JavaScript

JavaScript was originally developed in Netscape, by Brendan Eich. Battling with Microsoft over the Internet, Netscape considered their client-server solution as a distributed OS, running a portable version of Sun Microsystem's Java. Because Java was a competitor of C++ and aimed at professional programmers, Netscape also wanted a lightweight interpreted language that would complement Java by appealing to nonprofessional programmers, like Microsoft's VB.

Developed under the name Mocha, LiveScript was the official name for the language when it first shipped in beta releases of Netscape Navigator 2.0 in September 1995, but it was renamed JavaScript when it was deployed in the Netscape browser version 2.0B3.

JavaScript (sometimes abbreviated JS) is a prototype-based scripting language that is dynamic, weakly typed and has first-class functions. It is a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles.

JavaScript was formalized in the ECMAScript language standard and is primarily used in the form of client-side JavaScript, implemented as part of a Web browser in order to give enhanced user interfaces and dynamic websites. This enables programmatic access to computational objects within a host environment.

JavaScript's use in applications outside Web pages - for example in PDF documents, site-specific browsers, and desktop widgets - is also significant. Newer and faster JavaScript VMs and frameworks built upon them (notably Node.js) have also increased the popularity of JavaScript for server-side web applications.

JavaScript uses syntax influenced by that of C. JavaScript copies many names and naming conventions from Java, but the two languages are otherwise unrelated and have very different semantics. The key design principles within JavaScript are taken from the Self and Scheme programming languages. For more visit

JavaScript Basics Part 1

The code used to create the alert box that welcomed you to the tutorial as the page loaded is writen below, an alert is a function that opens up an alert box and the text that was displayed is written inside the brackets in between quotation marks you will also need to remember the semi-colon after the closing brackets in order to display the text in the alert box with your browser, also note that JavaScript is writen in between opening and closing script tags **<script></script>** these can be placed in between the head or the body tags of a document or both the head and body sections of the document, you can also link to an external script by including the src link and linking to the external file name which would need to be saved with the .js extension such as **<script src="externalscript.js"></script>.**

```
<html>
  <head>
   <title>JavaScript</title>
   <script>
   alert("Hello and Welcome to the JavaScript Tutorial Click OK to
   continue");
   </script>
  </head>
 <body>
 </body>
</html>
```

To write JavaScripts open your web browser you don't even need to be connected to the internet also you need to open a simple text editor such as notepad you wouldn't use Microsoft word because when you save the document it saves other metadata which can cause problems with the scripts, you will have to make sure your code is written the same throughout as JavaScript is case sensitive.

```
  <html>
    <head>
      <title>JavaScript</title>
      <script>
        function DisplayMsg (message)
        {

       alert("Are you ready: " + message);
        }

      </script>
    </head>
   <body>
    <form>
     <input type="button" value="Click here to see how the script
works"
     onclick="DisplayMsg('to begin your JavaScript Tutorial')">
    </form>
   </body>
  </html>
```

JavaScript Event Handlers

Event handlers are built-in mechanisms that basically scan for events or actions to happen. For example: when someone clicks the mouse on a piece of text: onClick. when someone moves the mouse over an image: onMouseOver. when a page is loaded into the web browser: onLoad

Examples of using various event handlers to fire off alerts()

```
    <p onclick="alert('This will open up an alert box when you click
here');
   ">Click Here!</p>
```

Note: when using the mouseover this will open the alert box wherever you move the mouse in line with the chosen text across the screen. to alter this in the code so that it just opens the alert box when hovering over just the text area in question add the following code:

```
    <p style="width:200px"
    onmouseover="alert('This will open up an alert box when you
move the
    mouse over it');" >Move mouse over</p>
```

You can also specify for an alert box to open up when the page is loaded into the browser by inserting onLoad within the body tag such as:
<body onLoad="alert('this will open this alert when the page is loading');">

JavaScript Basics Part 2

JavaScript code used to create the welcome dialogue

```
   <script>
   function welcome ()
   {

     var alerttext="Hello and welcome to JavaScript Part Two"
     var confirmtext="Do you wish to continue?"
     var username=prompt("Please enter your name","")
     alert(alerttext)
     confirm(confirmtext)
     alert("hello, "+username+" " + "lets begin Part Two")
   }

   </script>

   The code used in the body section

   <form class="center">
   <input type="button" value="Click here to see how the script
works"
   onclick="welcome ()">
   </form>
```

Statements

Statements can be run when the document is loaded, Immediately after the document loads, when responding to a users action or when called by another script statement, JavaScript statements invoke object methods such as write as in the document.write, initialise or define variables when naming the variable, manipulate variables, change the value of a variable, invoke a function routine that may need to be done over and over again, make programming decisions using conditionals.

Comments

```
            // this is a single line comment
/* this is a comment that can use multiple lines */
```

<!- - this is for the browser it comments out the javascript if the browser is not set up to read it, but a browser with javascript enabled will be able to read it //-- >

Programming concepts

Server side applications verses client side applications, server-side scripts rely on databases to generate the content, typically for frequently changing sites, PHP, ASP, .NET, JSP and ColdFusion are server-side applications, languages for server-side programs include Java, Python, Perl, C#, C++, Visual basic sometimes using JavaScript for server-side applications, a web author needs to have control or the server and backend databases, the server scripting is often done by accomplished programmers. With client side applications browsers can run separate plugins or applets for audio, video, animation (Adobe, flash, active X)

JavaScript code used to create a rollover image

```
      <script
      function MouseRollover(MyImage) {
        MyImage.src = "logo.gif";
      }

        function MouseOut(MyImage) {
         MyImage.src = "javascriptlogo.gif";
      }

      </script>

      The code used in the body section

      <div align="center">

      <img src="javascriptlogo.gif" alt="javascript logo"
          onMouseOver="MouseRollover(this)"
onMouseOut="MouseOut(this)" />
      </div>
```

JavaScript variables

Almost all statements manipulate data (values), JavaScript has several formal data value types, String a series of alphanumeric characters in quotes, Number any number that is not in quotes, Boolean a logical true or false, Object an entity that is defined by properties and methods, Function a function definition and Null a value that is totally devoid of content, the best way to work with data in script is to assign the data to a variable using the var keyword to declare and initialise and you should choose variable names carefully.

When naming variables you can't use spaces or dashes in between the names, although you could use the underscore to separate the words, camel case can be used such as firstName or lastName, also numbers can't be used at the start of the variable name although numbers can be used after the first letter, when declaring numbers in the variable you don't use them inside quotation marks.

JavaScript Basics Part 3

Operators

Operators are words or symbols in JavaScript expressions that perform on one or two values to get another value, operators perform actions called operands, expressions often contain operators and can work on variable values, Comparison operators are used to compare two values, assignment, Connubial, Boolean, Bitwise, Object and Misc.

Comparison Operators	Description
= =	Returns true if the values compared are equal
! =	Returns true if the values compared are not equal
>	Returns true if the first value compared is greater than the second one
> =	Returns true if the first value compared is greater than or equal to the second one
<	Returns true if the first value compared is less than the second one
< =	Returns true if the first value compared is less than or equal to the second one
Assignment Operators	
=	Assign the value of the right-hand operand to the variable on the left
+ =, - =, * =, / =)	Add/Minus/Multiply/Divide the value of the right-hand operand to the left-hand variable and stores the result in the left-hand variable
& =, | =	Assigns result of(left-hand operand && / || right-hand operand) to left-hand operand
Connubial Operators	
+	Adds two strings or numbers
-	Subtracts two numbers Negates a numerical value
*	Multiplies two numbers
/	Divides two numbers
Boolean Operators	
&&	AND operator returns true if the values compared are both true Example: x && y

Comparison Operators	Description
||	OR operator returns true if either of the values compared are true Example: x || y
!	NOT operator returns the opposite of a Boolean value Example: !x

Keywords and reserved words

Keywords are used by the JavaScript engine and cannot be used as objects, variable names, function names or methods; some keywords have been reserved for future use and are not presently interpreted.

abstract	do	implements	protected	typeof
as	double	import	public	use
boolean	else	in	return	var
break	enum	instanceof	short	void
byte	export	int	static	volatile
case	extends	interface	super	while
catch	false	is	switch	with
char	final	long	synchronized	
class	finally	namespace	this	
continue	float	native	throw	
const	for	new	throws	
debugger	function	null	translent	
default	goto	package	true	
delete	if	private	try	

JavaScript Basics Part 4

JavaScript code used to create a clock

```
window.onload = updateClock;

setInterval('updateClock()', 1000 );

function updateClock ( )
{

     var currentTime = new Date ( );

     var currentHours = currentTime.getHours ( );
     var currentMinutes = currentTime.getMinutes ( );
```

```
        var currentSeconds = currentTime.getSeconds ( );
        var currentDay = currentTime.getDate( );
        var currentMonth = (currentTime.getMonth ( ) + 1);
        var currentYear = currentTime.getFullYear ( );

        currentMinutes = ( currentMinutes < 10 ? "0" : "" ) +
currentMinutes;
        currentSeconds = ( currentSeconds < 10 ? "0" : "" ) +
currentSeconds;
        currentDay = ( currentDay < 10 ? "0" : "" ) + currentDay;
        currentMonth = ( currentMonth < 10 ? "0" : "" ) + currentMonth;

        var timeOfDay = ( currentHours < 12 ) ? "am" : "pm";

        currentHours = ( currentHours > 12 ) ? currentHours - 12 :
currentHours;

        currentHours = ( currentHours == 0 ) ? 12 : currentHours;

        var currentTimeString = currentHours + ":" + currentMinutes +

        ":" + currentSeconds + "   " + timeOfDay + ' - ' + currentDay +
'/' +

        currentMonth + '/' + currentYear;

        document.getElementById("SiteClock").innerHTML =
currentTimeString;

   }
```

Objects

The browser creates most objects, loading them into memory when the document loads in the browser, the JavaScript enhanced browser loads HTML elements and creates a hierarchy of objects whether the script calls for them or not, the property defines the particular settings of a object, the method is a command that control's the objects behaviour and the event handlers are how the object reacts to user events.

The window object is the master container, the root of the document object models hierarchy, it contains the documents content and chrome (toolbars, scroll bars, status bar, menu bars), the windows object can display modal dialog boxes (alerts), text in the status bar and create sub-windows, the windows object's events are handled by referencing with properties and methods.

Properties	Methods	Event Handlers
closed	Alert()	onLoad
defautStatus	Blur()	onUnload
frames	Close()	onFocus
name	Confirm()	onBlur
opener	Focus()	onError
parent	moveBy()	onResize
self	moveTo()	onClick
status	Open()	
top	Print()	
window	Prompt()	
	resizeBy()	
	resizeTo()	
	scrollBy()	
	scrollTo()	
	setInterval()	
	clearInterval()	
	setTimeout()	
	clearTimeout()	