JavaScript

(last updated April 15, 2013: LSS)

JavaScript is a scripting language, specifically for use on web pages. It runs within the browser (that is to say, it is a *client-side scripting language*), and it can be used to make web pages dynamic, that is, able to respond to events (such as mouse movements) and able to allow users to fill in forms which can then be sent to the server. In fact, JavaScript can do a great deal more than this.

JavaScript is described in detail in many books on the subject, and there is excellent tutorial material at http://www.w3schools.com/js/default.asp.

Basics

JavaScript is embedded in the HTML sent from the server to the browser. It is placed inside a `<script> </script>` tag:

```
<script>
document.write("Hello World!");
</script>
```

(In earlier versions of HTML, the attribute `type="text/javascript"` was required inside the `<script>` tag, but this is no longer the case for HTML5.)[1]

The script may be placed either in the header or the body of the HTML: if placed in the body, the JavaScript will be executed when the browser reaches that part of the HTML file. Those parts placed in the header will not be executed until (or unless) called (see later). It is also possible to place (some or all of the) the JavaScript in an external file using the attribute `src`, enabling sharing of JavaScript across web pages. For example,

```
<script src="scripts/usethisscript.js">
</script>
```

would use the JavaScript file `usethisscript.js` located in the local folder `scripts`. Note that the JavaScript file may be on a completely different machine: the source (`src`) may be a full URL.

[1] Also, in the past the JavaScript was often placed inside HTML comment brackets `<!-- ... -->`, so that if the browser did not support JavaScript, the JavaScript would have no effect. But these days, virtually all browsers do support JavaScript.

JavaScript is object oriented, which is to say, it sees the page it is inside, as an object with many sub-objects. The name for this structure is the Document Object Model (DOM), and for each web page this takes the form of a tree, with the `document` at the root.

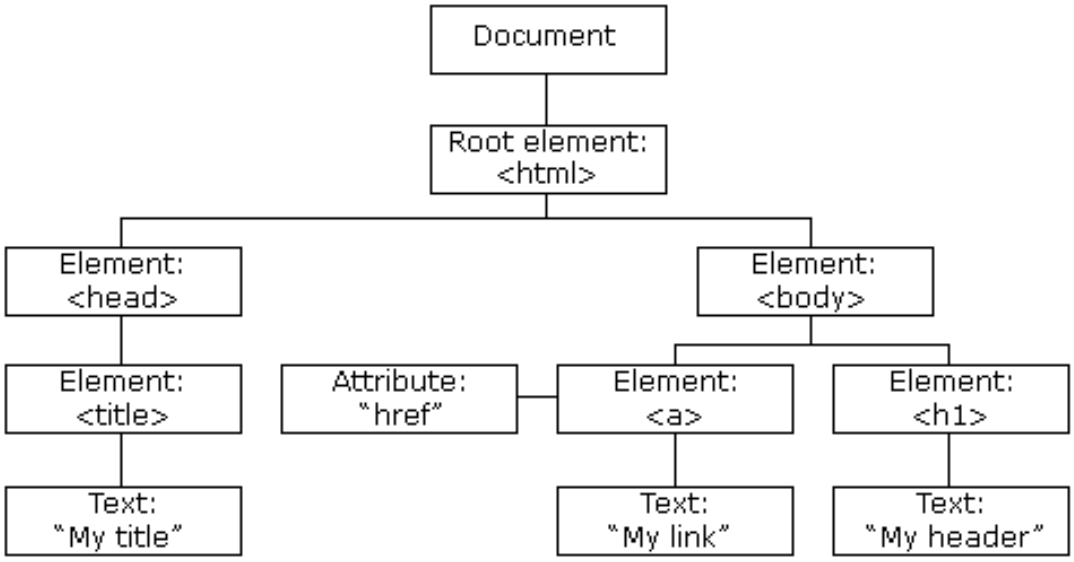

Figure 1: The Domain Object Model for a simple web page consisting of a head, with a title, and a body with a link and a header.

In addition, JavaScript has access to the identity of the browser itself (and, indeed, to the browser history), as well as to the display, and to the date and time. JavaScript can create new browser windows, as well as manipulating the contents of the window it is in. JavaScript sees all of these entities (browser, window, date etc.) as objects, and can access their constituent parts as object properties and/or use object methods to manipulate them. How to use *all* of these facilities is well beyond the scope of this document, however: there are quite a number of books available.

The navigator object

This is one of the simplest objects. It refers to the browser in use, and has just six properties, namely

- `navigator.appCodeName` which returns the code name of the browser,
- `navigator.appName` which returns the name of the browser,
- `navigator.appVersion` which returns the version information of the browser,
- `navigator.cookieEnabled` which determines whether cookies are enabled in the browser,
- `navigator.platform` which returns for which platform the browser is compiled, and
- `navigator.userAgent` which returns the user-agent header sent by the browser to the server,

and one method,

- `navigator.javaEnabled()` which returns whether or not the browser has Java enabled

What can one use this for? Platforms vary in size from iPhone sized, to Galaxy note sized, to iPad sized, to laptop to large screen desktops. Different browsers (Safari, Internet Explorer, Chrome, etc.) have slightly different characteristics, and with a single browser type, different versions have slightly different capabilities.

So, for example one might have code like

```
if (navigator.platform = "iPhone")
     // run one set of JavaScript statements
else if (navigator.platform = "MacIntel"
     // run a different set of JavaScript statements
```

In a similar way, one can test for the browser type and version, and run different JavaScript as required. It is, unfortunately, still the case that browsers are *not* all compatible, and also that many users are still using older versions of browsers (or browsers that *are pretending* to be older versions than they actually are!)

It would be quite inappropriate to provide the same level of detail on all the other objects – but there are books on JavaScript for this purpose.

Using JavaScript

Although JavaScript may be used in many different ways, the most common way of using it is to

- place JavaScript functions in the header of the HTML file
- call these functions (perhaps with parameters) from the body of the HTML file

The JavaScript functions may be called from anywhere in the HTML file (JavaScript scripts are executed as they are read by the Browser, so functions may be executed from inside a script placed at any point). However, one of the commonest JavaScript idioms is to call a function in response to an *event*. There are many different types of events, associated with different tags. For example

```
<body onload="startmessage()">
```

runs the `startmessage()` function when the page loads. Most HTML tags have associated event attributes which are used to call JavaScript functions.

Mouse events and JavaScript.

A very common use of JavaScript is to make the web page responsive to interaction using the mouse. Very nearly all tags have the following mouse event attributes:

Attribute	Value	Description
onclick	*script*	Script run on a mouse click
ondblclick	*script*	Script run on a mouse double-click
onmousedown	*script*	Script run when mouse button is pressed
onmousemove	*script*	Script run when mouse pointer moves
onmouseout	*script*	Script run when mouse pointer moves out of an element
onmouseover	*script*	Script run when mouse pointer moves over an element
onmouseup	*script*	Script run when mouse button is released

By using these, one can make nearly any HTML element be affected by mouse activity.

So for example, one can have

```
<img alt="Programme" name=prog src="programme_orig.png"
onMouseover="prog.src='programme_over.png'"
onMouseout="prog.src='programme_orig.png'">
```

which displays an image called `programme_orig.png` initially. When the mouse is over the image, the `onMouseover` event occurs, and the image is altered to `'programme_over.png'`. When the mouse moves away from the image, the original file `'programme_orig.png'` is used again. This type of code is frequently used to alter button images when the mouse is over them.

Note that in the above

1. the JavaScript is actually in the tag itself, rather than being in a function call.
2. We refer to the image as `prog.src`, rather than using its fully qualified name `document.images['prog'].src`

Declaring variables in JavaScript

JavaScript allows you to declare variables: these must be declared before they are used (i.e. they must have been encountered by the browser interpreting the HTML file prior to them being used: this usually means placing the variable declaration in the `<body>` of the HTML file, rather than in the `<head>`).

Variables are not typed (unlike Alice or Java or any of the C-based languages).

Thus

```
var xyz ;
```

declares a variable xyz that may later be used. It may be assigned a numeric value (e.g. `xyz = 5 ;`) or a string (e.g. `xyz = "Quasimodo" ;`). Often variables are initialised at their declaration:

```
var cartype = "Fiat" ;
```

The usual operators can be used with these variables: +, -, *, /, % (modulus division), ++ (increment), -- (decrement). Note that + can be used to join strings together.

The comparison operators include ==, !=, >, < , >=, <=, and also === which tests for equality of both value and type. Logical operators are && (logical and) , || (logical or), and ! (not).

Some common JavaScript usages.

Javascript is often used to alter the HTML inside a specific tag. Although it is possible to select the specific version of the tag inside a web page directly using the DOM, it is generally easier to (i) label the tag itself and then (ii) to use the specific identified tag. Labelling the tag is done using the `id=...` attribute (and this is available for virtually all HTML5 tags):

```
<p id="paragraph9">Some text</p>
```

then one can find the object with this identifier (inside a `<script>` tag):

```
var therightparagraph =
document.getElementById('paragraph9') ;
```

One can then alter any of the characteristics of the identified object: for example

```
therightparagraph.innerHTML = "Some different text" ;
```

Another example would be changing the name of a button: say we have a button (in HTML5)

```
<input type="button" id="button1" value="Click Me!">
```

which provides a button with "Click Me!" written on it, we could change its value (inside a `<script>` tag)

```
document.getElementById("button1").value = "Do not click
me again!" ;
```

The different objects in the document object model (everything from paragraphs, <p> to headers <h1> to <h6>, to forms, as well as introduced divisions) have a variety of *properties* and *methods*.

JavaScript statements.

JavaScript has if statements, switch statements, while statements, and for loops. See the www.w3schools.com website for more information on these. Indeed, JavaScript has many more capabilities which are beyond the scope of this introductory document.

Additional Information

For a detailed description of JavaScript language (i.e. for reference), try D. Flanagan, "JavaScript: The definitive Guide" 6th edition, O'Reilly 2011.

A facility for checking JavaScript syntax may be found at
http://www.javascriptlint.com/online_lint.php