill Your Procrastination and Absolutely Crush It With Your Ideas

---

[See all 1497 posts →](#)





#MOBILE APP DEVELOPMENT #ANDROID APP DEVELOPMENT #FIREBASE

## Why your Push Notifications never see the light of day

2 YEARS AGO



#JAVASCRIPT #TECH #DEVELOPMENT

## Quick, painless, automatic updates in Electron

2 YEARS AGO

freeCodeCamp is a donor-supported tax-exempt 501(c)(3) nonprofit organization (United States Federal Tax Identification Number: 82-0770548)



Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public. We also have thousands of freeCodeCamp study groups around the world.

Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff. You can [make a tax-deductible donation here](#).

### Our Nonprofit

About  
Donate  
Shop  
Sponsors  
Email Us

### Our Community

News  
Alumni Network  
Study Groups  
Forum  
Gitter  
GitHub  
Support  
Academic Honesty  
Code of Conduct  
Privacy Policy  
  
Terms of Service

### Our Learning Resources

Learn  
Guide  
Youtube  
Podcast  
Twitter  
Instagram