JAVASCRIPT

LEARN JAVASCRIPT WITH EASE

FROM BEGINNER TO EXPERT
IN LESS THAN A WEEK

STEPHEN BLUMENTHAL

JavaScript

By

Stephen Blumenthal

Table Of Contents

Introduction to JavaScript

JavaScript is an interpreted programming language, built on the ECMAScript standard. The language definition is really broad since it can be defined as aprocedural language based on prototypes, imperative, weakly typed, and dynamic.

JavaScript is mainly used as a client side programming language implemented as part of a web browser to allow developers an improved way to implement user interface and dynamic features in web pages, although there are implementations of JavaScript on the server side (SSJS) the popularity of the language is due to the client side implementations alone. JavaScript can also be found outside web applications, for example as a way to add interactivity to PDF documents and desktop widgets.

JavaScript was designed with a similar syntax as C, although it takes names and conventions from the Java programming language. However, despite the name Java and JavaScript are not related and have different semantics and purposes.

JavaScript was originally developed by Brendan Eich of Netscape under the name Mocha, which was later renamed to LiveScript, to finally being called JavaScript. The name change coincided approximately with the moment in which Netscape added support for Java technology in its web browser Netscape Navigator version 2.0B3 in late 1995. The name JavaScript was confusion, giving the impression that the language is an extension of Java, and it has been characterized by many as a marketing strategy for Netscape to gain prestige and innovate in what were the new web programming languages.

The following year Microsoft implemented a similar client side programming languages as part of its Internet Explorer 3.0 web browser. Microsoft called its client side language "jscript", to avoid problems related to the brand. The Jscript term seems so similar that the both "javascript" and "jscript" are often used interchangeably, but the specification of JScript is not 100% compatible with the ECMA specifications.

To avoid these incompatibilities, the World Wide Web Consortium (W3C) designed the standard Document Object Model (DOM, or document object model), which was incorporated in the version 6 of Internet Explorer and Netscape Navigator, Opera version 7, Mozilla Firefox since its first release, and all modern browsers thereafter.

In 1997 there was a proposal to submit JavaScript to the standard of the European Computer Manufacturers ' Association ECMA, which despite its name is not European but international, based in Geneva. In June 1997, it was adopted as an ECMA standard under the name of ECMAScript. JavaScript also became an ISO standard.

Because of its standardization and the great adoption of the internet, JavaScript has become the most used programming language in the planet.

Note: JavaScript is a registered trademark of Oracle Corporation. It is used under license by the products created by Netscape Communications and current entities such as the Mozilla Foundation.

Uses of JavaScript

JavaScript is present in most web pages today. Chances are that the page you are looking at right now contains the code for JavaScript. Try this activity: Right-click on a web page, then click 'View Source'. You should be able to find the word JavaScript somewhere in the code of the page.

While HTML markup language allows web developers to format content, JavaScript allows them to make the page dynamic. For example, HTML allows for making text bold, creating text boxes, and creating buttons, whereas JavaScript allows for changing text on the page, creating pop-up messages, and validating text in text boxes to make sure re q uired fields have been filled. JavaScript makes web pages more dynamic by allowing users to interact with web pages, click on elements, and change the pages.

What JavaScript can do for you

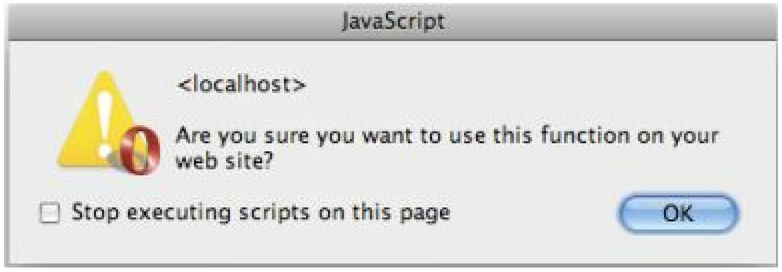
Let’s take a step back and count the merits of JavaScript:

- JavaScript is very easy to implement. All you need to do is put your code in the HTML document and tell the browser that it is JavaScript.
- JavaScript works on web users’ computers - even when they are offline!
- JavaScript allows you to create highly responsive interfaces that improve the user experience and provide dynamic functionality, without having to wait for the server to react and show another page.
- JavaScript can load content into the document if and when the user needs it, without reloading the entire page — this is commonly referred to as Ajax.
- JavaScript can test for what is possible in your browser and react accordingly — this is called Principles of unobtrusive JavaScript or sometimes defensive Scripting.
- JavaScript can help fix browser problems or patch holes in browser support — for example fixing CSS layout issues in certain browsers.

That is a lot for a language that until recently was laughed at by programmers favouring “higher programming languages”. One part of the renaissance of JavaScript is that we are building more and more complex web applications these days, and high interactivity either requires Flash (or other plugins) or scripting. JavaScript is arguably the best way to go, as it is a web standard, it is supported natively across browsers (more or less — some things differ across browsers, and these differences are discussed in appropriate places in the articles that follow this one), and it is compatible with other open web standards.

Common uses of JavaScript

The usage of JavaScript has changed over the years we've been using it. At first, JavaScript interaction with the site was mostly limited to interacting with forms, giving feedback to the user and detecting when they do certain things. We used alert() to notify the user of something (see Figure 1), confirm() to ask if something is OK to do or not and either prompt() or a form field to get user input.


JavaScript

<localhost>

Are you sure you want to use this function on your web site?

Figure 1: Telling the end user about errors using an alert() statement was all we could do in the early days of JavaScript. Neither pretty nor subtle.

This lead mostly to validation scripts that stopped the user to send a form to the server when there was a mistake, and simple converters and calculators. In addition, we also managed to build highly useless things like prompts asking the user for their name just to print it out immediately afterwards.

Another thing we used was document.write() to add content to the document. We also worked with popup windows and frames and lost many a nerve and pulled out hair trying to make them talk to each other. Thinking about most of these technologies should make any developer rock forward and backward and curl up into a fetal position stammering "make them go away", so let's not dwell on these things — there are better ways to use JavaScript!

Enter DOM scripting

When browsers started supporting and implementing the Document Object Model (DOM), which allows us to have much richer interaction with web pages, JavaScript started to get more interesting.

The DOM is an object representation of the document. The previous paragraph for example (check out its source using view source) in DOM speak is an element node with a nodeName of p. It contains three child nodes — a text node containing "When browsers started supporting and implementing the " as its nodeValue, an element node with a nodeName of a, and another text node with a nodeValue of ", which allows us to have much richer interaction with web pages, JavaScript started to get more interesting.". The a child node also has an attribute node called href with a value and a child node that is a text node with a nodeValue of "Document Object Model(DOM)".

You could also represent this paragraph visually using a tree diagram, as seen in Figure 2.

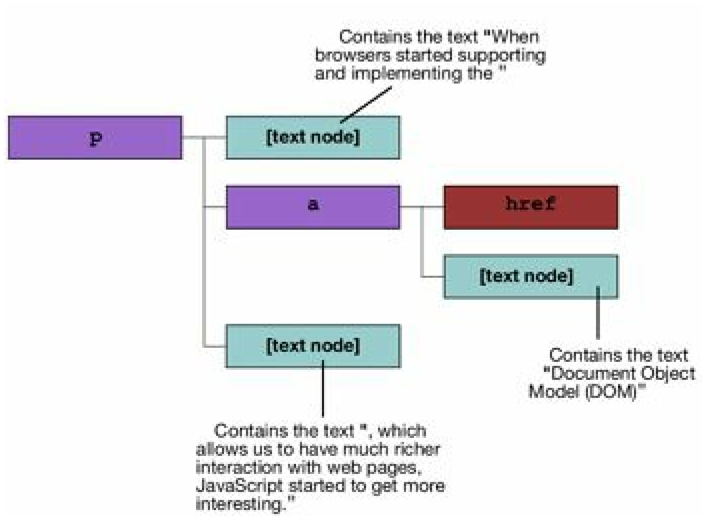
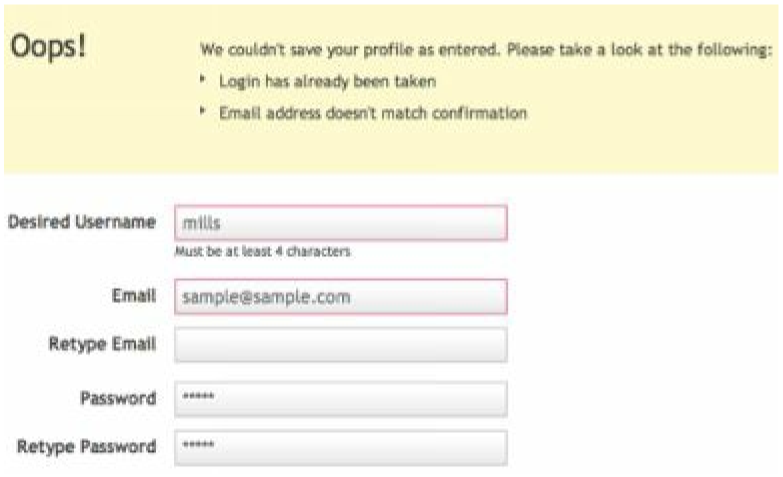

Contains the text "When browsers started supporting and implementing the "
p
[text node]
a
href
[text node]
[text node]
Contains the text "Document Object Model (DOM)"
Contains the text ", which allows us to have much richer interaction with web pages, JavaScript started to get more interesting."

Figure 2: A visual representation of our sample DOM tree.

In human words you can say that the DOM explains both the types, the values and the hierarchy of everything in the document — you don't need to know anything more for now.

- Access any element in the document and manipulate its look, content and attributes.
- Create new elements and content and apply them to the document when and if they are needed.

This means that we don't have to rely on windows, frames, forms and ugly alerts any longer, and can give feedback to the user in the document in a nicely styled manner, as indicated in Figure 3.



Oops!

We couldn't save your profile as entered. Please take a look at the following:

- Login has already been taken
- Email address doesn't match confirmation

Desired Username	mills Must be at least 4 characters
Email	sample@sample.com
Retype Email	
Password	*****
Retype Password	*****

Figure 3: Using the DOM you can create nicer and less intrusive error messages.

Together with event handling this is a very powerful arsenal to create interactive and beautiful interfaces.

Event handling means that our code reacts to things that happen in the browser. This could be things that happen automatically — like the page finishing loading — but most of the time we react to what the user did to the browser.

Users might resize the window, scroll the page, press certain keys or click on links, buttons and elements using the mouse. With event handling we can wait for these things to happen and tell the web page to respond to these actions as we wish. Whereas in the past, clicking any link would take the site visitor to another document, we can now hijack this functionality and do something else like showing and hiding a panel or taking the information in the link and using it to connect to a web service.

Other modern uses of JavaScript

And this is basically what we are doing these days with JavaScript. We enhance the old, tried and true web interface — clicking links, entering information and sending off forms, etc. — to be more responsive to the end user. For example:

A sign-up form can check if your user name is available when you enter it, preventing you from having to endure a frustrating reload of the page.

A search box can give you suggested results while you type, based on what you've entered so far (for example "bi" could bring up suggestions to choose from that contain this string, such as "bird", "big" and "bicycle"). This usage pattern is called autocomplete

Information that changes constantly can be loaded periodically without the need for user interaction, for example sports match results or stock market tickers.

Information that is a nice-to-have and runs the risk of being redundant to some users can be loaded when and if the user chooses to access it. For example the navigation menu of a site could be 6 links but display links to deeper pages on-demand when the user activates a menu item.

JavaScript can fix layout issues. Using JavaScript you can find the position and area of any element on the page, and the dimensions of the browser window. Using this information you can prevent overlapping elements and other such issues. Say for example you have a menu with several levels; by checking that there is space for the sub-menu to appear before showing it you can prevent scroll-bars or overlapping menu items.

JavaScript can enhance the interfaces HTML gives us. While it is nice to have a text input box you might want to have a combo box allowing you to choose from a list of preset values or enter your own. Using JavaScript you can enhance a normal input box to do that.

You can use JavaScript to animate elements on a page — for example to show and hide information, or highlight specific sections of a page — this can make for a more usable, richer user experience.

Using JavaScript sensibly and responsibly

There is not much you cannot do with JavaScript — especially when you mix it with other technologies like Canvas or SVG. However, with great power comes great responsibility, and you should always remember the following when using JavaScript.

- JavaScript might not be available — this is easy to test for so not really a problem. Things that depend on JavaScript should be created with this in mind however, and you should be careful that your site does not break (ie essential functionality is not available) if JavaScript is not available.
- If the use of JavaScript does not aid the user in reaching a goal more q uickly and efficiently you are probably using it wrong.
- Using JavaScript we often break conventions that people have got used to over the years of using the web (for example clicking links to go to other pages, or a little basket icon meaning “shopping cart”). Whilst these usage patterns might be outdated and inefficient, changing them still means making users change their ways — and this makes humans feel uneasy. We like being in control and once we understand something, it is hard for us to deal with change. Your JavaScript solutions should feel naturally better than the previous interaction, but not so different that the user cannot relate to it via their previous experience. If you manage to get a site visitor saying “ah ha — this means I don’t have to wait” or “Cool — now I don’t have to take this extra annoying step”— you have got yourself a great use for JavaScript.
- JavaScript should never be a security measure. If you need to prevent users from accessing data or you are likely to handle sensitive data then don’t rely on JavaScript. Any JavaScript protection can easily be reverse engineered and overcome, as all

the code is available to read on the client machine. Also, users can just turn JavaScript off in their browsers.

Get started with JavaScript

Getting Started: Setting Up your code.

Where do your JavaScript codes go? Well, basically anywhere inside the <html> tags of your page. The beginning of your code begins with <script type="text/javascript"> and ends with </script>

```
<html>

<head><title>This is an example page</title></head>

<body>

Welcome to the JavaScript course!

<script type="text/javascript">

<!--

document.write("Hi there. This text is written using JavaScript!")

//-->

</script>

</body>

</html>
```

Output: Hi there. This text is written using JavaScript!

As you can see, we began our script with the tag <script language="type/javascript"> The part in red is purely optional, as the browser by default assumes a <script> tag to be JavaScript, though you should include it nevertheless for validation reasons. The second and next to last lines of the above example are <!-- and //-->, which are HTML comment tags tailored for JavaScript. It is recommended you include them to hide your code against very old browsers that don't support JavaScript. If you don't include them and