[Template:Distinguish](/wiki/Template:Distinguish" \o "Template:Distinguish) [Template:Infobox programming language](/wiki/Template:Infobox_programming_language) [Template:Infobox file format](/wiki/Template:Infobox_file_format) [Template:JavaScriptSidebar](/wiki/Template:JavaScriptSidebar)

**JavaScript** ([Template:IPAc-en](/wiki/Template:IPAc-en)[[1]](#cite_note-1)) is a [high-level](/wiki/High-level_programming_language), [dynamic](/wiki/Dynamic_programming_language), [untyped](/wiki/Programming_language#Type_system), and [interpreted](/wiki/Interpreted_language) programming language.[Template:Sfn](/wiki/Template:Sfn) It has been standardized in the [ECMAScript](/wiki/ECMAScript) language specification.[Template:Sfn](/wiki/Template:Sfn) Alongside [HTML](/wiki/HTML) and [CSS](/wiki/CSS), it is one of the three core technologies of [World Wide Web](/wiki/World_Wide_Web) content production; the majority of [websites](/wiki/Website) employ it and it is supported by all modern [Web browsers](/wiki/Web_browser) without [plug-ins](/wiki/Browser_extension).[Template:Sfn](/wiki/Template:Sfn) JavaScript is [prototype-based](/wiki/Prototype-based_programming) with [first-class functions](/wiki/First-class_function), making it a [multi-paradigm](/wiki/Multi-paradigm) language, supporting [object-oriented](/wiki/Object-oriented_programming),[[2]](#cite_note-2) [imperative](/wiki/Imperative_programming), and [functional](/wiki/Functional_programming) programming styles.[Template:Sfn](/wiki/Template:Sfn) It has an [API](/wiki/Application_programming_interface) for working with text, [arrays](/wiki/Array_data_type), dates and [regular expressions](/wiki/Regular_expression), but does not include any [I/O](/wiki/Input/output), such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.[Template:Sfn](/wiki/Template:Sfn)

Although there are strong outward similarities between JavaScript and Java, including language name, [syntax](/wiki/Syntax_(programming_languages)), and respective [standard libraries](/wiki/Standard_library), the two are distinct languages and differ greatly in their design. JavaScript was influenced by programming languages such as [Self](/wiki/Self_(programming_language)) and [Scheme](/wiki/Scheme_(programming_language)).[[3]](#cite_note-3) JavaScript is also used in environments that are not Web-based, such as [PDF](/wiki/Portable_Document_Format) documents, [site-specific browsers](/wiki/Site-specific_browser), and [desktop widgets](/wiki/Desktop_widget). Newer and faster JavaScript [virtual machines](/wiki/Virtual_machine) (VMs) and platforms built upon them have also increased the popularity of JavaScript for server-side [Web applications](/wiki/Web_application). On the client side, JavaScript has been traditionally implemented as an [interpreted](/wiki/Interpreter_(computing)) language, but more recent browsers perform [just-in-time compilation](/wiki/Just-in-time_compilation). It is also used in game development, the creation of desktop and mobile applications, and server-side network programming with runtime environments such as [Node.js](/wiki/Node.js).

## Contents

* 1 History[[edit](/index.php?title=(none)&action=edit&section=1)]
  + 1.1 Beginnings at Netscape[[edit](/index.php?title=(none)&action=edit&section=2)]
  + 1.2 Server-side JavaScript[[edit](/index.php?title=(none)&action=edit&section=3)]
  + 1.3 Adoption by Microsoft[[edit](/index.php?title=(none)&action=edit&section=4)]
  + 1.4 Standardization[[edit](/index.php?title=(none)&action=edit&section=5)]
  + 1.5 Later developments[[edit](/index.php?title=(none)&action=edit&section=6)]
* 2 Trademark[[edit](/index.php?title=(none)&action=edit&section=7)]
* 3 Features[[edit](/index.php?title=(none)&action=edit&section=8)]
  + 3.1 Imperative and structured[[edit](/index.php?title=(none)&action=edit&section=9)]
  + 3.2 Dynamic[[edit](/index.php?title=(none)&action=edit&section=10)]
  + 3.3 Prototype-based (Object-oriented)[[edit](/index.php?title=(none)&action=edit&section=11)]
  + 3.4 Functional[[edit](/index.php?title=(none)&action=edit&section=12)]
  + 3.5 Delegative[[edit](/index.php?title=(none)&action=edit&section=13)]
  + 3.6 Miscellaneous[[edit](/index.php?title=(none)&action=edit&section=14)]
  + 3.7 Vendor-specific extensions[[edit](/index.php?title=(none)&action=edit&section=15)]
* 4 Syntax[[edit](/index.php?title=(none)&action=edit&section=16)]
  + 4.1 Simple examples[[edit](/index.php?title=(none)&action=edit&section=17)]
  + 4.2 More advanced example[[edit](/index.php?title=(none)&action=edit&section=18)]
* 5 Use in Web pages[[edit](/index.php?title=(none)&action=edit&section=19)]
  + 5.1 Example script[[edit](/index.php?title=(none)&action=edit&section=20)]
* [6 This is JavaScript](#header)
* 7 tag h1 = document.getElementsByTagName('h1')[0]; // accessing the same <h1> element </script> <noscript>Your browser either does not support JavaScript, or has it turned off.</noscript> </body> </html> </syntaxhighlight> === Compatibility considerations === [[Template:Main]] Because JavaScript runs in widely varying environments, an important part of testing and debugging is to test and verify that the JavaScript works across multiple browsers. The DOM interfaces for manipulating Web pages are not part of the ECMAScript standard, or of JavaScript itself. Officially, the DOM interfaces are defined by a separate standardization effort by the [[World Wide Web Consortium|W3C]]; in practice, browser implementations differ from the standards and from each other, and not all browsers execute JavaScript. To deal with these differences, JavaScript authors can attempt to write standards-compliant code that will also be executed correctly by most browsers; failing that, they can write code that checks for the presence of certain browser features and behaves differently if they are not available.<ref>Peter-Paul Koch, [http://www.quirksmode.org/js/support.html Object detection]</ref> In some cases, two browsers may both implement a feature but with different behavior, and authors may find it practical to detect what browser is running and change their script's behavior to match.<ref>Peter-Paul Koch, [http://www.evolt.org/node/23335 Mission Impossible – mouse position]</ref><ref>Peter-Paul Koch, [http://www.quirksmode.org/js/detect.html Browser detect]</ref> Programmers may also use libraries or toolkits that take browser differences into account. Furthermore, scripts may not work for some users. For example, a user may: \* use an old or rare browser with incomplete or unusual DOM support, \* use a [[Personal digital assistant|PDA]] or [[mobile phone]] browser that cannot execute JavaScript, \* have JavaScript execution disabled as a security precaution, \* use a speech browser due to, for example, a visual disability. To support these users, Web authors can try to create pages that [[Fault-tolerant system|degrade gracefully]] on user agents (browsers) that do not support the page's JavaScript. In particular, the page should remain usable albeit without the extra features that the JavaScript would have added. An alternative approach that many find preferable is to first author content using basic technologies that work in all browsers, then enhance the content for users that have JavaScript enabled. This is known as [[progressive enhancement]]. == Security == [[Template:See also]] JavaScript and the DOM provide the potential for malicious authors to deliver scripts to run on a client computer via the Web. Browser authors contain this risk using two restrictions. First, scripts run in a [[Sandbox (computer security)|sandbox]] in which they can only perform Web-related actions, not general-purpose programming tasks like creating files. Second, scripts are constrained by the [[same origin policy]]: scripts from one Web site do not have access to information such as usernames, passwords, or cookies sent to another site. Most JavaScript-related security bugs are breaches of either the same origin policy or the sandbox. There are subsets of general JavaScript — ADsafe, Secure ECMA Script (SES) — that provide greater level of security, especially on code created by third parties (such as advertisements).<ref>[[Template:Cite web]]</ref><ref>[[Template:Cite web]]</ref> [[Caja project|Caja]] is another project for safe embedding and isolation of third-party JavaScript and HTML. [[Content Security Policy]] is the main intended method of ensuring that only trusted code is executed on a Web page. [[Template:See also]] === Cross-site vulnerabilities === [[Template:Main]] A common JavaScript-related security problem is [[cross-site scripting]], or XSS, a violation of the [[same origin policy|same-origin policy]]. XSS vulnerabilities occur when an attacker is able to cause a target Web site, such as an online banking website, to include a malicious script in the webpage presented to a victim. The script in this example can then access the banking application with the privileges of the victim, potentially disclosing secret information or transferring money without the victim's authorization. A solution to XSS vulnerabilities is to use ''HTML escaping'' whenever displaying untrusted data. Some browsers include partial protection against ''reflected'' XSS attacks, in which the attacker provides a URL including malicious script. However, even users of those browsers are vulnerable to other XSS attacks, such as those where the malicious code is stored in a database. Only correct design of Web applications on the server side can fully prevent XSS. XSS vulnerabilities can also occur because of implementation mistakes by browser authors.<ref>MozillaZine, [http://www.mozillazine.org/talkback.html?article=4392 Mozilla Cross-Site Scripting Vulnerability Reported and Fixed]</ref> Another cross-site vulnerability is [[cross-site request forgery]] or CSRF. In CSRF, code on an attacker's site tricks the victim's browser into taking actions the user didn't intend at a target site (like transferring money at a bank). It works because, if the target site relies only on cookies to authenticate requests, then requests initiated by code on the attacker's site will carry the same legitimate login credentials as requests initiated by the user. In general, the solution to CSRF is to require an authentication value in a hidden form field, and not only in the cookies, to authenticate any request that might have lasting effects. Checking the HTTP Referrer header can also help. "JavaScript hijacking" is a type of CSRF attack in which a <tt><nowiki><script></nowiki></tt> tag on an attacker's site exploits a page on the victim's site that returns private information such as JSON or JavaScript. Possible solutions include: \* requiring an authentication token in the [[POST (HTTP)|POST]] and [[GET (HTTP)|GET]] parameters for any response that returns private information ==== Misplaced trust in the client ==== Developers of client-server applications must recognize that untrusted clients may be under the control of attackers. The application author cannot assume that his JavaScript code will run as intended (or at all) because any secret embedded in the code could be extracted by a determined adversary. Some implications are: \* Web site authors cannot perfectly conceal how their JavaScript operates because the raw source code must be sent to the client. The code can be [[obfuscated code|obfuscated]], but obfuscation can be reverse-engineered. \* JavaScript form validation only provides convenience for users, not security. If a site verifies that the user agreed to its terms of service, or filters invalid characters out of fields that should only contain numbers, it must do so on the server, not only the client. \* Scripts can be selectively disabled, so JavaScript can't be relied on to prevent operations such as right-clicking on an image to save it.<ref>[[Template:Cite journal]]</ref> \* It is extremely bad practice to embed sensitive information such as passwords in JavaScript because it can be extracted by an attacker. ==== Browser and plugin coding errors ==== JavaScript provides an interface to a wide range of browser capabilities, some of which may have flaws such as [[buffer overflow]]s. These flaws can allow attackers to write scripts that would run any code they wish on the user's system. This code is not by any means limited to another JavaScript application. For example, a buffer overrun exploit can allow an attacker to gain access to the operating system's API with superuser privileges. These flaws have affected major browsers including Firefox,<ref>Mozilla Corporation, [http://www.mozilla.org/security/announce/2006/mfsa2006-38.html Buffer overflow in crypto.signText()]</ref> Internet Explorer,<ref>[[Template:Cite web]]</ref> and Safari.<ref>SecurityTracker.com, [http://securitytracker.com/alerts/2006/Mar/1015713.html Apple Safari JavaScript Buffer Overflow Lets Remote Users Execute Arbitrary Code and HTTP Redirect Bug Lets Remote Users Access Files]</ref> Plugins, such as video players, [[Adobe Flash#Flash client security|Adobe Flash]], and the wide range of [[ActiveX]] controls enabled by default in Microsoft Internet Explorer, may also have flaws exploitable via JavaScript (such flaws have been exploited in the past).<ref>SecurityFocus, [http://www.securityfocus.com/bid/19030/info Microsoft WebViewFolderIcon ActiveX Control Buffer Overflow Vulnerability]</ref><ref>Fusion Authority, [http://www.fusionauthority.com/security/3234-macromedia-flash-activex-buffer-overflow.htm Macromedia Flash ActiveX Buffer Overflow]</ref> In Windows Vista, Microsoft has attempted to contain the risks of bugs such as buffer overflows by running the Internet Explorer process with limited privileges.<ref>Mike Friedman, [http://blogs.msdn.com/ie/archive/2006/02/09/528963.aspx Protected Mode in Vista IE7]</ref> [[Google Chrome]] similarly confines its page renderers to their own "sandbox". ==== Sandbox implementation errors ==== Web browsers are capable of running JavaScript outside the sandbox, with the privileges necessary to, for example, create or delete files. Of course, such privileges aren't meant to be granted to code from the Web. Incorrectly granting privileges to JavaScript from the Web has played a role in vulnerabilities in both Internet Explorer<ref>US CERT, [https://www.kb.cert.org/vuls/id/713878 Vulnerability Note VU#713878: Microsoft Internet Explorer does not properly validate source of redirected frame]</ref> and Firefox.<ref>Mozilla Foundation, [http://www.mozilla.org/security/announce/2005/mfsa2005-41.html Mozilla Foundation Security Advisory 2005–41: Privilege escalation via DOM property overrides]</ref> In Windows XP Service Pack 2, Microsoft demoted JScript's privileges in Internet Explorer.<ref>Microsoft Corporation, [http://technet.microsoft.com/en-us/library/bb457150.aspx#EHAA Changes to Functionality in Microsoft Windows XP Service Pack 2: Part 5: Enhanced Browsing Security]</ref> [[Microsoft Windows]] allows JavaScript source files on a computer's hard drive to be launched as general-purpose, non-sandboxed programs (see: [[Windows Script Host]]). This makes JavaScript (like [[VBScript]]) a theoretically viable vector for a [[Trojan horse (computing)|Trojan horse]], although JavaScript Trojan horses are uncommon in practice.<ref>For one example of a rare JavaScript Trojan Horse, see Symantec Corporation, [http://www.symantec.com/security\_response/writeup.jsp?docid=2003-100111-0931-99 JS.Seeker.K]</ref> == Uses outside Web pages == In addition to Web browsers and servers, JavaScript interpreters are embedded in a number of tools. Each of these applications provides its own [[object model]] that provides access to the host environment. The core JavaScript language remains mostly the same in each application. === Embedded scripting language === \* Google's [[Google Chrome|Chrome]] extensions, [[Opera (Web browser)|Opera]]'s extensions, Apple's [[Safari (Web browser)|Safari 5]] extensions, Apple's [[Dashboard (Mac OS)|Dashboard Widgets]], Microsoft's [[Microsoft Gadgets|Gadgets]], [[Yahoo! Widgets]], [[Google Desktop#Gadgets and plug-ins|Google Desktop Gadgets]], and [[Serence]] [[Klipfolio]] are implemented using JavaScript. \* The [[MongoDB]] database accepts queries written in JavaScript. [[MongoDB]] and [[NodeJS]] are the core components of [[MEAN (software bundle)|MEAN]]: a [[solution stack]] for creating Web applications using just JavaScript. \* The [[Clusterpoint]] database accept queries written in JS/SQL, which is a combination of [[SQL]] and JavaScript. [[Clusterpoint]] has built-in computing engine that allows execution of JavaScript code right inside the [[distributed database]]. \* Adobe's [[Adobe Acrobat#Security|Acrobat and Adobe Reader]] support JavaScript in [[Portable Document Format|PDF]] files.<ref>[[Template:Cite web]]</ref> \* Tools in the [[Adobe Creative Suite]], including [[Adobe Photoshop|Photoshop]], [[Adobe Illustrator|Illustrator]], [[Dreamweaver]], and [[InDesign]], allow scripting through JavaScript. \* [[OpenOffice.org]], an office application suite, as well as its popular fork [[LibreOffice]], allows JavaScript to be used as a scripting language. \* The interactive music signal processing software [[Max/MSP]] released by Cycling '74, offers a JavaScript model of its environment for use by developers. It allows much more precise control than the default GUI-centric programming model. \* Apple's Logic Pro X digital audio workstation (DAW) software can create custom MIDI effects plugins using JavaScript. [[Template:Citation needed]] \* ECMAScript was included in the [[VRML]]97 standard for scripting nodes of VRML scene description files.[[Template:Citation needed]] \* The [[Unity (game engine)|Unity]] game engine supports a modified version of JavaScript for scripting via Mono.<ref>[[Template:Cite web]]</ref> \* [[DX Studio]] (3D engine) uses the [[SpiderMonkey (JavaScript engine)|SpiderMonkey]] implementation of JavaScript for game and simulation logic.<ref>[[Template:Cite web]]</ref> \* [[Maxwell Render]] ([[Rendering (computer graphics)|rendering]] software) provides an ECMA standard based scripting engine for tasks automation.<ref>THINK! The Maxwell Render Resourcer Center, [http://think.maxwellrender.com/scripting\_references-269.html Scripting References]</ref> \* [[Google Apps Script]] in [[Google Spreadsheets]] and [[Google Sites]] allows users to create custom formulas, automate repetitive tasks and also interact with other Google products such as Gmail.<ref>[[Google Apps Script]], [https://www.google.com/script/start/ Google Apps Script]</ref> \* Many [[Internet Relay Chat clients|IRC clients]], like [[ChatZilla]] or [[XChat]], use JavaScript for their scripting abilities.<ref>[[Template:Cite web]]</ref><ref>[[Template:Cite web]]</ref> \* [[RPG Maker]] MV uses Javascript as its scripting language.<ref>[[Template:Cite web]]</ref> === Scripting engine === \* Microsoft's [[Active Scripting]] technology supports [[JScript]] as a scripting language.<ref name="VersionInformation">[[Template:Cite web]]</ref> \* The [[Java (programming language)|Java programming language]] introduced the <tt>javax.script</tt> package in version 6 that includes a JavaScript implementation based on [[Rhino (JavaScript engine)|Mozilla Rhino]]. Thus, Java applications can host scripts that access the application's variables and objects, much like Web browsers host scripts that access a webpage's [[Document Object Model]] (DOM).<ref>[[Template:Cite web]]</ref>[[Template:Sfn]] \* The [[Qt (toolkit)|Qt]] C++ toolkit includes a <tt>QtScript</tt> module to interpret JavaScript, analogous to Java's <tt>javax.script</tt> package.<ref>Nokia Corporation, [http://doc.qt.nokia.com/4.6/qtscript.html QtScript Module]</ref> \* [[OS X Yosemite]] introduced JavaScript for Automation (JXA), which is built upon [[JavaScriptCore]] and the [[Open Scripting Architecture]]. It features an [[Objective-C]] bridge that enables entire [[Cocoa (API)|Cocoa]] applications to be programmed in JavaScript. \* Late Night Software's [[JavaScript OSA]] (a.k.a. JavaScript for OSA, or JSOSA) is a freeware alternative to [[AppleScript]] for Mac OS X. It is based on the Mozilla 1.5 JavaScript implementation, with the addition of a <tt>MacOS</tt> object for interaction with the operating system and third-party applications.<ref>[[Open Scripting Architecture]]</ref> === Application platform === \* [[ActionScript]], the programming language used in [[Adobe Flash]], is another implementation of the ECMAScript standard. \* [[Adobe Integrated Runtime]] is a JavaScript runtime that allows developers to create desktop applications. \* [[CA, Inc.]]'s AutoShell cross-application scripting environment is built on the [[SpiderMonkey (JavaScript engine)|SpiderMonkey]] JavaScript engine. It contains [[preprocessor]]-like extensions for command definition, as well as custom classes for various system-related tasks like file I/O, operation system command invocation and redirection, and COM scripting. \* [[GNOME Shell]], the shell for the [[Template:Nobr]] desktop environment,<ref>[[Template:Cite web]]</ref> made JavaScript its default programming language in 2013.<ref>[[Template:Cite web]]</ref> \* The [[Mozilla]] platform, which underlies [[Firefox]], [[Mozilla Thunderbird|Thunderbird]], and some other Web browsers, uses JavaScript to implement the [[graphical user interface]] (GUI) of its various products. \* [[Qt Quick]]'s markup language (available since Qt 4.7) uses JavaScript for its application logic. Its declarative syntax is also similar to JavaScript. \* [[TypeScript]] is a programming language based on JavaScript that adds support for optional [[type annotation]]s and some other language extensions such as classes, interfaces and modules. A TS-script compiles into plain JavaScript and can be executed in any JS host supporting [[ECMAScript]] 3 or higher. The compiler is itself written in TypeScript. \* [[Ubuntu Touch]] provides a JavaScript API for its unified usability interface. \* [[webOS]] uses the [[WebKit]] implementation of JavaScript in its [[Software development kit|SDK]] to allow developers to create stand-alone applications solely in JavaScript. \* [[Windows Runtime#JavaScript|WinJS]] provides a special Windows Library for JavaScript functionality in [[Windows 8]] that enables the development of [[Metro (design language)|Modern style]] (formerly ''Metro style'') applications in [[HTML5]] and JavaScript. == Development tools == Within JavaScript, access to a [[debugger]] becomes invaluable when developing large, non-trivial programs. Because there can be implementation differences between the various browsers (particularly within the [[Document Object Model]]), it is useful to have access to a debugger for each of the browsers that a Web application targets.<ref>[[Template:Cite web]]</ref> Script debuggers are integrated within [[Internet Explorer]], [[Firefox]], [[Safari (Web browser)|Safari]], [[Google Chrome]], [[Opera (Web browser)|Opera]] and [[Node.js]]<ref>[[Template:Cite web]]</ref><ref>[[Template:Cite web]]</ref><ref>[[Template:Cite web]]</ref> In addition to the native [[Internet Explorer Developer Tools]], three debuggers are available for Internet Explorer: [[Microsoft Visual Studio]] is the richest of the three, closely followed by [[Microsoft Script Editor]] (a component of [[Microsoft Office]]),<ref>[http://msdn2.microsoft.com/en-us/library/aa202668(office.11).aspx JScript development in Microsoft Office 11] (MS InfoPath 2003)</ref> and finally the free [[Microsoft Script Debugger]] that is far more basic than the other two. The free [[Microsoft Visual Web Developer Express]] provides a limited version of the JavaScript debugging functionality in Microsoft Visual Studio. Internet Explorer has included developer tools since version 8 (reached by pressing the F12 key). In comparison to Internet Explorer, Firefox has a more comprehensive set of developer tools, which include a debugger as well. Old versions of Firefox without these tools used a [[Firefox addon]] called [[Firebug (Firefox extension)|Firebug]], or the older [[Venkman]] debugger. Also, [[WebKit]]'s Web Inspector includes a JavaScript debugger,<ref>[[Template:Cite web]]</ref> which is used in [[Safari (Web browser)|Safari]]. A modified version called Blink DevTools is used in [[Google Chrome]]. [[Node.js]] has node-inspector, an interactive debugger that integrates with the Blink DevTools, available in [[Google Chrome]]. Last but not least, [[Opera (Web browser)|Opera]] includes a set of tools called [[Opera Dragonfly|Dragonfly]].<ref>[[Template:Cite web]]</ref> [[Template:Anchor]]In addition to the native computer software, there are ''online JavaScript IDEs'', debugging aids are themselves written in JavaScript and built to run on the Web. An example is the program [[JSLint]], developed by [[Douglas Crockford]] who has written extensively on the language. JSLint scans JavaScript code for conformance to a set of standards and guidelines. Many libraries for JavaScript, such as [[three.js]], provide links to demonstration code that can be edited by users. They are also used as a pedagogical tool by institutions such as [[Khan Academy]]<ref>[[Template:Cite web]]</ref> to allow students to experience writing code in an environment where they can see the output of their programs, without needing any setup beyond a Web browser. == Version history == [[Template:See also]] JavaScript was initially developed in 1996 for use in the [[Netscape Navigator]] browser. In the same year Microsoft released an implementation for Internet Explorer. This implementation was called [[JScript]] due to trademark issues. In 1997 the first standardized version of the language was released under the name [[ECMAScript]]. The following table is based on information from multiple sources.<ref>[[Template:Cite web]]</ref><ref>[[Template:Cite web]]</ref><ref>[[Template:Cite web]]</ref> {| class="wikitable" style="text-align:center;" |- ! Version !! Release date !! Equivalent to !! Netscape<br />Navigator !! Mozilla<br />Firefox !! Internet<br />Explorer !! Opera !! Safari !! Google<br />Chrome |- | [[Template:Version]] || March 1996 || || 2.0 || || 3.0 || || || |- | [[Template:Version]] || August 1996 || || 3.0 || || || || || |- | [[Template:Version]] || June 1997 || || 4.0-4.05 || || || 3<ref>[[History of the Opera web browser#Version 3]]</ref> || || |- | [[Template:Version]] || October 1998 || ECMA-262 1st + 2nd edition || 4.06-4.7x || || 4.0 || 5<ref>[[Template:Cite web]]</ref> || || |- | [[Template:Version]] || || || Netscape<br />Server || || || 6 || || |- | [[Template:Version]] || November 2000 || ECMA-262 3rd edition || 6.0 || 1.0 || 5.5 (JScript 5.5),<br />6 (JScript 5.6),<br />7 (JScript 5.7),<br />8 (JScript 5.8) || 7.0 || 3.0-5 || 1.0-10.0.666 |- | [[Template:Version]] || November 2005 || 1.5 + array extras + array and string generics + [[E4X]] || || 1.5 || || || || |- | [[Template:Version]] || October 2006 || 1.6 + [https://developer.mozilla.org/en-US/docs/JavaScript/New\_in\_JavaScript/1.7?redirectlocale=en-US&redirectslug=New\_in\_JavaScript\_1.7#Generators Pythonic generators] + iterators + let || || 2.0 || || || ||28.0.1500.95 |- | [[Template:Version]] || June 2008 || 1.7 + [[generator (computer programming)|generator expressions]] + [[closure (computer programming)|expression closures]] || || 3.0 || || 11.50 || || |- | [[Template:Version]] || || 1.8 + [[JSON#Native encoding and decoding in browsers|native JSON]] support + minor updates || || 3.5 || || || || |- | [[Template:Version]] || June 22, 2009 || 1.8.1 + minor updates || || 3.6 || || || || |- |1.8.5 |July 27, 2010 |1.8.2 + new features for ECMA-262 Edition 5 compliance. | |4.0 | | | | |} == Related languages and features == [[JSON]], or JavaScript Object Notation, is a general-purpose data interchange format that is defined as a subset of JavaScript's object literal syntax. Like much of JavaScript (regexps and anonymous functions as 1st class elements, closures, flexible classes, 'use strict'), [[JSON]], except for replacing [[Perl]]'s key-value operator '=>' by an [[RFC 822]]<ref>[[Template:Cite web]]</ref> inspired ':', is syntactically pure Perl. [[jQuery]] is a popular JavaScript library designed to simplify [[Document Object Model|DOM]]-oriented client-side HTML scripting along with offering cross-browser compatibility because various browsers respond differently to certain vanilla JavaScript code. [[Underscore.js]] is a utility JavaScript library for data manipulation that is used in both client-side and server-side network applications. Mozilla browsers currently support [[LiveConnect]], a feature that allows JavaScript and Java to intercommunicate on the Web. However, Mozilla-specific support for LiveConnect is scheduled to be phased out in the future in favor of passing on the LiveConnect handling via [[NPAPI]] to the Java 1.6+ plug-in (not yet supported on the Mac [[Template:As of]]).<ref>[http://java.sun.com/javase/6/webnotes/6u10/plugin2/liveconnect/ Release Notes for the Next-Generation Java™ Plug-In Technology (introduced in Java SE 6 update 10)]. Java.sun.com. Retrieved on 2013-06-13.</ref> Most browser inspection tools, such as [[Firebug (software)|Firebug]] in Firefox, include JavaScript interpreters that can act on the visible page's DOM. [[asm.js]] is a subset of JavaScript that can be run in any JavaScript engine or run faster in an [[ahead-of-time]] (AOT) compiling engine.<ref>[[Template:Cite web]]</ref> [[JSFuck]] is an [[esoteric programming language]]. Programs are written using only six different characters, but are still valid JavaScript code. p5.js<ref>[[Template:Cite web]]</ref> is an object oriented JavaScript library designed for artists and designers. It is based on the ideas of the [[Processing (programming language)|Processing]] project but is for the web. === Use as an intermediate language === As JavaScript is the most widely supported client-side language that can run within a Web browser, it has become an [[intermediate language]] for other languages to target. This has included both newly created languages and ports of existing languages. Some of these include: \* Oberon Script, a full implementation of the [[Oberon (programming language)|Oberon Programming Language]] that compiles to high-level JavaScript.<ref>[[Template:Cite web]]</ref> \* [[Objective-J]], a superset of JavaScript that compiles to standard JavaScript. It adds traditional inheritance and [[Smalltalk]]/[[Objective-C]] style dynamic dispatch and optional pseudo-static typing to JavaScript. \* [[Processing.js]], a JavaScript port of Processing, a programming language designed to write visualizations, images, and interactive content. It allows Web browsers to display animations, visual applications, games and other graphical rich content without the need for a Java applet or Flash plugin. \* [[CoffeeScript]], an alternate syntax for JavaScript intended to be more concise and readable. It adds features like array comprehensions (also available in JavaScript since version 1.7)<ref>[[Template:Cite web]]</ref> and pattern matching. Like Objective-J, it compiles to JavaScript. Ruby and Python have been cited as influential on CoffeeScript syntax. \* [[Google Web Toolkit]] translates a subset of Java to JavaScript. \* [[Scala (programming language)|Scala]], an object-oriented and functional programming language, has a Scala-to-JavaScript compiler.<ref>[[Template:Cite web]]</ref> \* [[Pyjamas (software)|Pyjamas]], a port of [[Google Web Toolkit]] to [[Python (programming language)|Python]] (translates a subset of Python to JavaScript) \* [[Dart (programming language)|Dart]], an open-source programming language developed by Google, can be compiled to JavaScript. \* Whalesong,<ref>[[Template:Cite web]]</ref> a [[Racket (programming language)|Racket]]-to-JavaScript compiler. \* [[Emscripten]], a [[LLVM]]-backend for porting native libraries to JavaScript. \* [[Fantom (programming language)|Fantom]] a programming language that runs on JVM, .NET and JavaScript. \* [[TypeScript]], a free and open-source programming language developed by Microsoft. It is a superset of JavaScript, and essentially adds optional static typing and class-based object-oriented programming to the language. \* [[Haxe]], an open-source high-level multiplatform programming language and compiler that can produce applications and source code for many different platforms including JavaScript. \* ClojureScript,<ref>[[Template:Cite web]]</ref> a compiler for [[Clojure]] that targets JavaScript. It is designed to emit JavaScript code that is compatible with the advanced compilation mode of the Google Closure optimizing compiler. \* [[Kotlin (programming language)|Kotlin]], a [[Type system#Static type-checking|statically-typed]] language that also compiles to [[Java byte code]]. As JavaScript has unusual limitations – such as no separate integer type, using floating point – languages that compile to JavaScript commonly have slightly different behavior than in other environments. === JavaScript and Java === A common misconception is that JavaScript is similar or closely related to [[Java (programming language)|Java]]. It is true that both have a C-like syntax (the C language being their most immediate common ancestor language). They also are both typically [[Sandbox (computer security)|sandboxed]] (when used inside a browser), and JavaScript was designed with Java's syntax and standard library in mind. In particular, all Java keywords were reserved in original JavaScript, JavaScript's standard library follows Java's naming conventions, and JavaScript's Math and Date objects are based on classes from Java 1.0,<ref name="popularity">[[Template:Cite web]]</ref> but the similarities end there. The differences between the two languages are more prominent than their similarities. Java has [[static typing]], while JavaScript's typing is [[Dynamic typing|dynamic]]. Java is loaded from compiled bytecode, while JavaScript is loaded as human-readable source code. Java's objects are [[Class-based programming|class-based]], while JavaScript's are [[Prototype-based programming|prototype-based]]. Finally, Java did not support [[functional programming]] until Java 8, while JavaScript has done so from the beginning, being influenced by [[Scheme (programming language)|Scheme]]. == References == [[Template:Reflist]] == Further reading == [[Template:Refbegin]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] [[Template:Refend]] == External links == [[Template:Portal]] [[Template:Sisterlinks]] [[Template:Spoken Wikipedia]] \* [[Douglas Crockford]]'s [https://www.youtube.com/playlist?list=PL62E185BB8577B63D video lectures on JavaScript] \* Douglas Crockford's [http://javascript.crockford.com/survey.html A Survey of the JavaScript Programming Language] \* [[Template:DMOZ]] \* [https://github.com/jashkenas/coffee-script/wiki/List-of-languages-that-compile-to-JS/ List of languages that compile to JS] [[Template:JavaScript]] [[Template:Programming languages]] [[Template:ECMAScript]] [[Template:Web browsers]] [[Template:NodeJs]] [[Template:Authority control]] [[Category:1995 introductions]] [[Category:American inventions]] [[Category:Articles with example JavaScript code]] [[Category:Cross-platform software]] [[Category:Dynamically typed programming languages]] [[Category:Functional languages]] [[Category:JavaScript| ]] [[Category:Object-based programming languages]] [[Category:Programming languages created in 1995]] [[Category:Programming languages with an ISO standard]] [[Category:Prototype-based programming languages]] [[Category:Scripting languages]] [[Category:Web programming]]

## History[[edit](/index.php?title=(none)&action=edit&section=1)]

### Beginnings at Netscape[[edit](/index.php?title=(none)&action=edit&section=2)]

JavaScript was originally developed in 10 days in May 1995 by [Brendan Eich](/wiki/Brendan_Eich), while he was working for [Netscape Communications Corporation](/wiki/Netscape_Communications_Corporation). Indeed, while competing with [Microsoft](/wiki/Microsoft) for user adoption of Web technologies and platforms, Netscape considered their client-server offering a distributed OS with a portable version of [Sun Microsystems's](/wiki/Sun_Microsystems) Java providing an environment in which [applets](/wiki/Java_applet) could be run.[Template:Citation needed](/wiki/Template:Citation_needed) Because Java was a competitor of [C++](/wiki/C++) and aimed at professional programmers, Netscape also wanted a lightweight [interpreted language](/wiki/Interpreted_language) that would complement Java by appealing to nonprofessional programmers, like Microsoft's Visual Basic (see [JavaScript and Java](/wiki/#JavaScript_and_Java)).[[4]](#cite_note-4) Although it was developed under the name **Mocha**, the language was officially called **LiveScript** when it first shipped in beta releases of Netscape Navigator 2.0 in September 1995, but it was renamed JavaScript[[5]](#cite_note-5) when it was deployed in the Netscape browser version 2.0B3.[[6]](#cite_note-6) The change of name from LiveScript to JavaScript roughly coincided with Netscape adding support for Java technology in its [Netscape Navigator](/wiki/Netscape_Navigator) Web browser. The final choice of name caused confusion, giving the impression that the language was a spin-off of the Java programming language, and the choice has been characterized as a marketing ploy by Netscape to give JavaScript the cachet of what was then the hot new Web programming language.[[7]](#cite_note-7)[[8]](#cite_note-8) There is a common misconception that the JavaScript language was influenced by an earlier Web page scripting language developed by [Nombas](/wiki/Openwave) named C-- (not to be confused with the later [C--](/wiki/C--) created in 1997).[[9]](#cite_note-9)[[10]](#cite_note-10)[[11]](#cite_note-11) Brendan Eich, however, had never heard of C-- before he created LiveScript.[[12]](#cite_note-12) Nombas did pitch their embedded Web page scripting to Netscape, though Web page scripting was not a new concept, as shown by [ViolaWWW](/wiki/ViolaWWW).[[13]](#cite_note-13) Nombas later switched to offering JavaScript instead of C-- in their ScriptEase product and was part of the TC39 group that standardized [ECMAScript](/wiki/ECMAScript).[[14]](#cite_note-14)[[15]](#cite_note-15)

### Server-side JavaScript[[edit](/index.php?title=(none)&action=edit&section=3)]

Netscape introduced an implementation of the language for [server-side scripting](/wiki/Server-side_scripting) with [Netscape Enterprise Server](/wiki/Netscape_Enterprise_Server) in December, 1995, soon after releasing JavaScript for browsers.[[16]](#cite_note-16)[[17]](#cite_note-17)Since the mid-2000s, there has been a resurgence of [server-side JavaScript implementations](/wiki/Comparison_of_server-side_JavaScript_solutions), such as [Node.js](/wiki/Node.js).[[18]](#cite_note-18)[[19]](#cite_note-19)

### Adoption by Microsoft[[edit](/index.php?title=(none)&action=edit&section=4)]

[Microsoft Windows](/wiki/Microsoft_Windows) script technologies including [VBScript](/wiki/VBScript) and [JScript](/wiki/JScript) were released in 1996. JScript, a reverse-engineered implementation of Netscape's JavaScript, was released on July 16, 1996 and was part of [Internet Explorer 3](/wiki/Internet_Explorer_3), as well as being available server-side in [Internet Information Server](/wiki/Internet_Information_Server). IE3 also included Microsoft's first support for [Cascading Style Sheets](/wiki/Cascading_Style_Sheets) and various extensions to HTML, but in each case the implementation was noticeably different to that found in [Netscape Navigator](/wiki/Netscape_Navigator) at the time.[[20]](#cite_note-20)[[21]](#cite_note-21) These differences made it difficult for designers and programmers to make a single website work well in both browsers leading to the use of 'best viewed in Netscape' and 'best viewed in Internet Explorer' logos that characterised these early years of the [browser wars](/wiki/Browser_wars).[[22]](#cite_note-22) JavaScript began to acquire a reputation for being one of the roadblocks to a cross-platform and standards-driven Web. Some developers took on the difficult task of trying to make their sites work in both major browsers, but many could not afford the time.[[20]](#cite_note-20) With the release of [Internet Explorer 4](/wiki/Internet_Explorer_4), Microsoft introduced the concept of [Dynamic HTML](/wiki/Dynamic_HTML), but the differences in language implementations and the different and proprietary [Document Object Models](/wiki/Document_Object_Model) remained, and were obstacles to widespread take-up of JavaScript on the Web.[[20]](#cite_note-20)

### Standardization[[edit](/index.php?title=(none)&action=edit&section=5)]

In November 1996, Netscape announced that it had submitted JavaScript to [Ecma International](/wiki/Ecma_International) for consideration as an industry standard, and subsequent work resulted in the standardized version named [ECMAScript](/wiki/ECMAScript). In June 1997, [Ecma International](/wiki/Ecma_International) published the first edition of the [ECMA-262](/wiki/ECMA-262) [specification](/wiki/Specification). In June 1998, some modifications were made to adapt it to the ISO/IEC-16262 standard, and the second edition was released. The third edition of [ECMA-262](/wiki/ECMA-262) was published on December 1999.[[23]](#cite_note-23) Development of the fourth edition of the ECMAScript standard was never completed.[[24]](#cite_note-24) The fifth edition was released in December 2009. The current edition of the ECMAScript standard is 6, released in June 2015.[[25]](#cite_note-25)

### Later developments[[edit](/index.php?title=(none)&action=edit&section=6)]

JavaScript has become one of the most popular programming languages on the Web. Initially, however, many professional programmers denigrated the language because its target audience consisted of Web authors and other such "amateurs", among other reasons.[[26]](#cite_note-26) The advent of [Ajax](/wiki/Ajax_(programming)) returned JavaScript to the spotlight and brought more professional programming attention. The result was a proliferation of comprehensive [frameworks and libraries](/wiki/List_of_JavaScript_libraries), improved JavaScript programming practices, and increased usage of JavaScript outside Web browsers, as seen by the proliferation of [server-side JavaScript](/wiki/Server-side_JavaScript) platforms.

In January 2009, the [CommonJS](/wiki/CommonJS) project was founded with the goal of specifying a common standard library mainly for JavaScript development outside the browser.[[27]](#cite_note-27) With the rise of the [single-page application](/wiki/Single-page_application) and JavaScript-heavy sites, it is increasingly being used as a compile target for [source-to-source compilers](/wiki/Source-to-source_compiler) from both [dynamic languages](/wiki/Dynamic_typing) and [static languages](/wiki/Static_typing).

## Trademark[[edit](/index.php?title=(none)&action=edit&section=7)]

"JavaScript" is a [trademark](/wiki/Trademark) of [Oracle Corporation](/wiki/Oracle_Corporation).[[28]](#cite_note-28) It is used under license for technology invented and implemented by Netscape Communications and current entities such as the [Mozilla Foundation](/wiki/Mozilla_Foundation).[[29]](#cite_note-29)

## Features[[edit](/index.php?title=(none)&action=edit&section=8)]

The following features are common to all conforming ECMAScript implementations, unless explicitly specified otherwise.

### Imperative and structured[[edit](/index.php?title=(none)&action=edit&section=9)]

JavaScript supports much of the [structured programming](/wiki/Structured_programming) syntax from [C](/wiki/C_(computer_language)) (e.g., if statements, while loops, switch statements, do while loops, etc.). One partial exception is [scoping](/wiki/Scope_(computer_science)): JavaScript originally had only [function scoping](/wiki/Function_scoping) with var. ECMAScript 2015 adds a let keyword for block scoping, meaning JavaScript now has both function and block scoping. Like C, JavaScript makes a distinction between [expressions](/wiki/Expression_(computer_science)) and [statements](/wiki/Statement_(computer_science)). One syntactic difference from C is [automatic semicolon insertion](/wiki/Defensive_semicolon), which allows the semicolons that would normally terminate statements to be omitted.[Template:Sfn](/wiki/Template:Sfn)

### Dynamic[[edit](/index.php?title=(none)&action=edit&section=10)]

Typing

As with most [scripting languages](/wiki/Scripting_language), JavaScript is [dynamically typed](/wiki/Dynamic_typing); a [type](/wiki/Type_system) is associated with each [value](/wiki/Value_(computer_science)), rather than just with each expression. For example, a [variable](/wiki/Variable_(programming)) that is at one time bound to a number may later be re-bound to a [string](/wiki/String_(computer_science)).[[30]](#cite_note-30) JavaScript supports various ways to test the type of an object, including [duck typing](/wiki/Duck_typing).[Template:Sfn](/wiki/Template:Sfn)

Run-time evaluation

JavaScript includes an [eval](/wiki/Eval) function that can execute statements provided as strings at run-time.

### Prototype-based (Object-oriented)[[edit](/index.php?title=(none)&action=edit&section=11)]

JavaScript is almost entirely [object-based](/wiki/Object-based). In JavaScript, an [object](/wiki/Object_(computer_science)) is an [associative array](/wiki/Associative_array), augmented with a prototype (see below); each string key provides the name for an object property, and there are two syntactical ways to specify such a name: dot notation (obj.x = 10) and bracket notation (obj['x'] = 10). A property may be added, rebound, or deleted at run-time. Most properties of an object (and any property that belongs to an object's prototype inheritance chain) can be enumerated using a for...in loop.

JavaScript has a small number of built-in objects, including [Template:Mono](/wiki/Template:Mono) and [Template:Mono](/wiki/Template:Mono).

Prototypes

JavaScript uses [prototypes](/wiki/Prototype-based_programming) where many other object-oriented languages use [classes](/wiki/Class_(computer_science)) for [inheritance](/wiki/Inheritance_(computer_science)).[[31]](#cite_note-31) It is possible to simulate many class-based features with prototypes in JavaScript.[[32]](#cite_note-32); Functions as object constructors: Functions double as object constructors along with their typical role. Prefixing a function call with *new* will create an instance of a prototype, inheriting properties and methods from the constructor (including properties from the Object prototype).[[33]](#cite_note-33) ECMAScript 5 offers the Object.create method, allowing explicit creation of an instance without automatically inheriting from the Object prototype (older environments can assign the prototype to null).[[34]](#cite_note-34) The constructor's prototype property determines the object used for the new object's internal prototype. New methods can be added by modifying the prototype of the function used as a constructor. JavaScript's built-in constructors, such as Array or Object, also have prototypes that can be modified. While it is possible to modify the Object prototype, it is generally considered bad practice because most objects in JavaScript will inherit methods and properties from the Object prototype and they may not expect the prototype to be modified.[[35]](#cite_note-35); Functions as methods: Unlike many object-oriented languages, there is no distinction between a function definition and a [method](/wiki/Method_(computer_science)) definition. Rather, the distinction occurs during function calling; when a function is called as a method of an object, the function's local *this* keyword is bound to that object for that invocation.

### Functional[[edit](/index.php?title=(none)&action=edit&section=12)]

A [function](/wiki/Subroutine) is [first-class](/wiki/First-class_function); a function is considered to be an object. As such, a function may have properties and methods, such as .call() and .bind().[[36]](#cite_note-36) A *nested* function is a function defined within another function. It is created each time the outer function is invoked. In addition, each nested function forms a [lexical closure](/wiki/Closure_(computer_programming)): The [lexical scope](/wiki/Scope_(programming)#Lexical_scoping_vs._dynamic_scoping) of the outer function (including any constant, local variable, or argument value) becomes part of the internal state of each inner function object, even after execution of the outer function concludes.[Template:Sfn](/wiki/Template:Sfn) JavaScript also supports [anonymous functions](/wiki/Anonymous_function).

### Delegative[[edit](/index.php?title=(none)&action=edit&section=13)]

JavaScript supports implicit and explicit [delegation](/wiki/Delegation_(object-oriented_programming)).

Functions as Roles (Traits and Mixins)

JavaScript natively supports various function-based implementations of [Role](/wiki/Role-oriented_programming)[[37]](#cite_note-37) patterns like [Traits](/wiki/Traits_(computer_science))[[38]](#cite_note-38)[[39]](#cite_note-39) and [Mixins](/wiki/Mixin).[[40]](#cite_note-40) Such a function defines additional behavior by at least one method bound to the this keyword within its function body. A Role then has to be delegated explicitly via call or apply to objects that need to feature additional behavior that is not shared via the prototype chain.

Object Composition and Inheritance

Whereas explicit function-based delegation does cover [composition](/wiki/Object_composition) in JavaScript, implicit delegation already happens every time the prototype chain is walked in order to, e.g., find a method that might be related to but is not directly owned by an object. Once the method is found it gets called within this object's context. Thus [inheritance](/wiki/Inheritance_(computer_science)) in JavaScript is covered by a delegation automatism that is bound to the prototype property of constructor functions.

### Miscellaneous[[edit](/index.php?title=(none)&action=edit&section=14)]

Run-time environment

JavaScript typically relies on a run-time environment (e.g., a [Web browser](/wiki/Web_browser)) to provide objects and methods by which scripts can interact with the environment (e.g., a webpage DOM). It also relies on the run-time environment to provide the ability to include/import scripts (e.g., [HTML](/wiki/HTML) <script> elements). This is not a language feature per se, but it is common in most JavaScript implementations.

JavaScript processes [messages](/wiki/Message_(computer_science)) from a [queue](/wiki/Queue_(abstract_data_type)) one at a time. Upon loading a new message, JavaScript calls a [function](/wiki/Subroutine) associated with that message, which creates a [call stack](/wiki/Call_stack) frame (the function's [arguments](/wiki/Parameter_(computer_programming)) and [local variables](/wiki/Local_variable)). The call stack shrinks and grows based on the function's needs. Upon function completion, when the stack is empty, JavaScript proceeds to the next message in the queue. This is called the [event loop](/wiki/Event_loop), described as "run to completion" because each message is fully processed before the next message is considered. However, the language's [concurrency model](/wiki/Concurrency_(computer_science)) describes the event loop as [non-blocking](/wiki/Asynchronous_I/O): program [input/output](/wiki/Input/output) is performed using [events](/wiki/Event_(computing)) and [callback functions](/wiki/Callback_(computer_programming)). This means, for instance, that JavaScript can process a mouse click while waiting for a database query to return information.[[41]](#cite_note-41)

Variadic functions

An indefinite number of parameters can be passed to a function. The function can access them through [formal parameters](/wiki/Formal_parameter) and also through the local [Template:Mono](/wiki/Template:Mono) object. [Variadic functions](/wiki/Variadic_functions) can also be created by using the [bind](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function/bind) method.

Array and object literals

Like many scripting languages, arrays and objects ([associative arrays](/wiki/Associative_arrays) in other languages) can each be created with a succinct shortcut syntax. In fact, these [literals](/wiki/Object_literal) form the basis of the [JSON](/wiki/JSON) data format.

Regular expressions

JavaScript also supports [regular expressions](/wiki/Regular_expression) in a manner similar to [Perl](/wiki/Perl), which provide a concise and powerful syntax for text manipulation that is more sophisticated than the built-in string functions.[[42]](#cite_note-42)

### Vendor-specific extensions[[edit](/index.php?title=(none)&action=edit&section=15)]

JavaScript is officially managed by [Mozilla Foundation](/wiki/Mozilla_Foundation), and new language features are added periodically. However, only some [JavaScript engines](/wiki/JavaScript_engine) support these new features:

* property getter and setter functions (supported by WebKit, Gecko, Opera,[[43]](#cite_note-43) ActionScript, and Rhino)[[44]](#cite_note-44)\* conditional catch clauses
* iterator protocol (adopted from [Python](/wiki/Python_(programming_language)))
* shallow [generators](/wiki/Generator_(computer_programming))-[coroutines](/wiki/Coroutine) (adopted from Python)
* [array comprehensions](/wiki/List_comprehension) and generator expressions (adopted from Python)
* proper block scope via the let keyword
* array and object destructuring (limited form of [pattern matching](/wiki/Pattern_matching))
* concise function expressions (function(args) expr)
* [ECMAScript for XML](/wiki/ECMAScript_for_XML) (E4X), an extension that adds native XML support to ECMAScript (unsupported in Firefox since version 21[[45]](#cite_note-45))

## Syntax[[edit](/index.php?title=(none)&action=edit&section=16)]

[Template:Main](/wiki/Template:Main)

### Simple examples[[edit](/index.php?title=(none)&action=edit&section=17)]

[Variables](/wiki/Variable_(computer_science)) in JavaScript can be defined using the var keyword: [[46]](#cite_note-46) <syntaxhighlight lang="javascript"> var x; // defines the variable x, the special value “undefined” (not to be confused with an undefined value) is assigned to it by default var y = 2; // defines the variable y and assigns the value of 2 to it </syntaxhighlight>

Note the [comments](/wiki/Comment_(computer_programming)) in the example above, both of which were preceded with two [forward slashes](/wiki/Slash_(punctuation)).

There is no built-in [I/O](/wiki/Input/output) functionality in JavaScript; the run-time environment provides that. The ECMAScript specification in edition 5.1 mentions: [[47]](#cite_note-47)

… indeed, there are no provisions in this specification for input of external data or output of computed results.

However, most runtime environments have a console object [[48]](#cite_note-48) that can be used to print output. Here is a minimalist [Hello World program](/wiki/Hello_World_program):

<syntaxhighlight lang="javascript"> console.log("Hello World!"); </syntaxhighlight>

A simple [recursive](/wiki/Recursion_(computer_science)) function: <syntaxhighlight lang="javascript"> function factorial(n) {

if (n == 0) {

return 1;

}

return n\*factorial(n - 1);

} </syntaxhighlight>

[Anonymous function](/wiki/Anonymous_function) (or lambda) syntax and [closure](/wiki/Closure_(computer_programming)) example: <syntaxhighlight lang="javascript"> var displayClosure = function() {

var count = 0;

return function () {

return ++count;

};

} var inc = displayClosure(); inc(); // returns 1 inc(); // returns 2 inc(); // returns 3 </syntaxhighlight>

[Variadic function](/wiki/Variadic_function) demonstration (arguments is a special [variable](/wiki/Variable_(programming))). [[49]](#cite_note-49)<syntaxhighlight lang="javascript"> var sum = function() {

var i, x = 0;

for (i = 0; i < arguments.length; ++i) {

x += arguments[i];

}

return x;

} sum(1, 2, 3); // returns 6 </syntaxhighlight>

[Immediately-invoked function expressions](/wiki/Immediately-invoked_function_expression) allow functions to pass around variables under their own closures.

<syntaxhighlight lang="JavaScript"> var v; v = 1; var getValue = (function(v) {

return function() {return v;};

})(v);

v = 2;

getValue(); // 1 </syntaxhighlight>

### More advanced example[[edit](/index.php?title=(none)&action=edit&section=18)]

This sample code displays various JavaScript features.

<syntaxhighlight lang="javascript"> /\* Finds the lowest common multiple (LCM) of two numbers \*/ function LCMCalculator(x, y) { // constructor function

var checkInt = function (x) { // inner function

if (x % 1 !== 0) {

throw new TypeError(x + " is not an integer"); // throw an exception

}

return x;

};

this.a = checkInt(x)

// semicolons ^^^^ are optional, a newline is enough

this.b = checkInt(y);

} // The prototype of object instances created by a constructor is // that constructor's "prototype" property. LCMCalculator.prototype = { // object literal

constructor: LCMCalculator, // when reassigning a prototype, set the constructor property appropriately

gcd: function () { // method that calculates the greatest common divisor

// Euclidean algorithm:

var a = Math.abs(this.a), b = Math.abs(this.b), t;

if (a < b) {

// swap variables

t = b;

b = a;

a = t;

}

while (b !== 0) {

t = b;

b = a % b;

a = t;

}

// Only need to calculate GCD once, so "redefine" this method.

// (Actually not redefinition—it's defined on the instance itself,

// so that this.gcd refers to this "redefinition" instead of LCMCalculator.prototype.gcd.

// Note that this leads to a wrong result if the LCMCalculator object members "a" and/or "b" are altered afterwards.)

// Also, 'gcd' === "gcd", this['gcd'] === this.gcd

this['gcd'] = function () {

return a;

};

return a;

},

// Object property names can be specified by strings delimited by double (") or single (') quotes.

lcm : function () {

// Variable names don't collide with object properties, e.g., |lcm| is not |this.lcm|.

// not using |this.a\*this.b| to avoid FP precision issues

var lcm = this.a/this.gcd()\*this.b;

// Only need to calculate lcm once, so "redefine" this method.

this.lcm = function () {

return lcm;

};

return lcm;

},

toString: function () {

return "LCMCalculator: a = " + this.a + ", b = " + this.b;

}

};

// Define generic output function; this implementation only works for Web browsers function output(x) {

document.body.appendChild(document.createTextNode(x));

document.body.appendChild(document.createElement('br'));

}

// Note: Array's map() and forEach() are defined in JavaScript 1.6. // They are used here to demonstrate JavaScript's inherent functional nature. [25, 55], [21, 56], [22, 58], [28, 56](/wiki/25,_55%5d,_%5b21,_56%5d,_%5b22,_58%5d,_%5b28,_56).map(function (pair) { // array literal + mapping function

return new LCMCalculator(pair[0], pair[1]);

}).sort(function (a, b) { // sort with this comparative function

return a.lcm() - b.lcm();

}).forEach(function (obj) {

output(obj + ", gcd = " + obj.gcd() + ", lcm = " + obj.lcm());

}); </syntaxhighlight>

The following output should be displayed in the browser window. <syntaxhighlight lang="html4strict"> LCMCalculator: a = 28, b = 56, gcd = 28, lcm = 56 LCMCalculator: a = 21, b = 56, gcd = 7, lcm = 168 LCMCalculator: a = 25, b = 55, gcd = 5, lcm = 275 LCMCalculator: a = 22, b = 58, gcd = 2, lcm = 638 </syntaxhighlight>

## Use in Web pages[[edit](/index.php?title=(none)&action=edit&section=19)]

[Template:See also](/wiki/Template:See_also)

The most common use of JavaScript is to add client-side behavior to [HTML](/wiki/HTML) pages, a.k.a. [Dynamic HTML](/wiki/Dynamic_HTML) (DHTML). Scripts are embedded in or included from [HTML](/wiki/HTML) pages and interact with the [Document Object Model](/wiki/Document_Object_Model) (DOM) of the page. Some simple examples of this usage are:

* Loading new page content or submitting data to the server via [AJAX](/wiki/AJAX) without reloading the page (for example, a social network might allow the user to post status updates without leaving the page)
* Animation of page elements, fading them in and out, resizing them, moving them, etc.
* Interactive content, for example games, and playing audio and video
* [Validating](/wiki/Data_validation) input values of a [Web form](/wiki/Form_(HTML)) to make sure that they are acceptable before being submitted to the server.
* Transmitting information about the user's reading habits and browsing activities to various websites. Web pages frequently do this for [Web analytics](/wiki/Web_analytics), [ad tracking](/wiki/Ad_tracking), [personalization](/wiki/Personalization) or other purposes.[[50]](#cite_note-50)

Because JavaScript code can run locally in a user's browser (rather than on a remote server), the browser can respond to user actions quickly, making an application more responsive. Furthermore, JavaScript code can detect user actions that HTML alone cannot, such as individual keystrokes. Applications such as [Gmail](/wiki/Gmail) take advantage of this: much of the user-interface logic is written in JavaScript, and JavaScript dispatches requests for information (such as the content of an e-mail message) to the server. The wider trend of [Ajax](/wiki/Ajax_(programming)) programming similarly exploits this strength.

A [JavaScript engine](/wiki/JavaScript_engine) (also known as *JavaScript interpreter* or *JavaScript implementation*) is an [interpreter](/wiki/Interpreter_(computing)) that interprets JavaScript [source code](/wiki/Source_code) and executes the [script](/wiki/Computer_program) accordingly. The first JavaScript engine was created by [Brendan Eich](/wiki/Brendan_Eich) at [Netscape Communications Corporation](/wiki/Netscape_Communications_Corporation), for the [Netscape Navigator](/wiki/Netscape_Navigator) [Web browser](/wiki/Web_browser). The engine, code-named [SpiderMonkey](/wiki/SpiderMonkey_(JavaScript_engine)), is implemented in [C](/wiki/C_(programming_language)). It has since been updated (in JavaScript 1.5) to conform to ECMA-262 Edition 3. The [Rhino](/wiki/Rhino_(JavaScript_engine)) engine, created primarily by Norris Boyd (formerly of Netscape; now at Google) is a JavaScript implementation in [Java](/wiki/Java_(programming_language)). Rhino, like SpiderMonkey, is ECMA-262 Edition 3 compliant.

A Web browser is by far the most common host environment for JavaScript. Web browsers typically create "host objects" to represent the [Document Object Model](/wiki/Document_Object_Model) (DOM) in JavaScript. The [Web server](/wiki/Web_server) is another common host environment. A [JavaScript Web server](/wiki/Server-side_JavaScript) would typically expose host objects representing [HTTP](/wiki/HTTP) request and response objects, which a JavaScript program could then interrogate and manipulate to dynamically generate Web pages.

Because JavaScript is the only language that the most popular browsers share support for, it has become a [target language](/wiki/Target_language_(computing)) for many frameworks in other languages, even though JavaScript was never intended to be such a language.[[51]](#cite_note-51) Despite the performance limitations inherent to its dynamic nature, the increasing speed of JavaScript engines has made the language a surprisingly feasible compilation target.

### Example script[[edit](/index.php?title=(none)&action=edit&section=20)]

Below is a minimal example of a standards-conforming Web page containing JavaScript (using [HTML](/wiki/HTML) 5 syntax) and the [DOM](/wiki/Document_object_model):

<syntaxhighlight lang="html5"> <!DOCTYPE html> <html> <meta charset="utf-8"> <title>Minimal Example</title>

<body>

# This is JavaScript

<script> document.body.appendChild(document.createTextNode('Hello World!'));

var h1 = document.getElementById('header'); // holds a reference to the

# tag h1 = document.getElementsByTagName('h1')[0]; // accessing the same <h1> element </script> <noscript>Your browser either does not support JavaScript, or has it turned off.</noscript> </body> </html> </syntaxhighlight> === Compatibility considerations === [[Template:Main]] Because JavaScript runs in widely varying environments, an important part of testing and debugging is to test and verify that the JavaScript works across multiple browsers. The DOM interfaces for manipulating Web pages are not part of the ECMAScript standard, or of JavaScript itself. Officially, the DOM interfaces are defined by a separate standardization effort by the [[World Wide Web Consortium|W3C]]; in practice, browser implementations differ from the standards and from each other, and not all browsers execute JavaScript. To deal with these differences, JavaScript authors can attempt to write standards-compliant code that will also be executed correctly by most browsers; failing that, they can write code that checks for the presence of certain browser features and behaves differently if they are not available.<ref>Peter-Paul Koch, [http://www.quirksmode.org/js/support.html Object detection]</ref> In some cases, two browsers may both implement a feature but with different behavior, and authors may find it practical to detect what browser is running and change their script's behavior to match.<ref>Peter-Paul Koch, [http://www.evolt.org/node/23335 Mission Impossible – mouse position]</ref><ref>Peter-Paul Koch, [http://www.quirksmode.org/js/detect.html Browser detect]</ref> Programmers may also use libraries or toolkits that take browser differences into account. Furthermore, scripts may not work for some users. For example, a user may: \* use an old or rare browser with incomplete or unusual DOM support, \* use a [[Personal digital assistant|PDA]] or [[mobile phone]] browser that cannot execute JavaScript, \* have JavaScript execution disabled as a security precaution, \* use a speech browser due to, for example, a visual disability. To support these users, Web authors can try to create pages that [[Fault-tolerant system|degrade gracefully]] on user agents (browsers) that do not support the page's JavaScript. In particular, the page should remain usable albeit without the extra features that the JavaScript would have added. An alternative approach that many find preferable is to first author content using basic technologies that work in all browsers, then enhance the content for users that have JavaScript enabled. This is known as [[progressive enhancement]]. == Security == [[Template:See also]] JavaScript and the DOM provide the potential for malicious authors to deliver scripts to run on a client computer via the Web. Browser authors contain this risk using two restrictions. First, scripts run in a [[Sandbox (computer security)|sandbox]] in which they can only perform Web-related actions, not general-purpose programming tasks like creating files. Second, scripts are constrained by the [[same origin policy]]: scripts from one Web site do not have access to information such as usernames, passwords, or cookies sent to another site. Most JavaScript-related security bugs are breaches of either the same origin policy or the sandbox. There are subsets of general JavaScript — ADsafe, Secure ECMA Script (SES) — that provide greater level of security, especially on code created by third parties (such as advertisements).<ref>[[Template:Cite web]]</ref><ref>[[Template:Cite web]]</ref> [[Caja project|Caja]] is another project for safe embedding and isolation of third-party JavaScript and HTML. [[Content Security Policy]] is the main intended method of ensuring that only trusted code is executed on a Web page. [[Template:See also]] === Cross-site vulnerabilities === [[Template:Main]] A common JavaScript-related security problem is [[cross-site scripting]], or XSS, a violation of the [[same origin policy|same-origin policy]]. XSS vulnerabilities occur when an attacker is able to cause a target Web site, such as an online banking website, to include a malicious script in the webpage presented to a victim. The script in this example can then access the banking application with the privileges of the victim, potentially disclosing secret information or transferring money without the victim's authorization. A solution to XSS vulnerabilities is to use ''HTML escaping'' whenever displaying untrusted data. Some browsers include partial protection against ''reflected'' XSS attacks, in which the attacker provides a URL including malicious script. However, even users of those browsers are vulnerable to other XSS attacks, such as those where the malicious code is stored in a database. Only correct design of Web applications on the server side can fully prevent XSS. XSS vulnerabilities can also occur because of implementation mistakes by browser authors.<ref>MozillaZine, [http://www.mozillazine.org/talkback.html?article=4392 Mozilla Cross-Site Scripting Vulnerability Reported and Fixed]</ref> Another cross-site vulnerability is [[cross-site request forgery]] or CSRF. In CSRF, code on an attacker's site tricks the victim's browser into taking actions the user didn't intend at a target site (like transferring money at a bank). It works because, if the target site relies only on cookies to authenticate requests, then requests initiated by code on the attacker's site will carry the same legitimate login credentials as requests initiated by the user. In general, the solution to CSRF is to require an authentication value in a hidden form field, and not only in the cookies, to authenticate any request that might have lasting effects. Checking the HTTP Referrer header can also help. "JavaScript hijacking" is a type of CSRF attack in which a <tt><nowiki><script></nowiki></tt> tag on an attacker's site exploits a page on the victim's site that returns private information such as JSON or JavaScript. Possible solutions include: \* requiring an authentication token in the [[POST (HTTP)|POST]] and [[GET (HTTP)|GET]] parameters for any response that returns private information ==== Misplaced trust in the client ==== Developers of client-server applications must recognize that untrusted clients may be under the control of attackers. The application author cannot assume that his JavaScript code will run as intended (or at all) because any secret embedded in the code could be extracted by a determined adversary. Some implications are: \* Web site authors cannot perfectly conceal how their JavaScript operates because the raw source code must be sent to the client. The code can be [[obfuscated code|obfuscated]], but obfuscation can be reverse-engineered. \* JavaScript form validation only provides convenience for users, not security. If a site verifies that the user agreed to its terms of service, or filters invalid characters out of fields that should only contain numbers, it must do so on the server, not only the client. \* Scripts can be selectively disabled, so JavaScript can't be relied on to prevent operations such as right-clicking on an image to save it.<ref>[[Template:Cite journal]]</ref> \* It is extremely bad practice to embed sensitive information such as passwords in JavaScript because it can be extracted by an attacker. ==== Browser and plugin coding errors ==== JavaScript provides an interface to a wide range of browser capabilities, some of which may have flaws such as [[buffer overflow]]s. These flaws can allow attackers to write scripts that would run any code they wish on the user's system. This code is not by any means limited to another JavaScript application. For example, a buffer overrun exploit can allow an attacker to gain access to the operating system's API with superuser privileges. These flaws have affected major browsers including Firefox,<ref>Mozilla Corporation, [http://www.mozilla.org/security/announce/2006/mfsa2006-38.html Buffer overflow in crypto.signText()]</ref> Internet Explorer,<ref>[[Template:Cite web]]</ref> and Safari.<ref>SecurityTracker.com, [http://securitytracker.com/alerts/2006/Mar/1015713.html Apple Safari JavaScript Buffer Overflow Lets Remote Users Execute Arbitrary Code and HTTP Redirect Bug Lets Remote Users Access Files]</ref> Plugins, such as video players, [[Adobe Flash#Flash client security|Adobe Flash]], and the wide range of [[ActiveX]] controls enabled by default in Microsoft Internet Explorer, may also have flaws exploitable via JavaScript (such flaws have been exploited in the past).<ref>SecurityFocus, [http://www.securityfocus.com/bid/19030/info Microsoft WebViewFolderIcon ActiveX Control Buffer Overflow Vulnerability]</ref><ref>Fusion Authority, [http://www.fusionauthority.com/security/3234-macromedia-flash-activex-buffer-overflow.htm Macromedia Flash ActiveX Buffer Overflow]</ref> In Windows Vista, Microsoft has attempted to contain the risks of bugs such as buffer overflows by running the Internet Explorer process with limited privileges.<ref>Mike Friedman, [http://blogs.msdn.com/ie/archive/2006/02/09/528963.aspx Protected Mode in Vista IE7]</ref> [[Google Chrome]] similarly confines its page renderers to their own "sandbox". ==== Sandbox implementation errors ==== Web browsers are capable of running JavaScript outside the sandbox, with the privileges necessary to, for example, create or delete files. Of course, such privileges aren't meant to be granted to code from the Web. Incorrectly granting privileges to JavaScript from the Web has played a role in vulnerabilities in both Internet Explorer<ref>US CERT, [https://www.kb.cert.org/vuls/id/713878 Vulnerability Note VU#713878: Microsoft Internet Explorer does not properly validate source of redirected frame]</ref> and Firefox.<ref>Mozilla Foundation, [http://www.mozilla.org/security/announce/2005/mfsa2005-41.html Mozilla Foundation Security Advisory 2005–41: Privilege escalation via DOM property overrides]</ref> In Windows XP Service Pack 2, Microsoft demoted JScript's privileges in Internet Explorer.<ref>Microsoft Corporation, [http://technet.microsoft.com/en-us/library/bb457150.aspx#EHAA Changes to Functionality in Microsoft Windows XP Service Pack 2: Part 5: Enhanced Browsing Security]</ref> [[Microsoft Windows]] allows JavaScript source files on a computer's hard drive to be launched as general-purpose, non-sandboxed programs (see: [[Windows Script Host]]). This makes JavaScript (like [[VBScript]]) a theoretically viable vector for a [[Trojan horse (computing)|Trojan horse]], although JavaScript Trojan horses are uncommon in practice.<ref>For one example of a rare JavaScript Trojan Horse, see Symantec Corporation, [http://www.symantec.com/security\_response/writeup.jsp?docid=2003-100111-0931-99 JS.Seeker.K]</ref> == Uses outside Web pages == In addition to Web browsers and servers, JavaScript interpreters are embedded in a number of tools. Each of these applications provides its own [[object model]] that provides access to the host environment. The core JavaScript language remains mostly the same in each application. === Embedded scripting language === \* Google's [[Google Chrome|Chrome]] extensions, [[Opera (Web browser)|Opera]]'s extensions, Apple's [[Safari (Web browser)|Safari 5]] extensions, Apple's [[Dashboard (Mac OS)|Dashboard Widgets]], Microsoft's [[Microsoft Gadgets|Gadgets]], [[Yahoo! Widgets]], [[Google Desktop#Gadgets and plug-ins|Google Desktop Gadgets]], and [[Serence]] [[Klipfolio]] are implemented using JavaScript. \* The [[MongoDB]] database accepts queries written in JavaScript. [[MongoDB]] and [[NodeJS]] are the core components of [[MEAN (software bundle)|MEAN]]: a [[solution stack]] for creating Web applications using just JavaScript. \* The [[Clusterpoint]] database accept queries written in JS/SQL, which is a combination of [[SQL]] and JavaScript. [[Clusterpoint]] has built-in computing engine that allows execution of JavaScript code right inside the [[distributed database]]. \* Adobe's [[Adobe Acrobat#Security|Acrobat and Adobe Reader]] support JavaScript in [[Portable Document Format|PDF]] files.<ref>[[Template:Cite web]]</ref> \* Tools in the [[Adobe Creative Suite]], including [[Adobe Photoshop|Photoshop]], [[Adobe Illustrator|Illustrator]], [[Dreamweaver]], and [[InDesign]], allow scripting through JavaScript. \* [[OpenOffice.org]], an office application suite, as well as its popular fork [[LibreOffice]], allows JavaScript to be used as a scripting language. \* The interactive music signal processing software [[Max/MSP]] released by Cycling '74, offers a JavaScript model of its environment for use by developers. It allows much more precise control than the default GUI-centric programming model. \* Apple's Logic Pro X digital audio workstation (DAW) software can create custom MIDI effects plugins using JavaScript. [[Template:Citation needed]] \* ECMAScript was included in the [[VRML]]97 standard for scripting nodes of VRML scene description files.[[Template:Citation needed]] \* The [[Unity (game engine)|Unity]] game engine supports a modified version of JavaScript for scripting via Mono.<ref>[[Template:Cite web]]</ref> \* [[DX Studio]] (3D engine) uses the [[SpiderMonkey (JavaScript engine)|SpiderMonkey]] implementation of JavaScript for game and simulation logic.<ref>[[Template:Cite web]]</ref> \* [[Maxwell Render]] ([[Rendering (computer graphics)|rendering]] software) provides an ECMA standard based scripting engine for tasks automation.<ref>THINK! The Maxwell Render Resourcer Center, [http://think.maxwellrender.com/scripting\_references-269.html Scripting References]</ref> \* [[Google Apps Script]] in [[Google Spreadsheets]] and [[Google Sites]] allows users to create custom formulas, automate repetitive tasks and also interact with other Google products such as Gmail.<ref>[[Google Apps Script]], [https://www.google.com/script/start/ Google Apps Script]</ref> \* Many [[Internet Relay Chat clients|IRC clients]], like [[ChatZilla]] or [[XChat]], use JavaScript for their scripting abilities.<ref>[[Template:Cite web]]</ref><ref>[[Template:Cite web]]</ref> \* [[RPG Maker]] MV uses Javascript as its scripting language.<ref>[[Template:Cite web]]</ref> === Scripting engine === \* Microsoft's [[Active Scripting]] technology supports [[JScript]] as a scripting language.<ref name="VersionInformation">[[Template:Cite web]]</ref> \* The [[Java (programming language)|Java programming language]] introduced the <tt>javax.script</tt> package in version 6 that includes a JavaScript implementation based on [[Rhino (JavaScript engine)|Mozilla Rhino]]. Thus, Java applications can host scripts that access the application's variables and objects, much like Web browsers host scripts that access a webpage's [[Document Object Model]] (DOM).<ref>[[Template:Cite web]]</ref>[[Template:Sfn]] \* The [[Qt (toolkit)|Qt]] C++ toolkit includes a <tt>QtScript</tt> module to interpret JavaScript, analogous to Java's <tt>javax.script</tt> package.<ref>Nokia Corporation, [http://doc.qt.nokia.com/4.6/qtscript.html QtScript Module]</ref> \* [[OS X Yosemite]] introduced JavaScript for Automation (JXA), which is built upon [[JavaScriptCore]] and the [[Open Scripting Architecture]]. It features an [[Objective-C]] bridge that enables entire [[Cocoa (API)|Cocoa]] applications to be programmed in JavaScript. \* Late Night Software's [[JavaScript OSA]] (a.k.a. JavaScript for OSA, or JSOSA) is a freeware alternative to [[AppleScript]] for Mac OS X. It is based on the Mozilla 1.5 JavaScript implementation, with the addition of a <tt>MacOS</tt> object for interaction with the operating system and third-party applications.<ref>[[Open Scripting Architecture]]</ref> === Application platform === \* [[ActionScript]], the programming language used in [[Adobe Flash]], is another implementation of the ECMAScript standard. \* [[Adobe Integrated Runtime]] is a JavaScript runtime that allows developers to create desktop applications. \* [[CA, Inc.]]'s AutoShell cross-application scripting environment is built on the [[SpiderMonkey (JavaScript engine)|SpiderMonkey]] JavaScript engine. It contains [[preprocessor]]-like extensions for command definition, as well as custom classes for various system-related tasks like file I/O, operation system command invocation and redirection, and COM scripting. \* [[GNOME Shell]], the shell for the [[Template:Nobr]] desktop environment,<ref>[[Template:Cite web]]</ref> made JavaScript its default programming language in 2013.<ref>[[Template:Cite web]]</ref> \* The [[Mozilla]] platform, which underlies [[Firefox]], [[Mozilla Thunderbird|Thunderbird]], and some other Web browsers, uses JavaScript to implement the [[graphical user interface]] (GUI) of its various products. \* [[Qt Quick]]'s markup language (available since Qt 4.7) uses JavaScript for its application logic. Its declarative syntax is also similar to JavaScript. \* [[TypeScript]] is a programming language based on JavaScript that adds support for optional [[type annotation]]s and some other language extensions such as classes, interfaces and modules. A TS-script compiles into plain JavaScript and can be executed in any JS host supporting [[ECMAScript]] 3 or higher. The compiler is itself written in TypeScript. \* [[Ubuntu Touch]] provides a JavaScript API for its unified usability interface. \* [[webOS]] uses the [[WebKit]] implementation of JavaScript in its [[Software development kit|SDK]] to allow developers to create stand-alone applications solely in JavaScript. \* [[Windows Runtime#JavaScript|WinJS]] provides a special Windows Library for JavaScript functionality in [[Windows 8]] that enables the development of [[Metro (design language)|Modern style]] (formerly ''Metro style'') applications in [[HTML5]] and JavaScript. == Development tools == Within JavaScript, access to a [[debugger]] becomes invaluable when developing large, non-trivial programs. Because there can be implementation differences between the various browsers (particularly within the [[Document Object Model]]), it is useful to have access to a debugger for each of the browsers that a Web application targets.<ref>[[Template:Cite web]]</ref> Script debuggers are integrated within [[Internet Explorer]], [[Firefox]], [[Safari (Web browser)|Safari]], [[Google Chrome]], [[Opera (Web browser)|Opera]] and [[Node.js]]<ref>[[Template:Cite web]]</ref><ref>[[Template:Cite web]]</ref><ref>[[Template:Cite web]]</ref> In addition to the native [[Internet Explorer Developer Tools]], three debuggers are available for Internet Explorer: [[Microsoft Visual Studio]] is the richest of the three, closely followed by [[Microsoft Script Editor]] (a component of [[Microsoft Office]]),<ref>[http://msdn2.microsoft.com/en-us/library/aa202668(office.11).aspx JScript development in Microsoft Office 11] (MS InfoPath 2003)</ref> and finally the free [[Microsoft Script Debugger]] that is far more basic than the other two. The free [[Microsoft Visual Web Developer Express]] provides a limited version of the JavaScript debugging functionality in Microsoft Visual Studio. Internet Explorer has included developer tools since version 8 (reached by pressing the F12 key). In comparison to Internet Explorer, Firefox has a more comprehensive set of developer tools, which include a debugger as well. Old versions of Firefox without these tools used a [[Firefox addon]] called [[Firebug (Firefox extension)|Firebug]], or the older [[Venkman]] debugger. Also, [[WebKit]]'s Web Inspector includes a JavaScript debugger,<ref>[[Template:Cite web]]</ref> which is used in [[Safari (Web browser)|Safari]]. A modified version called Blink DevTools is used in [[Google Chrome]]. [[Node.js]] has node-inspector, an interactive debugger that integrates with the Blink DevTools, available in [[Google Chrome]]. Last but not least, [[Opera (Web browser)|Opera]] includes a set of tools called [[Opera Dragonfly|Dragonfly]].<ref>[[Template:Cite web]]</ref> [[Template:Anchor]]In addition to the native computer software, there are ''online JavaScript IDEs'', debugging aids are themselves written in JavaScript and built to run on the Web. An example is the program [[JSLint]], developed by [[Douglas Crockford]] who has written extensively on the language. JSLint scans JavaScript code for conformance to a set of standards and guidelines. Many libraries for JavaScript, such as [[three.js]], provide links to demonstration code that can be edited by users. They are also used as a pedagogical tool by institutions such as [[Khan Academy]]<ref>[[Template:Cite web]]</ref> to allow students to experience writing code in an environment where they can see the output of their programs, without needing any setup beyond a Web browser. == Version history == [[Template:See also]] JavaScript was initially developed in 1996 for use in the [[Netscape Navigator]] browser. In the same year Microsoft released an implementation for Internet Explorer. This implementation was called [[JScript]] due to trademark issues. In 1997 the first standardized version of the language was released under the name [[ECMAScript]]. The following table is based on information from multiple sources.<ref>[[Template:Cite web]]</ref><ref>[[Template:Cite web]]</ref><ref>[[Template:Cite web]]</ref> {| class="wikitable" style="text-align:center;" |- ! Version !! Release date !! Equivalent to !! Netscape<br />Navigator !! Mozilla<br />Firefox !! Internet<br />Explorer !! Opera !! Safari !! Google<br />Chrome |- | [[Template:Version]] || March 1996 || || 2.0 || || 3.0 || || || |- | [[Template:Version]] || August 1996 || || 3.0 || || || || || |- | [[Template:Version]] || June 1997 || || 4.0-4.05 || || || 3<ref>[[History of the Opera web browser#Version 3]]</ref> || || |- | [[Template:Version]] || October 1998 || ECMA-262 1st + 2nd edition || 4.06-4.7x || || 4.0 || 5<ref>[[Template:Cite web]]</ref> || || |- | [[Template:Version]] || || || Netscape<br />Server || || || 6 || || |- | [[Template:Version]] || November 2000 || ECMA-262 3rd edition || 6.0 || 1.0 || 5.5 (JScript 5.5),<br />6 (JScript 5.6),<br />7 (JScript 5.7),<br />8 (JScript 5.8) || 7.0 || 3.0-5 || 1.0-10.0.666 |- | [[Template:Version]] || November 2005 || 1.5 + array extras + array and string generics + [[E4X]] || || 1.5 || || || || |- | [[Template:Version]] || October 2006 || 1.6 + [https://developer.mozilla.org/en-US/docs/JavaScript/New\_in\_JavaScript/1.7?redirectlocale=en-US&redirectslug=New\_in\_JavaScript\_1.7#Generators Pythonic generators] + iterators + let || || 2.0 || || || ||28.0.1500.95 |- | [[Template:Version]] || June 2008 || 1.7 + [[generator (computer programming)|generator expressions]] + [[closure (computer programming)|expression closures]] || || 3.0 || || 11.50 || || |- | [[Template:Version]] || || 1.8 + [[JSON#Native encoding and decoding in browsers|native JSON]] support + minor updates || || 3.5 || || || || |- | [[Template:Version]] || June 22, 2009 || 1.8.1 + minor updates || || 3.6 || || || || |- |1.8.5 |July 27, 2010 |1.8.2 + new features for ECMA-262 Edition 5 compliance. | |4.0 | | | | |} == Related languages and features == [[JSON]], or JavaScript Object Notation, is a general-purpose data interchange format that is defined as a subset of JavaScript's object literal syntax. Like much of JavaScript (regexps and anonymous functions as 1st class elements, closures, flexible classes, 'use strict'), [[JSON]], except for replacing [[Perl]]'s key-value operator '=>' by an [[RFC 822]]<ref>[[Template:Cite web]]</ref> inspired ':', is syntactically pure Perl. [[jQuery]] is a popular JavaScript library designed to simplify [[Document Object Model|DOM]]-oriented client-side HTML scripting along with offering cross-browser compatibility because various browsers respond differently to certain vanilla JavaScript code. [[Underscore.js]] is a utility JavaScript library for data manipulation that is used in both client-side and server-side network applications. Mozilla browsers currently support [[LiveConnect]], a feature that allows JavaScript and Java to intercommunicate on the Web. However, Mozilla-specific support for LiveConnect is scheduled to be phased out in the future in favor of passing on the LiveConnect handling via [[NPAPI]] to the Java 1.6+ plug-in (not yet supported on the Mac [[Template:As of]]).<ref>[http://java.sun.com/javase/6/webnotes/6u10/plugin2/liveconnect/ Release Notes for the Next-Generation Java™ Plug-In Technology (introduced in Java SE 6 update 10)]. Java.sun.com. Retrieved on 2013-06-13.</ref> Most browser inspection tools, such as [[Firebug (software)|Firebug]] in Firefox, include JavaScript interpreters that can act on the visible page's DOM. [[asm.js]] is a subset of JavaScript that can be run in any JavaScript engine or run faster in an [[ahead-of-time]] (AOT) compiling engine.<ref>[[Template:Cite web]]</ref> [[JSFuck]] is an [[esoteric programming language]]. Programs are written using only six different characters, but are still valid JavaScript code. p5.js<ref>[[Template:Cite web]]</ref> is an object oriented JavaScript library designed for artists and designers. It is based on the ideas of the [[Processing (programming language)|Processing]] project but is for the web. === Use as an intermediate language === As JavaScript is the most widely supported client-side language that can run within a Web browser, it has become an [[intermediate language]] for other languages to target. This has included both newly created languages and ports of existing languages. Some of these include: \* Oberon Script, a full implementation of the [[Oberon (programming language)|Oberon Programming Language]] that compiles to high-level JavaScript.<ref>[[Template:Cite web]]</ref> \* [[Objective-J]], a superset of JavaScript that compiles to standard JavaScript. It adds traditional inheritance and [[Smalltalk]]/[[Objective-C]] style dynamic dispatch and optional pseudo-static typing to JavaScript. \* [[Processing.js]], a JavaScript port of Processing, a programming language designed to write visualizations, images, and interactive content. It allows Web browsers to display animations, visual applications, games and other graphical rich content without the need for a Java applet or Flash plugin. \* [[CoffeeScript]], an alternate syntax for JavaScript intended to be more concise and readable. It adds features like array comprehensions (also available in JavaScript since version 1.7)<ref>[[Template:Cite web]]</ref> and pattern matching. Like Objective-J, it compiles to JavaScript. Ruby and Python have been cited as influential on CoffeeScript syntax. \* [[Google Web Toolkit]] translates a subset of Java to JavaScript. \* [[Scala (programming language)|Scala]], an object-oriented and functional programming language, has a Scala-to-JavaScript compiler.<ref>[[Template:Cite web]]</ref> \* [[Pyjamas (software)|Pyjamas]], a port of [[Google Web Toolkit]] to [[Python (programming language)|Python]] (translates a subset of Python to JavaScript) \* [[Dart (programming language)|Dart]], an open-source programming language developed by Google, can be compiled to JavaScript. \* Whalesong,<ref>[[Template:Cite web]]</ref> a [[Racket (programming language)|Racket]]-to-JavaScript compiler. \* [[Emscripten]], a [[LLVM]]-backend for porting native libraries to JavaScript. \* [[Fantom (programming language)|Fantom]] a programming language that runs on JVM, .NET and JavaScript. \* [[TypeScript]], a free and open-source programming language developed by Microsoft. It is a superset of JavaScript, and essentially adds optional static typing and class-based object-oriented programming to the language. \* [[Haxe]], an open-source high-level multiplatform programming language and compiler that can produce applications and source code for many different platforms including JavaScript. \* ClojureScript,<ref>[[Template:Cite web]]</ref> a compiler for [[Clojure]] that targets JavaScript. It is designed to emit JavaScript code that is compatible with the advanced compilation mode of the Google Closure optimizing compiler. \* [[Kotlin (programming language)|Kotlin]], a [[Type system#Static type-checking|statically-typed]] language that also compiles to [[Java byte code]]. As JavaScript has unusual limitations – such as no separate integer type, using floating point – languages that compile to JavaScript commonly have slightly different behavior than in other environments. === JavaScript and Java === A common misconception is that JavaScript is similar or closely related to [[Java (programming language)|Java]]. It is true that both have a C-like syntax (the C language being their most immediate common ancestor language). They also are both typically [[Sandbox (computer security)|sandboxed]] (when used inside a browser), and JavaScript was designed with Java's syntax and standard library in mind. In particular, all Java keywords were reserved in original JavaScript, JavaScript's standard library follows Java's naming conventions, and JavaScript's Math and Date objects are based on classes from Java 1.0,<ref name="popularity">[[Template:Cite web]]</ref> but the similarities end there. The differences between the two languages are more prominent than their similarities. Java has [[static typing]], while JavaScript's typing is [[Dynamic typing|dynamic]]. Java is loaded from compiled bytecode, while JavaScript is loaded as human-readable source code. Java's objects are [[Class-based programming|class-based]], while JavaScript's are [[Prototype-based programming|prototype-based]]. Finally, Java did not support [[functional programming]] until Java 8, while JavaScript has done so from the beginning, being influenced by [[Scheme (programming language)|Scheme]]. == References == [[Template:Reflist]] == Further reading == [[Template:Refbegin]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] \* [[Template:Cite book]] [[Template:Refend]] == External links == [[Template:Portal]] [[Template:Sisterlinks]] [[Template:Spoken Wikipedia]] \* [[Douglas Crockford]]'s [https://www.youtube.com/playlist?list=PL62E185BB8577B63D video lectures on JavaScript] \* Douglas Crockford's [http://javascript.crockford.com/survey.html A Survey of the JavaScript Programming Language] \* [[Template:DMOZ]] \* [https://github.com/jashkenas/coffee-script/wiki/List-of-languages-that-compile-to-JS/ List of languages that compile to JS] [[Template:JavaScript]] [[Template:Programming languages]] [[Template:ECMAScript]] [[Template:Web browsers]] [[Template:NodeJs]] [[Template:Authority control]] [[Category:1995 introductions]] [[Category:American inventions]] [[Category:Articles with example JavaScript code]] [[Category:Cross-platform software]] [[Category:Dynamically typed programming languages]] [[Category:Functional languages]] [[Category:JavaScript| ]] [[Category:Object-based programming languages]] [[Category:Programming languages created in 1995]] [[Category:Programming languages with an ISO standard]] [[Category:Prototype-based programming languages]] [[Category:Scripting languages]] [[Category:Web programming]]