

The Wayback Machine - <https://web.archive.org/web/20040812085853/http://en.wikipedia.org:80/wiki/JavaScript>

JavaScript

From Wikipedia, the free encyclopedia.

JavaScript is an object-oriented scripting language commonly used in websites. It was originally developed by Brendan Eich of Netscape Communications under the name "Mocha" and then "LiveScript" but then renamed to "JavaScript" and given a syntax closer to that of Sun Microsystems' Java language. JavaScript was first standardized in 1997–1999 by ECMA under the name **ECMAScript**. The standard (as of December, 1999) is ECMA-262 Edition 3, and corresponds to JavaScript 1.5. This is also now an ISO standard.

Contents

- 1 Java, JavaScript, and JScript
- 2 Usage
- 3 Environment
- 4 Incompatibilities
- 5 Language elements
 - 5.1 Variables
 - 5.2 Objects
 - 5.3 Data structures
 - 5.4 Control structures
 - 5.4.1 If ... else
 - 5.4.2 While loop
 - 5.4.3 Do ... while
 - 5.4.4 For loop
 - 5.4.5 For ... in loop
 - 5.4.6 Switch statement
 - 5.5 Functions
 - 5.6 User interaction
 - 5.7 Events
 - 5.8 Error handling
- 6 Offspring
- 7 See also
- 8 References
- 9 External links

Java, JavaScript, and JScript

The change of name from LiveScript to JavaScript happened at roughly the time that Netscape was including support for Java technology in its Netscape Navigator browser. The choice of name proved to be a source of much confusion. There is no real relation between Java and JavaScript; their similarities are mostly in syntax (both derived from C); their semantics are quite different, notably their object models are unrelated and largely incompatible.

Due to the de facto success of JavaScript as a web page enhancement language, Microsoft had little choice but to develop a compatible language; this is known as JScript. The need for common specifications for that language was the basis of the ECMA work which led to ECMAScript (and in turn set the stage for the standardization of C# a few years later).

Usage

Netflix v. GoTV
IPR2023-00759
Netflix Ex. 1027

JavaScript is an object-based scripting language with a syntax loosely based on C. Like C, it has the concept of *reserved keywords*, which (being executed from source) means it is almost impossible to extend the language without breakage.

Also like C, the language has no input or output constructs of its own; where C relies on standard I/O libraries, a JavaScript interpreter relies on a **host program** into which it is trivially (or more complexly) embedded. There are many such host programs, of which Web technologies are the most well known examples. These are examined first.

JavaScript embedded in a Web browser connects through interfaces called Document Object Models (DOMs) to applications, especially to the server side (web servers) and the client side web browser of internet applications. Many web sites use client-side JavaScript technology to create powerful dynamic web applications. It may use Unicode and can evaluate regular expressions (introduced in version 1.2 in Netscape Navigator 4 and Internet Explorer 4). JavaScript expressions contained in a string can be evaluated using the `eval` function.

One major use of Web-based JavaScript is to write little functions that are embedded in HTML pages and interact with the DOM of the browser to perform certain tasks not possible in static HTML alone, such as opening a new window, checking input values, changing images as the mouse cursor moves over etc. Unfortunately, the DOMs of browsers are not standardized, different browsers expose different objects or methods to the script, and it is therefore often necessary to write different variants of a JavaScript function for the various browsers.

Outside of the Web, JavaScript interpreters are embedded in a number of tools. The ActionScript interpreter inside PDF document readers such as Adobe's Acrobat Reader is JavaScript. The Mozilla platform, which underlies several common Web browsers, uses JavaScript to implement the user interface and transaction logic of its various products. JavaScript interpreters are also embedded in proprietary applications that lack scriptable interfaces. Finally, Microsoft's WSH technology supports JavaScript (via JScript) as an operating system scripting language.

When used outside a Web browsers, the interaction between JavaScript and the host system can be expressed in a number of ways. DOM standards might still be implemented, but other access techniques like COM, XPCOM, Java interfaces and Application Object Models are also equally possible.

JavaScript/ECMAScript is implemented by:

- KJS
- Rhino
- SpiderMonkey
- Narcissus
- NJS
- NGS
- Resin
- FESI
- SEE
- DMDScript

Environment

The Internet media type for JavaScript source code is `application/x-javascript`, but the unregistered `text/javascript` is more commonly used.

To embed JavaScript code in an HTML document, it must be preceded with:

```
<script type="text/javascript">
```

and followed with:

```
</script>
```

Older browsers typically require JavaScript to begin with:

```
<script language="JavaScript" type="text/javascript">
<!--
```

and end with:

```
// -->
</script>
```

The `<!-- ... -->` comment markup is required in order to ensure that the code is not rendered as text by very old browsers which do not recognize the `<script>` tag in HTML documents, and the `LANGUAGE` attribute is a deprecated HTML attribute which may be required for old browsers. However, `<script>` tags in XHTML/XML documents will not work if commented out, as conformant XHTML/XML parsers ignore comments and also may encounter problems with `--`, `<` and `>` signs in scripts (for example, the integer decrement operator and the comparison operators). XHTML documents should therefore have scripts included as XML CDATA sections, by preceding them with

```
<script type="text/javascript">
//</pre>
</div>
<div data-bbox="42 392 226 408" data-label="Text">
<p>and following them with</p>
</div>
<div data-bbox="59 434 128 459" data-label="Text">
<pre>//]]&gt;
&lt;/script&gt;</pre>
</div>
<div data-bbox="42 481 923 515" data-label="Text">
<p>(<code>/*</code> At the start of a line marks a Javascript comment, which prevents the <code>&lt;![CDATA[</code> and <code>]]&gt;</code> from being parsed by the script.)</p>
</div>
<div data-bbox="42 528 950 595" data-label="Text">
<p>HTML elements[1] (<a href="http://www.w3.org/TR/html4/interact/scripts.html#h-18.2.3">http://www.w3.org/TR/html4/interact/scripts.html#h-18.2.3</a>) may contain intrinsic events to which you can associate a script handler. To write valid HTML 4.01, the web server should return a 'Content-Script-Type' with value 'text/javascript'. If the web server cannot be so configured, the website author can optionally insert the following declaration for the default scripting language in the header section of the document.</p>
</div>
<div data-bbox="65 620 552 634" data-label="Text">
<pre>&lt;meta http-equiv="Content-Script-Type" content="text/javascript" /&gt;</pre>
</div>
<div data-bbox="42 662 243 686" data-label="Section-Header">
<h2>Incompatibilities</h2>
</div>
<div data-bbox="42 702 962 786" data-label="Text">
<p>Javascript, like HTML, is often not compliant to standards, instead being built to work with specific web browsers. The current ECMAScript standard should be the base for all Javascript implementations in theory, but in practice the Netscape (and Mozilla) browsers use <i>JavaScript</i>, Microsoft Internet Explorer uses <i>JScript</i>, and other browsers such as Opera and Safari use other <i>ECMAScript</i> implementations, often with additional nonstandard properties to allow compatibility with JavaScript and JScript.</p>
</div>
<div data-bbox="42 799 949 867" data-label="Text">
<p>JavaScript and JScript contain several properties which are not part of the official ECMAScript standard, and may also miss several properties. As such, they are in points incompatible, which requires script authors to work around these bugs. JavaScript is more standards-compliant than Microsoft's JScript, which means that a script file written according to the ECMA standards will work for most browsers, except those based on Internet Explorer.</p>
</div>
<div data-bbox="42 879 923 930" data-label="Text">
<p>This also means every browser may treat the same script differently, and what works for one browser may fail in another browser, or even in a different version of the same browser. Like with HTML, it is thus advisable to write standards-compliant code.</p>
</div>
<div data-bbox="42 942 649 961" data-label="Text">
<p>MSIE's VBScript is <i>not</i> JavaScript, and it is incompatible with the ECMA standard.</p>
</div>
<div data-bbox="38 964 526 980" data-label="Page-Footer">
<p><a href="https://web.archive.org/web/20040812085853/http://en.wikipedia.org/wiki/JavaScript">https://web.archive.org/web/20040812085853/http://en.wikipedia.org/wiki/JavaScript</a></p>
</div>
<div data-bbox="928 964 966 979" data-label="Page-Footer">
<p>3/10</p>
</div>
```

Language elements

Variables

Variables are generally dynamically typed. Variables are defined by either just assigning them a value or by using the `var` statement. Variables declared outside of any function are in "global" scope, visible in the entire web page; variables declared inside a function are local to that function. To pass variables from one page to another, a developer can set a cookie or use a hidden frame or window in the background to store them.

Objects

Everything in JavaScript is either a primitive value or an object. Objects are entities that have an identity (they are only equal to themselves) and that map property names to values. That is, an object is an associative array similar to hashes in the Perl programming language, or dictionaries in Python, PostScript and Smalltalk.

JavaScript has several kinds of built in objects, namely object, Array, string, Number, Boolean, Function, Date and Math. Other objects are "host objects", defined not by the language but by the runtime environment. In a browser, typical host objects belong to the DOM (window, form, links etc.).

By defining a constructor function it is possible to define objects. JavaScript is a prototype based object-oriented language. That mean that inheritance is between objects, not between classes (Javascript has no classes). Objects inherit properties from their prototypes.

One can add additional properties or methods to individual objects after they have been created. To do this for all instances created by a single constructor function, one can use the **prototype** property of the constructor to access the prototype object.

Example: Creating an object

```
// constructor function
function MyObject(attributeA, attributeB) {
  this.attributeA = attributeA
  this.attributeB = attributeB
}
```

```
// create an Object
obj = new MyObject('red', 1000)
```

```
// access an attribute of obj
alert(obj.attributeA)
```

```
// access an attribute with the associative array notation
alert(obj["attributeA"])
```

Object hierarchy can be emulated in JavaScript. For example:

```
function Base()
{
  this.Override = _Override;
  this.BaseFunction = _BaseFunction;
  function _Override()
  {
    alert("Base::Override()");
  }
  function _BaseFunction()
  {
    alert("Base::BaseFunction()");
  }
}
```

```
}
}
```

```
function Derive()
{
  this.Override = _Override;
  function _Override()
  {
    alert("Derive::Override()");
  }
}
Derive.prototype = new Base();
```

```
d = new Derive();
d.Override();
d.BaseFunction();
```

will result in the display:

```
Derive::Override()
Base::BaseFunction()
```

Data structures

A typical data structure is the Array, which is a map from integers to values. In Javascript, all objects can map from integers to values, but Arrays are a special type of objects that has extra behavior and methods specializing in integer indices (e.g., `join`, `slice`, and `push`).

Arrays have a `length` property that is guaranteed to always be larger than the largest integer index used in the array. It is automatically updated if one creates a property with an even larger index. Writing a smaller number to the `length` property will remove larger indices. This `length` property is the only special feature of Arrays, that distinguishes it from other objects.

Elements of Arrays may be accessed using normal object property access notation:

```
myArray[1]
myArray["1"]
```

These two are equivalent. Its not possible to use the "dot"-notation or strings with alternative representations of the number:

```
myArray.1 (syntax error)
myArray["01"] (not the same as myArray[1])
```

Declaration of an array can use either an Array literal or the `Array` constructor:

```
myArray = [0,1,,4,5]; (array with length 6 and 4 elements)
myArray = new Array(0,1,2,3,4,5); (array with length 6 and 6 elements)
myArray = new Array(365); (an empty array with length 365)
```

Arrays are implemented so that only the elements defined use memory; they are "sparse arrays". Setting `myArray[10] = 'something'` and `myArray[57] = 'somethingOther'` only uses space for these two elements, just like any other object. The length of the array will still be reported as 58.

Control structures

If ... else

```
if (condition) {  
    statements  
}  
[else {  
    statemesnts  
}]
```

While loop

```
while (condition) {  
    statements  
}
```

Do ... while

```
do {  
    statements  
} while (condition);
```

For loop

```
for ([initial-expression]; [condition]; [increment-expression]) {  
    statements  
}
```

For ... in loop

This loop goes through all properties of an object (or elements of an array).

```
for (variable in object) {  
    statement  
}
```

Switch statement

```
switch (expression) {  
    case label1 :  
        statements;  
        break;  
    case label2 :  
        statements;  
        break;  
    default :  
        statements;  
}
```

Functions

A function is a block with a (possibly empty) argument list that is normally given a name. A function may give back a return value.

```
function(arg1, arg2, arg3) {  
    statements;  
}
```

```
    return expression;
}
```

Example: Euclid's original algorithm of finding the greatest common divisor. (This is a geometrical solution which subtracts the longer segment from the shorter):

```
function gcd(segmentA, segmentB)
{
    while(segmentA!=segmentB)
        if(segmentA>segmentB)
            segmentA -= segmentB;
        else
            segmentB -= segmentA;
    return(segmentA);
}
```

The number of arguments given when calling a function must not necessarily accord to the number of arguments in the function definition. Within the function the arguments may as well be accessed through the **arguments** array.

Every function is an instance of **Function**, a type of base object. Functions can be created and assigned like any other objects:

```
var myFunc1 = new Function("alert('Hello')");
var myFunc2 = myFunc1;
myFunc2();
```

results in the display:

Hello

User interaction

Most interaction with the user is done by using HTML forms which can be accessed through the HTML DOM. However there are as well some very simple means of communicating with the user:

- Alert dialog box
- Confirm dialog box
- Prompt dialog box
- Status bar
- Console

Events

Text elements may be the source of various events which can cause an action if a ECMAScript event handler is registered. In HTML these event handler functions are often defined as anonymous functions directly within the HTML tag.

List of events

- onAbort
- onBlur¹
- onChange
- onClick
- onDoubleClick
- onDragDrop
- onError
- onFocus²
- onKeyDown

- `onKeyPress`
- `onKeyUp`
- `onLoad`
- `onMouseDown`
- `onMouseMove`
- `onMouseOut`
- `onMouseOver`
- `onMouseUp`
- `onMove`
- `onReset`
- `onResize`
- `onSelect`
- `onSubmit`
- `onUnload`

¹ Blur is when the object is deselected or something else is selected instead.

² Focus is when the object is selected, usually in a form.

See also <http://tech.irt.org/articles/js058/>

Error handling

Newer versions of JavaScript (as used in Internet Explorer 5 and Netscape 6) include a `try ... catch` error handling statement.

The `try ... catch ... finally` statement catches exceptions resulting from an error or a `throw` statement. Its syntax is as follows:

```
try {  
    // Statements in which exceptions might be thrown  
} catch(error) {  
    // Statements that execute in the event of an exception  
} finally {  
    // Statements that execute afterward either way  
}
```

Initially, the statements within the `try` block execute. If an exception is thrown, the script's control flow immediately transfers to the statements in the `catch` block, with the exception available as the error argument. Otherwise the `catch` block is skipped. Once the `catch` block finishes, or the `try` block finishes with no exceptions thrown, the statements in the `finally` block execute. This figure summarizes the operation of a `try...catch...finally` statement:

Here's a script that shows `try ... catch ... finally` in action step by step.

```
try {  
    statements  
}
```

```
catch (err) {  
    // handle error  
}  
finally {  
    statements  
}
```

The `finally` part may be omitted:

```
try {
    statements
}
catch (err) {
    // handle error
}
```

Offspring

A novel example of the use of JavaScript are Bookmarklets, small sections of code within browser Bookmarks or Favorites.

The programming language used in Macromedia Flash (called ActionScript) bears a strong resemblance to JavaScript due to their shared relationship with ECMAScript. ActionScript has nearly the same syntax as JavaScript, but the object model is dramatically different.

See also

- CorbaScript
- LiveConnect
- DHTML

References

- Nigel McFarlane: *Rapid Appliation Development with Mozilla*, Prentice Hall Professional Technical References, ISBN 0131423436
- David Flanagan, Paula Ferguson: *JavaScript: The Definitive Guide*, O'Reilly & Associates, ISBN 0596000480
- Danny Goodman, Scott Markel: *JavaScript and DHTML Cookbook*, O'Reilly & Associates, ISBN 0596004672
- Danny Goodman, Brendan Eich: *JavaScript Bible*, Wiley, John & Sons, ISBN 0764533428
- Andrew H. Watt, Jinjer L. Simon, Jonathan Watt: *Teach Yourself JavaScript in 21 Days*, Pearson Education, ISBN 0672322978
- Thomas A. Powell, Fritz Schneider: *JavaScript: The Complete Reference*, McGraw-Hill Companies, ISBN 0072191279
- Scott Duffy: *How to do Everything with JavaScript*, Osborne, ISBN 0072228873
- Andy Harris, Andrew Harris: *Javascript Programming*, Premier Press, ISBN 0761534105
- Joe Burns, Andree S. Growney, Andree Growney: *JavaScript Goodies*, Pearson Education, ISBN 0789726122
- Gary B. Shelly, Thomas J. Cashman, William J. Dorin, Jeffrey Quasney: *Javascript: Complete Concepts and Techniques*, Course Technology, ISBN 0789562332
- Nick Heinle, Richard Koman: *Designing with Javascript*, O'Reilly & Associates, ISBN 1565923006
- Sham Bhangal, Tomasz Jankowski: *Foundation Web Design: Essential HTML, Javascript, CSS, PhotoShop, Fireworks, and Flash*, APress L. P., ISBN 1590591526

External links

- ECMAScript Language Specification (<http://www.ecma-international.org/publications/standards/Ecma-262.htm>)
- Proposal for Javascript 2.0 (<http://mozilla.org/js/language/js20/>)
- Reference for JavaScript 1.5 (<http://devedge.netscape.com/library/manuals/2000/javascript/1.5/reference/>)
- Reference for JavaScript 1.4 (<http://devedge.netscape.com/library/manuals/2000/javascript/1.4/reference/>)
- Reference for JavaScript 1.3 (<http://devedge.netscape.com/library/manuals/2000/javascript/1.3/reference/>)
- Useful JavaScript Collection (<http://www.js-vault.us>)
- Free JavaScript tutorials, reference, code, and help (<http://javascript.internet.com/>)
- Reference for JavaScript 1.3 (<http://devedge.netscape.com/library/manuals/2000/javascript/1.3/reference/>)
- Microsoft JScript resources in the MSDN Library (<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/js56jsoriJScript.asp>)
- W3Schools.com JavaScript Tutorials (<http://www.w3schools.com/js/default.asp>)

- JavaScript Search Engine (<http://www.javascript-2.com/>)
- Javascript Code Generators (<http://www.scriptomizers.com/scripts/javascript>)
- Crockford on Private Member Variables (<http://www.crockford.com/javascript/private.html>)
- Tips & Tricks (<http://www.quirksmode.org/>) - "... studying and defeating browser incompatibilities. ..."
- JavaScript Test Page - see if your browser supports JavaScript (<http://www.dcn.davis.ca.us/GO/EXPLORIT/java/IntroJavaScript.html>)
- Newsgroup (<news:comp.lang.javascript>)
- Cooperation with ...
 - PHP (<http://pear.php.net/package-info.php?pacid=93>)
 - TCL (<http://tcllib.sourceforge.net/doc/javascript.html>)
 - Perl (<http://search.cpan.org/~jpierce/Data-JavaScript-1.08/JavaScript.pm>)

Major programming languages

Edit
(http://en.wikipedia.org/w/wiki.phtml?title=Template:List_of_programming_languages&action=edit)

Ada | ALGOL | APL | AWK | BASIC | C | C++ | C# | COBOL | ColdFusion | Common Lisp | Delphi | Eiffel | Fortran | Haskell | IDL | Java | **JavaScript** | Lisp | Objective C | OCaml | Pascal | Perl | PHP | Prolog | Python | Ruby | Scheme | Smalltalk | SQL | Visual Basic

All programming languages...

Retrieved from "<http://en.wikipedia.org/wiki/JavaScript>"

Categories: Programming languages

- This page was last modified 15:49, 11 Aug 2004.
- All text is available under the terms of the GNU Free Documentation License (see **Copyrights** for details).