Contents

Overview

- JavaScript is a full weight programming language that can be combined with HTML to expand its functionality.
- JavaScript is interpreted; it is not compiled.
- Client-side code is included in web pages, providing dynamic capabilities.
- Server-side JavaScript is embedded in Netscape's web servers.
- It is untyped, meaning variables do not have to have a data type specified.
- W3C document object model (DOM) elements can be accessed and manipulated with JavaScript.
- The language also has: variables and literals, conditions, flow control (aka looping), operators and expressions, functions.
- Specification: JavaScript was originally developed by Brendan Eich of Netscape under the name Mocha, later LiveScript, and finally renamed to JavaScript. Netscape delivered JavaScript to Ecma International[1] for standardization and the work on the specification, ECMA-262, began in November 1996. Four versions of ECMA-262 were published, most recently in December 2009. The version commonly used today is #3 from December 1999, it is available at http://www.ecma-international.org/publications/standards/ecma-262.htm . JavaScript is considered a dialect of ECMAScript. JavaScript includes extensions like the W3C-specified DOM (document object model), which is an API for HTML and XML documents; the DOM enables it to access web document elements.
- Web-browser implementation is not fully compliant with the specifications. No surprise here, as the lack of compliance also holds for HTML and CSS.

What can you do with JavaScript

- create content
- change content
- restyle content
- handle events
- change properties of the window
- interact with HTML forms

General

- case sensitive
- white space is ignored
- continue-break a code line with a backslash:

```
document.write("hello \
world")
```

NOT

```
document.write \
    ("hello")
```

- start comments on single line with "//": `sum=a + b  //explanation`
- surround multi-line comment:

```
/* line one
line two
line three */
```

[1] The name Ecma is, since 1994, no longer considered an acronym and no longer uses full capitalization. Ecma is an international, private (membership-based) non-profit standards organization for information and communication systems.

- to put 2 or more statements on same line, separate each with semicolon
- when nesting quoted strings, alternate double quotes with single quotes; for example, to indicate a quoted string inside a string literal, use single quotation marks:

```
document.write("<HR ALIGN='left' WIDTH=50>"
<INPUT TYPE="button" VALUE="Here" onclick="myfunc('astring')">
```

Variables

- Names must begin with letter or underscore: `hh,  _ww`
- Declare variables: `var hh="Susan", _idx=5`
- Variables declared within a function are local to the function.
- Variables declared outside a function are global to the page.
- Refer to an array entry: `ArrayName[index]`
- Variables are only accessible in the page in which they are created. A variable created in index.html cannot be accessed by about.html which was opened by index.html in a separate window.

Executing Javascript

How to run JavaScript

Code can be run as:

- an event handler
- automatically when the page loads (not the load event), typically used to generate content
- a hypertext link when it is activated

Declare JavaScript in META tag

```
<META http-equiv="Content-Script-Type" content="text/javascript">
```

JavaScript can be located in external file, access it with link

External JavaScript file has no HTML. The use of the LANGUAGE attribute is necessary in IE6.

```
<SCRIPT TYPE="text/javascript" SRC="x.js"></SCRIPT>
<SCRIPT TYPE="text/javascript" SRC="x.js" LANGUAGE="javascript"></SCRIPT>
```

Embed JavaScript within HTML

JavaScript code can be embedded within HTML in the HEAD section and/or the BODY section. Code in the HEAD section must be explicitly run; it is loaded before the BODY section. Code in the BODY section is run as the page is loaded.

Embed JavaScript in BODY if it generates content of page

Place JavaScript after the HTML elements that it references. For example, the code to hide a section of text must follow that text.

```
<SCRIPT TYPE="text/javascript">
<!--   begin to hide script from old browsers
document.open("log.html")
function showlog()
{window.open("log.html")}
function docstatus()
{window.status="message"}
// end hide -->
</SCRIPT>
```

Embed JavaScript statement or function in HTML statement as a Link object

The code is run when the link is activated.

```
<A HREF=”javascript: statement”> . . . </A>
<A HREF=”javascript: xyz()”>slower</A>
```

Invoking an event handler

An event handler must be “registered” in order to be run or invoked. There are three registration methods: by attribute, by property, and advanced. The first two methods are defined in the Level 0 API, the third is defined in the Level 2 DOM event model.

Event registration by HTML attribute. In this method the event handler is specified in HTML as an attribute, and applies only to the particular instance of the element. Examples:

```
<BODY onload="setfocus()"
<A HREF="somewhere.html" onClick="alert('I\'ve been clicked!')">
<BODY onload="window.open('log.html')">
<INPUT TYPE="button" VALUE="try" onClick="statement 1;statement 2">
```

If “x” is a function it must be defined in the HEAD section or in a linked script file. In this method, the event handler is run before the default action (the link). If the event handler returns a **false**, the default action is not taken. Not all default actions can be prevented, e.g., **unload**. This method is not recommended for XHTML because it requires you to write JavaScript behavior code in your XHTML structure layer, where it doesn't belong.

A limitation of this approach is that when the event handler is specified as a function, it can have no arguments.

If the function uses arguments, it must be invoked as a variable:

```
var varxyz = new Function("a", "b", "return a * b")
. . .
<input name="operand1" type="text" value="5"> First number
<input name="operand2" type="text" value="6"> Second number
<input name="button1" type="button" value="Multiply"
 onclick="document.form1.result.value=varxyz(document.form1.operand1.value,
          document.form1.operand2.value)"> Result here: 
<input name="result" type="text" value="">
```

Event registration by JavaScript property. In this method, the event handler is specified solely in JavaScript, not in HTML, and applies to all instances of the particular HTML element; it is specified as an element property. Examples:

```
<DIV ID="menu1"> . . .
document.getElementById("menu1").onmouseover = x

element.onclick = doSomething
```

applies to all instances of <element>.

The event handler can be removed:

```
element.onclick = null;

function doSomething() { this.style.backgroundColor = '#cc0000'; }
```

The element gets a red background whenever the user clicks on it.

In this method you can specify only one function as the event handler. Should you want to run more than one function, specify them as an anonymous function composed of several named functions:

```
element.onclick = function () {startDragDrop(); spyOnUser()}
```

As with the inline registration method, one limitation of this approach is that when the event handler is specified as a function, it can have no arguments.

Another limitation of this approach is that if the user interacts with a document element before the document is fully loaded (and before all its scripts have executed), the event handlers for the document element may not yet be defined.

Advanced event registration. There are two versions, W3C and Microsoft IE. While the W3C version is considered better, few browsers support it. I avoid it bcause of the cross-browser inconsistencies.

W3C's DOM Level 2 Event specification offers a simple way to register as many event handlers as you like for the same event on one element: with the method **addEventListener()**.

```
element.addEventListener('click', doSomething, false)
```

We can add as many event listeners as we want to the element, however the model does not state which event handler is fired first:

```
element.addEventListener('click', startDragDrop, false)
element.addEventListener('click', spyOnUser, false)
```

To remove an event handler, use the removeEventListener() method:

```
element.removeEventListener('click', spyOnUser, false)
```

One problem with the current implementation of W3C's event registration model is that you can't find out if any event handlers are already registered to an element. However, because **removeEventListener()** doesn't give any errors if the event listener you want to remove has not been added to the element, using it has no penalty.

Microsoft's IE model is similar. Instead of adding and removing event listeners, it attaches and detaches events:

```
element.attachEvent('onclick', doSomething)
element.detachEvent('onclick', doSomething)
```

The major drawback to the Microsoft model is that the **this** keyword always refers to the window and is completely useless.

Use JavaScript expression as HTML attribute value

* for Netscape 4 only

```
<SCRIPT>
barwidth=50
</SCRIPT>
<HR WIDTH="&{barwidth};%" ALIGN=”LEFT>
```

Manipulate elements of a document

You can refer to a document element in different ways:

a) by its ID attribute

```
EX:  <SPAN ID=a>
var xa = document.getElementById("a");
var yb = document.all["b"];                        AVOID
```

b) by its NAME attribute

```
EX:  <IMG NAME=b>
var xb = document.getElementsByName("b");    // returns an array
```

c) by its HTML tag name

```
EX:  <IMG . . . >
var xc = document.getElementsByTagName("img")[0];
var allnodes = document.getElementsByTagName("*");    // returns an array
```

In the last example, variable *allnodes* is a NodeList object which behaves like an array. It contains all the nodes in the document in the order in which they appear in the HTML text (not the rendered page). Use its length property to access the number of nodes in the list.

Once you have a reference to a document element, you can:

- change the text with document.createTextNode() and .replaceChild()
- insert a node with document.createElement() and .appendChild()
- delete a node
- re-parent a node with .replaceCild() and .appendChild()
- rearrange nodes
- change CSS attributes

Refer to the DOM for the various nodes.

You can read/write HTML attributes:

```
var eid = document.getElementById("x");
eid.setAttribute("class", "wow");
```

Text in an HTML document is represented in the DOM by a Text node (which is a special case of CharacterData. This node has several methods which can be employed to change the text with JavaScript code:
appendData(), deleteData(), InsertData(), replaceData(), and substringData(). The replaceData() method replaces one string with another.
The node has two properties:
data: the text contained by the node.
length: the number of characters in the text.

However, text in a SPAN tag, in IE6, is an Element node, not a Text node. There is no method to replace the text in an Element node.

The HTMLElement superclass has an innerHTML property that provides read/write access to the text contained in the element.

You can use JavaScript to create content with the document.write() method.

Apply style rules to elements

```
<p id=x> … </p>
. . .
var eid = document.getElementById("x");
eid.style.visibility = "hidden";
eid.style.display = "none"
```

Document Object Model (DOM)

"The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents." per the W3C. It is an API.

Version history:

Level 1 was published October 1998.
Most of Level 2 was published November 2000.
Core Level 3 was published April 2004.

DOM conformance is incomplete. IE 6 is non-compliant in the use of node-type constants defined for the Node object; it does use the integer literals. At this time you should avoid features that are known or likely to not be implemented by the major browsers: all of Level 3 and much of Level 2.

arrays are zero-based: [0] is the first item.

Nodes

The DOM is a group of Node objects arranged in a tree structure. A pseudo array of all the nodes in a document can be accessed as in the following example:

```
var allnodes = document.getElementsByTagName("*");
```

Variable allnodes is a NodeList object (technically an interface) which behaves like an array. It contains all the nodes in the document in the order in which they appear. Use its length property to access the number of nodes in the list. The NodeList always and automatically reflects the current nodes; there is no need to refresh it with code.

Objects

array
boolean
function
string
number
error — instances are thrown as exceptions when runtime errors occur
date
window — self, window, parent, top
navigator
frames[ ]
location—properties refer to various portions of current document's URL
history[ ] — You can access the number of entries in the history list via the history.length property
screen — info about display monitor
document
    styleSheets[ ]
    forms[ ]
        elements[ ] HTMLElement is a superclass
    anchors[ ]
    links[ ] array of hypertext links
    images[ ]
    applets[ ]
    embeds[ ]
    all[ ] an Internet Explorer 4+ array of all HTML elements AVOID
    elements[ ]
        CSS2Properties

Refer to objects	
object.property, also object[“property”] and object[varproperty]	the second form lets you access a property using a string variable
object.array[index], also object.array[“itemname”] and object.array[varname]	the second form lets you access an array item using a string variable
document.forms[0]	first form in document
document.f1	named form. NOTE: In IE 6 you can refer to a form with just the form name, but not in Netscape 7.
document.f1.elements[varname]	named form element; uses associative array feature
document.f1.elements[6]	seventh form element (array is zero-based)
document.f1.elements[i]	variable i + 1 form element
document.soq.zipcode	object that is form element named zipcode in form named soq
document.soq.zipcode.name	name of form element
document.soq.zipcode.value	value of named element in named form
document.soq.biztype	group of same-named elements, e.g., radio buttons
document.soq.biztype[i]	individual element in a group of same-named elements
document.getElementById(“TOC”)	element with named ID
document.getElementsByTag(“dd”)[0]	first element with named tag
document.getElementsByTagName(name)	returns a NodeList of elements with the given tag name in the order they appear in the tree; “name” is a string variable containing the tag name, the special string "*" represents all elements; CAUTION: this method of referencing a document node is unreliable, use ID instead
document.getElementsByTagName("TD").item(0)	first TD tag
document.getElementsByName(“special”)[i]	element with particular *name* attribute
document.elements[i]	i + 1 element in HTML
frames[1].frames[2]	third subframe in second main frame
parent.frame[1]	second frame in a window referred to by first frame
parent.mfg	when <FRAME name=mfg src=”mfg.html”> referred to by sibling frame
this	in a function, refers to the object that invoked the function: o.m() . . . function m() {this.length . . . } “this” refers to o. this.form refers to the form object of a form element. this.href refers to the document containing the link object
this.id	ID attribute of “this” object
element.style.fontFamily = “Georgia”	CSS property object

- Reference only works after objects have been loaded. So best to put scripts just before </BODY> tag.

Date methods

vardate = new Date() creates date object for today's date and time
vardate = new Date("December 25, 2003") creates date object dated 12-25-2003, time is zero
vardate = new Date("2003, 12, 25, 9, 30, 0) creates date object for 12-25-2003 09:30:00
setDay, setMonth, setFullYear, setTime
getDay, getMonth, getFullYear, getTime, getHours, getMinutes, getSeconds
to (returns string values from Date objects)

Document methods

close()	
getElementById()	returns element which can then be manipulated
getElementByTagName()	returns a NodeList array of all Element nodes with the specified tag in the order in which they appear in the source
open()	
write(text)	append text
writeln(text)	append text followed by newline character

Document properties

cookie	
domain	
lastModified	
location	deprecated synonym for URL, but needed sometimes
URL	excludes the ? part
referrer	names the document containing the link that brought the browser to the current document, if any
title	
offsetWidth	width of the named element. example: document.body.offsetWidth is the width of the browser window

Document events

onload
onunload
onbeforeload

Link object: a hypertext link

document.links[ ]

- properties refer to part of the URL. For example, the URL = "http://www.me.com:1234/here/there/us.html#we?x=y&a=b".
- properties are read/write.

hash = #we
host = www.me.com:1234
hostname = www.me.com
href = complete URL
pathname = /here/there/us.html
port = 1234

protocol = http:
search = ?x=y&a=b
target = name of a Window object
text = plain text, if any, between <a> and tags

Location properties

- properties refer to part of the URL. For example, the URL = "http://www.me.com:1234/here/there/us.html#we?x=y&a=b".

hash = #we
host = www.me.com:1234
hostname = www.me.com
href = complete URL, including the ? part
pathname = /here/there/us.html
port = 1234
protocol = http:
search = ?x=y&a=b

Navigator properties

appCodeName	code name of browser, e.g., "Mozilla"
appName	name of browser, e.g., "Netscape", "Microsoft Internet Explorer"
appVersion	version and platform of browser, contents vary widely. Must be parsed parseInt(appVersion) returns major version number parseFloat(appVersion) returns major and minor version in floating-point
cookieEnabled	true/false
platform	e.g., "Win32", "MacPPC", "Linux i586"

Screen properties

These are from JavaScript 1.2. They provide information about the monitor display.

availHeight	available height in pixels exclusive of task bar etc.
availWidth	available width in pixels exclusive of task bar etc.
colorDepth	base-2 logarithm of number of colors available as bits per pixel for each of the three colors (red, green, and blue). for 128 colors value is 7; for 16 colors value is 3; value 32 yields 16,777,216 colors plus transparency (256 × 256 × 256) "Truecolor").
pixelDepth	bits per pixel; unique to Netscape 4, undefined in IE 6
height	total height of screen in pixels
width	total width of screen in pixels

StyleSheet properties

document.styleSheets[0]	accesses first stylesheet on the page; stylesheet can be external defined with <LINK REL-"stylesheet" . . .> or embedded defined within the paired <STYLE> tag.
href	read-only; URL of external stylesheet
title	read-only; title attribute if defined

Some people say you can immediately change a stylesheet with code:
`document.styleSheets[0].href="new.css"`

I have done this with IE, but it fails under Mozilla

Window properties

frames	an array of window objects, each of which is a frame contained within the window. If window has no frames, the property is empty. Applies to both <FRAME> and <IFRAME> tags.
name	of the window, can be used as target of <A> tag
parent	if current window is a frame, this is a reference to the frame of the window that contains it
self	refers to current window object
top	if window is in a frame, refers to the top-level window that contains the frame
window	same as self
opener	refers to Window object that opened this one, or "null" if opened by user
location	refers to the Location object

Element style declarations

The syntax is like:

```
element.style.fontFamily = "sans-serif"
```

Where *style* is a property that returns a **CSS2Properties** object. This ability to test and set styles applies only to in-line styles. There is a JavaScript property for each CSS1 and CSS2 style atribute. The names in JavaScript are the same as in CSS with the exception of CSS names with hyphens. In this case "camel case" is used. For example:

font-family	fontFamily
border-top-color	borderTopColor

There is no correspondence to CSS class and pseudo-class selectors. It is not possible to access them. It is not possible to refer directly to an element by its *class* name. If you need to do something like this, use the HTML **name** attribute with the function **getElementsByName()**.

There is a way to read/write an element's class:

```
element.className = "special"
if (element.className == "special") element.className = element.className + "
    temp"  // adds a class to the element in order to invoke different styling
```

That said, some developers have created custom JavaScript libraries that mimic the things you cannot do directly in basic JavaScript. See http://ajaxian.com/archives/javascript-css-selector-engine-timeline for a list of possibilities.

HTML Document Object Model

The HTML DOM API consists of nodes that correspond to the various HTML elements. These nodes exist in a hierarchical tree structure. There are different types of nodes. The Core DOM API includes interfaces that can apply to a HTML document including Node, Element, and Document. HTMLDocument is an HTML-specific sub-interface of Document and HTMLElement is an HTML-specific sub-interface of Element. In addition the DOM defines tag-specific interfaces for many HTML elements.

The nodes of an HTML document can be traversed recursively. Individual nodes can be accessed with document methods.

Functions

What a function can do

A function can
(a) perform one or more actions and stop
(b) perform a calculation and return the resulting value and stop

For case (b) the statement return is used to set the value and stop.

Basic syntax

A function has a name, arguments, and code. It is defined by the word "function."

no arguments	function abc() { . . . . }
one argument	function abc(a) { . . . . }
four arguments	function abc(a,b,c,d) { . . . . }
function variable is an expression, no argument	var fv function { . . . . } this defines an unamed function which is invoked: y = fv
function variable is an expression, one argument	var fx = function(x) { . . . . } this defines an unamed function which is invoked: y = fx(20)

A function stops after the last statement or after a return statement.

Functions can be recursive, i.e., they can call themselves.

Function arguments

While a function is declared with a fixed number of arguments, any number can be passed when the function is invoked. All arguments are accessible with the arguments[ ] pseudo-array, zero-based.

arguments.length	provides the number of arguments. When 2 arguments are provided, the value =2

A function can test its value to ensure the corrrect number of arguments were passed.

Individual arguments can be referred to in two ways. For a function abc(x)
(a) if x = 3
(b) arguments[0] = 5

The Arguments object has a property callee that refers to the function that is currently being executed. This allows unnamed functions to invoke themselves recursively.

```
function check(args)
{
var actual = args.length;              // the actual number of arguments
var expected = args.callee.length;     // the expected number of arguments
```

```
if (actual != expected) { throw new Error("error message text"); }
{
```

Can a function be an argument of a second function? Yes.

Bad function names—don't use

"test"

Use as re-usable block of code

Executed when function is called or triggered by an event (i.e., called by event handler)

Example for function named abc:

```
abc()
abc(x)
xyz(2,8)
<A ONCLICK="abc(x)"> . . . </A>
```

Assign a function to a variable

```
functionObjectName = new Function([arg1, arg2, ... argn], functionBody)
```

where:

functionObjectName is a variable or object property or object event handler (e.g., `window.onError`);

argx are arguments to be used by the function;

functionBody is string specifying JavaScript code to be evaluated as the function.

example:

```
var setBGColor = new Function("document.bgColor='antiquewhite'")
```

The function is called: `setBGColor()` and by event handler: `onClick="setBGColor()"`

and by event handler as: `document.form(0).element(0).onClick=setBGColor`

Function objects are evaluated each time they are used. This is less efficient than declaring a function because the latter is compiled. I wonder if the advantage isn't using a variable inside the functionBody, but cannot find an example.

Process Control

Conditional processing

```
if (expression)
    statement;

if (expression) statement;

if (expression)
{   statement1;
    statement2; }

if (expression)
    statement;
else
    statementb;

if (expression)
    statement;
else if (expression2)
    statementb;
else
    statementc;
```

```
switch (expression) {
    case value1:
        statement1; break;
    case value2:
        statement2; break;
    default:
        statement3; break; }

switch (x) {
    case value1:
        return y;
    case value2:
        return z; }
```

Looping

```
while (expression)
    statement
while (expression) {
    statement1;
    statement2; }

do
    statement
while (expression)

for (initialize; test; increment)
    statement
```

example: for (var count=0; count < 10; count++)
document.write(count + "
");

The for statement(s) are done while the test is true.

```
for (variable in object)
    statement;
```

Exit loop prematurely

only for: while, do/while, for, for/in.

In the version with labelname, a line break is not allowed before "labelname".

```
break;
break labelname;

continue;
continue labelname;

labelname:  statement;
```

Example:

```
    loopa:
        for . . .
           {  . . .
            break loopa;
            . . . }
```

Expressions

Literals

numeric 1-7

string "fun"

boolean	true
null	null
array	[2, 3, 5, 7]
variable	I

Operators

array number	[ ]	
function call	()	
increment value	++	(example: a = ++i;)
decrement value	- -	(two dashes)
logical complement	!	(operand is boolean)
multiply	*	
division	/	
remainder	%	
add	+	
subtract	-	
equal	==	
less than	<	
less than or equal	<=	
greater than	>	
greater than or equal	>=	
unequal	!=	
bitwise AND	&	
bitwise OR	|	
bitwise logical AND	&&	
bitwise logical OR	||	
assignment	=	
concatenate	+	

Encoding special characters in URL strings and HTML text

The W3C specification for a URL includes: "Where the local naming scheme uses ASCII characters which are not allowed in the URI, these may be represented in the URL by a percent sign '%' immediately followed by two hexadecimal digits (0–9, A–F) giving the ISO Latin 1 [ISO 8859-1] code for that character."

In addition to URL strings, you may use JavaScript code to format text extracted from a database. That text might include characters that are special to HTML like "&", "<", ">" and paragraph breaks (possible in memo fields). The HTML BLOCKQUOTE tag can handle some but not all problems. The safest approach is to encode disallowed and HTML-reserved characters.

Common disallowed characters and their encoded values:

%20	is a space
%3B	is a semicolon (;)
%21	is an exclamation point (!)
%26	is an ampersand (&)

Two functions address the encoding:

- escape: encodes a string
- unescape: decodes an encoded string

unescape(*encoded string*)
searches for 2- and 4-digit hexidecimal escape sequences and replaces them in the string with their single character ISO Latin 1 equivalent.

Thus
document.write(unescape("Miss%20Piggy%20loves%20Kermit%21")
yields
Miss Piggy loves Kermit!

Arrays

Definition

An array is akin to a table.
Arrays can be nested.
Array items are referenced with an index (x[i]) or by its itemname (x["itemname"]).
Arrays are zero-based: [0] is the first item.

Create an array

- Create an array from list of values:

```
var mailbill = new Array("billstreet", "billcity", "billstate", "billzip");
```

- Create an array and populate it with a function:

```
var geo = new Array(48);
initGeoArray(geo);
. . .
function initGeoArray(a)
{
a[0] = "allca";
a[1] = "alameda";
a[2] = "alpine";
. . .
}
```

- An array item can contain an array.

How many items in an array

```
vararray.length
```

If an object is an array, its length property is "number".

```
if (typeof(e.length) = "number")
```

If an object is not an array, its length property is "undefined".

```
if (typeof(e.length) = "undefined")
```

Enumerate array items

- Use the for/in loop:

```
var a = new Array();
. . .  (populate array)
var i = 0;
for (i in a) alert(i);
```

Assign custom property to array

In this example, biztype is a group of same-named radio buttons.

```
document.soq.biztype.required = true;
```

then document.soq.biztype[i] is not required.

String Manipulation

Two objects:

- String
- RegExp

String concatenation is done with the + operator.

String properties

length	returns integer Example: var s = "abc"; s.length // returns 3

String methods

charAt(n)	extracts the character at a given position (n) from a string
indexOf(substring) indexOf(substring, start)	searches the string for a character or substring; optionally, begin at a numbered position; if none found, returns -1
lastIndexOf(substring) lastIndexOf(substring, start)	searches the string backwards for a character or substring
match(regexp)	pattern matching with a regular expression
replace(regexp, replacement)	search and replace with a regular expression
search(regexp)	search a string for a substring that matches a regular expression

slice(start)
slice(start, end)
	returns a substring in terms of its end position; if end not specified, returns substring through end of string; "end" is position immediately after the end of the slice
	Example: var s = "abcdefg";
	s.slice(0,4) // returns "abcd"
	s.slice(2,4) // returns "cd"
	s.slice(4) // returns "efg"

toLowerCase()	converts all characters to lower case
toUpperCase()	converts all characters to upper case

Regular Expressions

Pattern matching involves Regular Expressions. A Regular Expression is an object, RegExp, that describes a pattern of characters. REs in JavaScript are strongly based on Perl regular expressions. REs have a complicated grammar are and discussed separately.

REs are composed of text between a pair of slashes. The second slash in the pair can be followed by one or more letters which serve to modify the meaning of the pattern.

Example of using regular expression:

```
var re = /^\d+$/;
if (!re.test(e.value))           // then there is an error
…
text.replace(/javascript/gi, "JavaScript");
```

Special Techniques: Existence Determination

Determine if things exist

Determine if object exists: `if (object_name == null)` returns true when object does not exist

Determine if method or property exists: `if (!mywin.opener)` returns true when the Window object mywin was opened by the user (and not another file/window). You can use this technique to test if the browser supports a JavaScript method or property. Example sets the value of an unsupported property:

```
if (!mywin.opener) mywin.opener = self //where mywin = window.open(file,
winname)
```

Determine if variable exists: don't know.

Special Techniques: Windows

Coping with pop-up blockers

Windows XP SP2 includes the Information Bar which automatically blocks "pop-up" windows. Microsoft defines these as any window opened automatically from script, with the exception of **createPopup**(). Common functions that are affected are:

window.open()
showModelessDialog()
showModalDialog()
showHelp()

A window opened as a direct result of user action (e.g., clicking a page element) is not blocked.

You can tell if IE blocks a window: functions that return a window object will return **null** if the window is blocked. Always check the function's return value.

Open different document in the current window

```
location.href="url"
```

One page can open a second page in another window

Use the "onload" attribute in the BODY tag to open a second window at the same time the first page is opened.

```
<HEAD>
<SCRIPT>
function showlog()
// "new" is the name of the window in which the second page is opened
{ window.open("log.html","new","toolbar=no,location=no,directories=no,\
status=no,menubar=yes,scrollbars=yes,resizable=no,copyhistory=no,width=600,\
height=400") }
</SCRIPT>
…
<BODY onload ="showlog()">

* * * * *
function smallwin(u)
{
//opens passed url in new small window
var winsize = "width=" + (.8 * screen.width) + ", height=" + (.6 *
    screen.height) + ", left=80, top=80";
var controls = ", toolbar=yes, location=yes, menubar=yes, status=yes,
    scrollbars=yes, resizable=yes";
window.open(u, "new", winsize + controls);
}
```

Open second page in new window. If window is already open give it the focus.

This code sometimes results in an error in IE 5.1, although still works; it seems that focus() doesn't work consistently.

```
function openwin(file, winname)
{
    mywin = window.open(file, winname)
    mywin.focus()
}}
```

Display a popup window with content and close it remotely

```
<SCRIPT LANGUAGE="JavaScript">
<!--
function launch() {
  self.name = "index";    // name of current window
  remote = open("path/filename.html",    // name of file to be opened
"popup",                  // name of new window
"width=256,height=155,left=50,top=50")     // size and location of new window
//  the popup window can be closed with remote.close("") - I think
}
// -->
</SCRIPT>
```

** Close window that opened the current window - JavaScript

```
window.opener.close()
```

Open new window when current page is closed (by Back or new URL)

```
<body onUnload="open_windowx();">
```

Close current window

```
window.close()
```

Go to top

```
<a href="javascript: self.scrollTo(0,0)">Go to top</a>
```

A page can change its window size and position when it opens

```
window.resize(300,200)
window.moveTo(180,50)
self.moveBy (x,y)         // x is the number of pixels on x axis, y on the y
    axis; can be negative
self.moveTo (xx,yy)     // xx is number of pixels to right of left side, yy
    down from top
self.resizeBy(x,y)        // x is number of pixels of width, y is height; can
    be negative
self.resizeTo(xx,yy)    // xx is number of pixels of width, yy is height
    windows can be no smaller than (100,100)
self.scrollBy(x,y)
self.scrollTo(xx,yy)
```

Open page in full size window

```
<script type="text/javascript">
<!--
function maxwin() {
  if (window.screen) {
    var aw = screen.availwidth;
    var ah = screen.availheight;
    window.moveTo(0,0);
    window.resizeTo(aw, ah) }
}
//-->
</script>
</head>
```

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

OR

Note: setting window height to screen height will hide the status bar at the bottom.

```
function maxwin() {
  window.moveTo(0,0)
  window.resizeTo(screen.width, screen.height)
}
```

OR

```
window.open(filename, “”, “fullscreen, …”)
```

Open window, then write to it

Could be reused, with no separate HTML file

```
function openSmallWind(pic)
{
smwin = window.open(“”,”smallWin”, “dependent, resizable, …”)
var td = smwin.document
td.open()
td.write(‘<html><head><title>Small Window for Temporary
    Display</title>,</head>’)
td.write(‘<body onLoad=”…”>’)
td.write(‘<img src=”pic.jpg”>’)   // or td.write(‘<img src="' + 'Images/' + pic
    + '.jpg">’)  where pic is a variable
td.write(‘</body></html>’)
td.close()
}

function closeIt() {
  if (td != null && td.open) td.close()
}
window.onfocus=closeIt()   //close small window when main window gets the focus
// -->
</script>
```

Open popup window only once

Put this code in the HEAD section:

```
<SCRIPT LANGUAGE="JavaScript">
<!-- This script and many more are available free online at -->
<!-- The JavaScript Source!! http://javascript.Internet.com -->
<!-- Begin
var expDays = 1; // number of days the cookie should last
var page = "http://www.slackerhtml.com/javascript/windows/only-popup-
    once.html";
var windowprops =
    "width=300,height=200,location=no,toolbar=no,menubar=no,scrollbars=no,resiz
    able=yes";
function GetCookie (name) {  var arg = name + "=";  var alen = arg.length;  var
    clen = document.cookie.length;  var i = 0;  while (i < clen) {    var j = i
    + alen;
if (document.cookie.substring(i, j) == arg)      return getCookieVal (j);    i
    = document.cookie.indexOf(" ", i) + 1;    if (i == 0) break;   }  return
    null;}
function SetCookie (name, value) {  var argv = SetCookie.arguments;  var argc =
    SetCookie.arguments.length;  var expires = (argc > 2) ? argv[2] : null;
    var path = (argc > 3) ? argv[3] : null;  var domain = (argc > 4) ? argv[4]
    : null;  var secure = (argc > 5) ? argv[5] : false;  document.cookie = name
    + "=" + escape (value) + ((expires == null) ? "" : ("; expires=" +
    expires.toGMTString())) + ((path == null) ? "" : ("; path=" + path)) +
    ((domain == null) ? "" : ("; domain=" + domain)) +    ((secure == true) ?
    "; secure" : "");}
```

```
function DeleteCookie (name) {  var exp = new Date();  exp.setTime
    (exp.getTime() - 1);  var cval = GetCookie (name);  document.cookie = name
    + "=" + cval + "; expires=" + exp.toGMTString();}var exp = new Date();
    exp.setTime(exp.getTime() + (expDays*24*60*60*1000));
function amt(){var count = GetCookie('count')if(count == null)
    {SetCookie('count','1')return 1}
else {var newcount = parseInt(count) +
    1;DeleteCookie('count')SetCookie('count',newcount,exp)return count   }}
function getCookieVal(offset) {var endstr = document.cookie.indexOf (";",
    offset);if (endstr == -1)endstr = document.cookie.length;return
    unescape(document.cookie.substring(offset, endstr));}
function checkCount() {var count = GetCookie('count');if (count == null)
    {count=1;SetCookie('count', count, exp);window.open(page, "",
    windowprops);}
else {count++;SetCookie('count', count, exp);   }}
//  End
-->
</script>
```

Add the OnLoad event handler to the BODY tag: OnLoad="checkCount()">

Special Techniques: Frames

Break out of someone's frames

```
if (self.parent.frames.length != 0)
    self.parent.location="http://www.mydomain.com"
```

also seen as (but do not understand differences yet):

```
if (self != top) top.location.replace(self.location)
```

Force document to open within a frame

```
if (top == window) top.location.href= “frameset.html”
```

where frameset.html is a fixed file

to use as a generic technique applicable to a set of documents, frameset.html needs to be a dynamic document where the contents of one frame is controlled by a query string and some of the actual html is created dynamically by document.write:

```
if (top == window) top.location.href= “frameset.html?” +
    escape(window.location.href);
```

frameset.html includes:

```
<html>
<head>
var url = location.search ?
    unescape(location.search.substring(1)) : “default.html”;
var html = “”
html += “<frameset rows=’150, *’>”
html += “<frame name=’desktopframe’ src=’desktoplinks.htm’>”
html += “<frame name=’deskcontent’  scr=’” + url + “’>”
html += “<\/frameset>”
</script>
</head>
<script>
document.write(html)
</script>
</html>
```

Special Techniques: Dialog Boxes (modal windows)

There are three kinds of dialog boxes:

a. alert box: has message, OK button, and no title.

b. confirmation box: has message, OK and Cancel buttons, and no title.
c. prompt box: has message, OK and Cancel buttons, title, and field to capture entered text.

Open an "alert" box

```
<script type="text/javascript">
alert("Hello World!")
</script>
```

Open a "confirm" box

```
<script type="text/javascript">
var name = confirm("Press a button")
if (name == true)
{ document.write("You pressed OK") }
else
{ document.write("You pressed Cancel") }
</script>
```

Prompt box

Displays window with title "Explorer User Prompt", label "JavaScript Prompt", buttons OK and Cancel. Entered text is captured as variable.

```
<body>
<script type="text/javascript">
var name = prompt("Please enter your name","")
if (name != null && name != "")
    document.write("Hello " + name)
</script>
```

Force a line break with "\n"

```
alert("Hello you\nFrom me")
```

presents a message box like:

> Hello you
> From me

Special Techniques: Cross Window Scripting

Browser instance (?) knows about all its open windows

When about.html does `window.open("index.html","home")` the file is loaded into the window named "home" if it is already open (regardless of its contents). If the window is not open, it is opened, the file is loaded, and the window is given the focus.
The browser will retain knowledge of a named window after it is closed as long as references to it exist.

It is possible to control window B from window A but only in certain limited ways

Helpful code:
`winb.focus()` – gives focus to the Window object named winb. NOTE: this does not work right in IE5.
`self.blur()` – takes focus away from the current window. This has the effect of the focus() method above ONLY when there are only two browser windows open.
`window.opener` – refers to the Window object that opened the current window. This is the only way to know anything about that window. Can be coupled with the ability to create custom

properties. Does not work in IE5!

Example: `if (window.opener.name = “you”)` …, `if (window.opener.custom = “yes”)`

`window.location.href` – setting this property causes the browser to load a new file.

Example: `window.location.href = myurl`

`window.open (file, windowname)` – opens file in named window; if window already exists, it does not automatically get the focus.

Theoretically, you should be able to create a custom property of the global object (Window object) which can be accessed in a second window: `window.creator = “index”`; the access in the second window is like: `if (window.opener.creator = “index”)`. At least in IE5 a spawned window cannot access it: it is not defined.

The following samples are used to control navigation between two windows—a home page window and a subject window used for all subjects listed on the home page. The code handles the situation where a subject file is opened by the user or some page not the home page. (This code works in IE 5.0, but in IE 5.5 it spawns additional windows.)

```
window.name – “stmain”           // done in home page
/* function openwin opens the named file in the named window.  If the window is
    already open, it makes it active.  Used on the home page; winname is
    “subject” in most cases */
var w = ''                      // variable for window object created by page
    linked to this file
function openwin(file, winname)
{
if (w.location && !w.closed)    // if window object w exists and is not closed
  { w.location.href = file }    // load page into its location.href
else
  { w=window.open(file,winname);  // open the new window
//    if (!w.opener)  {w.opener = self} ;
  }
self.blur()
}

// next function used to link (back) to Home page from a subject page
function openhome(file)
{
var home = ''
if (window.name == "subject")  // then I was opened by home page
    { home = window.open(file, "stmain")
          self.blur() }
else
    {
    window.name = "subject"
    home = window.open(file, "stmain")
    }
}
```

Special Techniques: Messages

Set text in status bar

```
window.status = "put your message here"
```

Status bar message

```
<BODY OnLoad="window.defaultStatus='Hello!';">
```

Scroll text in the browser's status bar to the left

```
<HTML>
<HEAD>
<TITLE>The Herballadies : MicroWater</TITLE>
<script language="JavaScript">
<!-- Begin code
function scroll(seed)
{
    var m1  = "Welcome to The Herballadies call 1-888-258-5949";
    var msg=m1;
    var out = " ";
    var c   = 1;
    if (seed > 100) {
       seed--;
       var cmd="scroll(" + seed + ")";
       timerTwo=window.setTimeout(cmd,100);
    }
    else if (seed <= 100 && seed > 0) {
       for (c=0 ; c < seed ; c++) {
           out+=" ";
       }
       out+=msg;
       seed--;
       var cmd="scroll(" + seed + ")";
       window.status=out;
       timerTwo=window.setTimeout(cmd,100);
    }
    else if (seed <= 0) {
       if (-seed < msg.length) {
          out+=msg.substring(-seed,msg.length);
          seed--;
          var cmd="scroll(" + seed + ")";
          window.status=out;
          timerTwo=window.setTimeout(cmd,100);
       }
       else {
          window.status=" ";
          timerTwo=window.setTimeout("scroll(100)",75);
       }
    }
}
// -- End code -->
</script>
</HEAD>
<BODY BACKGROUND="sky.jpg" BGCOLOR=#CCCCFF TEXT=#000000 LINK=#0000FF
VLINK=#0000AA ALINK=#FFFF00
onLoad="timerONE=window.setTimeout('scroll(100)',500);">
<CENTER> The Herballadies</CENTER>
```

Special Techniques: Element Dimensions

Determine width of the browser window

You may want to know how wide a window is. This can be helpful in designing a page. It may also be helpful in other contexts.

You can write a JavaScript program to display the width of the browser window. Display the width in an alert box when the page loads. Locate the program in the BODY element. When the page loads an alert box presents the width. Then adjust the width of the page to suit yourself, then refresh the contents of the page. You'll see a message box with the current window width.

```
<script type="text/javascript">
var w = document.body.offsetWidth
var m = "The width of the browser window: " + w
```

```
alert(m);
</script>
```

Change height of element

You can change the height of an element after it has been rendered, meaning the code must be located below the HTML of the element or executed as a load event and resize event. In this example, a DIV (ID=xx) is contained within a TD table cell (ID=cell1), the table has 2 rows and 2 columns, but only 3 cells as the first cell spans both rows. I want to force the inner DIV, which has a colored background, to fill the first cell. The final height is reduced by 20 pixels to accommodate a padding set in the stylesheet.

In this example offsetHeight is a number with an intrinsic UOM of pixels, it is a element property. To the contrary, paddingBottom is an attribute of style and has a value that is a string that must end in a UOM. You must be careful to add the UOM in code.

```
<script type="text/javascript">
var cellheight= document.getElementById("cell1").offsetHeight;
var textheight = document.getElementById("xx").offsetHeight;
var diff = cellheight - textheight;
var diff = cellheight - textheight;
var adj = diff - 20
document.getElementById("xx").style.paddingBottom = adj + "px"
</script>
```

Special Techniques: Browser Tools

Refresh page

```
location.reload()
```

Print the page with JavaScript

```
window.print()

<A HREF=”javascript:window.print()”>Click to print this page</A>

<FORM><INPUT TYPE=”button” onClick=”window.print()”></FORM>

<a href='javascript:;' onClick='window.print();return false'>Print this
    page.</a>
```

Go Back with Link in Text

```
<a href="javascript:history.go(-1)">Back</a>
```

Print and close a different document with Link in Text

```
<script  type=”text/javascript”>
function printclose()
{ var x = window.open(“different.html”, “whatever”)
  x.print()
  x.close
}
</script>
. . .
<a href="javascript:printclose()">. . . </a>
```

prompts for print options, then closes the window

Determine browser

- Call this function in the <BODY onload=”getBrowser()”> tag.

```
function getBrowser()
{
var b = new Object();
b.name = navigator.appName;
b.version = parseInt(navigator.appVersion);
b.agent = navigator.userAgent;
document.soq.browser.value = b.name + ", " + b.version  + ", " +  b.agent;
}
```

- if user agent string contains "SV1" then the browser is IE with SP2

```
If (b.agent.indexOf("SV1")!= -1   // browser is IE with SP2
```

Browser info

```
<body>
<script type="text/javascript">
document.write("BROWSER: ")
document.write(navigator.appName + "<br>")
document.write("BROWSERVERSION: ")
document.write(navigator.appVersion + "<br>")
document.write("CODE: ")
document.write(navigator.appCodeName + "<br>")
document.write("PLATFORM: ")
document.write(navigator.platform + "<br>")
document.write("REFERRER: ")
document.write(document.referrer + "<br>")
</script>
</body>
```

yields:
BROWSER: Microsoft Internet Explorer
BROWSERVERSION: 4.0 (compatible; MSIE 5.01; Windows NT 5.0; PGE)
CODE: Mozilla
PLATFORM: Win32
REFERRER: http://www.w3schools.com/js/tryit.asp?filename=tryjs_browserdetails

Special Techniques: Forms

Set focus on a particular form element

It is useful to set the focus on the first element when a form opens.

```
<head>
<script type="text/javascript">
// puts focus on first element in first form
function setfocus(a,b)
{  document.forms[a].elements[b].focus()  }
</script>
</head>
<body  onLoad="setfocus(0,0)">
<form>
<input type="text" name="field" size="30">
<input type="text" name="userid" size="10">
<input type="button" value="Get Focus">
</form>
</body>
```

Show/hide text

1. Group text to be shown/hidden by division and assign each an id.
   ```
   <DIV id=all>
   . . .
   ```

```
    </DIV>
```

2. In script just before </BODY> tag set style property:
```
    var e = document.getElementById("all");
    e.style.display = "none";          // hide division
```
3. When appropriate button is clicked,
```
    var e = document.getElementById("all");
    e.style.display = "block";         // show division
```

Special Techniques: Editing Form

Convert to upper case

```
field.value = field.value.toUpperCase();
```

Convert to lower case

```
field.value = field.value.toLowerCase();
```

Check for numeric values

form element includes event procedure;
```
    onChange="checkNumeric(this, 0)"
```

JavaScript function:
```
function checkNumeric(which, x)
{
// the optional second parameter allows you to edit for presence of dash
var digits;
if (x == 1)
    digits="0123456789-;.";
else
    digits="0123456789";
var temp;
for (var i=0;i<which.value.length;i++)
    {
    temp=which.value.substring(i,i+1);
    if (digits.indexOf(temp)==-1)
        {
    alert("Enter only digits!");
        which.focus();
        return false;
        }
    }
return true;
}
```

Another way to edit for numbers—integers uses regular expression:
```
var re = /^\d+$/;
if (!re.test(e.value))          // then there is an error
```

Edit dates

Uses regular expression to check for the following valid date formats:
MM/DD/YY MM/DD/YYYY MM-DD-YY MM-DD-YYYY
```
function isDate(field) {
    var dateStr = field.value;
        if (dateStr != "") {
//          var datePat = /^(\\d{1,2})(\\/|-)(\\d{1,2})\\2(\\d{4})\$/; //
    requires 4 digit year
            var datePat = /^(\d{1,2})(\/|-)(\d{1,2})\2(\d{4})$/; // requires 4
    digit year
         var matchArray = dateStr.match(datePat); // is the format ok?
               if (matchArray == null) {
```

```
                    alert(dateStr + " Date is not in a valid format(MM-DD-
    YYYY).");
                        field.focus();
                        return false;
                }
                month = matchArray[1]; // parse date into variables
                day = matchArray[3];
                year = matchArray[4];
                if (month < 1 || month > 12) { // check month range
                    alert("Month must be between 1 and 12.");
                        field.focus();
                        return false;
                }
                if (day < 1 || day > 31) {
                    alert("Day must be between 1 and 31.");
                        field.focus();
                        return false;
                }
                if ((month==4 || month==6 || month==9 || month==11) && day==31)
    {
                    alert("Month "+month+" doesn't have 31 days!")
                        field.focus();
                        return false;
                }
        if (month == 2) { // check for february 29th
                    var isleap = (year % 4 == 0 && (year % 100 != 0 || year %
    400 == 0));
                        if (day>29 || (day==29 && !isleap)) {
                           alert("February " + year + " doesn't have " + day + "
    days!");
                                field.focus();
                                return false;
                        }
                }
    }
        return true;
}
```

Handle radio buttons

- When accessing elements using the document.soq.elements[i] array, you will access individual radio buttons. In order to refer to the group of radio buttons of the same name, you must use the element name:

```
var e = document.forms[0].elements[i];
var radiogroup = document.forms[0].elements[e.name];
```

- Check to see if one radio button in its group is checked:

```
if (e.type == "radio")
    {
    if (!isVisible(e))
        return "N";
    var radiogroup = document.soq.elements[e.name];  // get the whole group
    var itemchecked = false;
    for(var j = 0 ; j < radiogroup.length ; j++)
        {
        if (radiogroup[j].checked)
            itemchecked = true;
        }
    if ((!itemchecked) && (e == radiogroup[0]))     // do only once for group
    // handle the error – but only once for the group
    ...
    }
```

- Test an index:

```
if (document.soq.biztype[0])
```

- How to tell how many elements are in a group?

```
document.soq.biztype.length
```

- Only way to tell if a form element document.soq.elements[varname] is a radio button:

```
if (typeof(e.length) = “number”)

e = document.soq.elements[i];
if (e.type == “radio”)
```

Edit radio button or check box

Many times you will want to be able to get the selected radio button and/or check box in a browser environment. For the radio button, you will want to check the value and perform some task. Or, you will want to verify that one radio button has been selected. For check boxes, you may want to see which ones are selected or you could want to check that at least one is selected. Here's a couple of functions that work with radio buttons and check boxes. These are set up to be generic, so they can be reusable.

```
function getSelectedRadio(buttonGroup) {
   // returns the array number of the selected radio button or -1 if no button
    is selected
   if (buttonGroup[0]) { // if the button group is an array (one button is not
    an array)
      for (var i=0; i<buttonGroup.length; i++) {
         if (buttonGroup[i].checked) {
            return i
         }
      }
   } else {
      if (buttonGroup.checked) { return 0; } // if the one button is checked,
    return zero
   }
   // if we get to this point, no radio button is selected
   return -1;
} // Ends the "getSelectedRadio" function

function getSelectedRadioValue(buttonGroup) {
   // returns the value of the selected radio button or "" if no button is
    selected
   var i = getSelectedRadio(buttonGroup);
   if (i == -1) {
      return "";
   } else {
      if (buttonGroup[i]) { // Make sure the button group is an array (not just
    one button)
         return buttonGroup[i].value;
      } else { // The button group is just the one button, and it is checked
         return buttonGroup.value;
      }
   }
} // Ends the "getSelectedRadioValue" function

function getSelectedCheckbox(buttonGroup) {
   // Go through all the check boxes. return an array of all the ones
   // that are selected (their position numbers). if no boxes were checked,
   // returned array will be empty (length will be zero)
   var retArr = new Array();
   var lastElement = 0;
   if (buttonGroup[0]) { // if the button group is an array (one check box is
    not an array)
      for (var i=0; i<buttonGroup.length; i++) {
         if (buttonGroup[i].checked) {
            retArr.length = lastElement;
            retArr[lastElement] = i;
            lastElement++;
         }
```

```
      }
   } else { // There is only one check box (it's not an array)
      if (buttonGroup.checked) { // if the one check box is checked
         retArr.length = lastElement;
         retArr[lastElement] = 0; // return zero as the only array value
      }
   }
   return retArr;
} // Ends the "getSelectedCheckbox" function

function getSelectedCheckboxValue(buttonGroup) {
   // return an array of values selected in the check box group. if no boxes
   // were checked, returned array will be empty (length will be zero)
   var retArr = new Array(); // set up empty array for the return values
   var selectedItems = getSelectedCheckbox(buttonGroup);
   if (selectedItems.length != 0) { // if there was something selected
      retArr.length = selectedItems.length;
      for (var i=0; i<selectedItems.length; i++) {
         if (buttonGroup[selectedItems[i]]) { // Make sure it's an array
            retArr[i] = buttonGroup[selectedItems[i]].value;
         } else { // It's not an array (there's just one check box and it's
    selected)
            retArr[i] = buttonGroup.value;// return that value
         }
      }
   }
   return retArr;
} // Ends the "getSelectedCheckBoxValue" function
```

To use one of these functions, just make a call and pass the radio button or check box object. For example, if you want to find out if at least one check box is selected and the check box field name is MyCheckBox, then write the following statements:

```
var checkBoxArr = getSelectedCheckbox(document.forms[0].MyCheckBox);
if (checkBoxArr.length == 0) { alert("No check boxes selected"); }
```

Check for whitespace characters

```
function isBlank(e)
{
// e is a form element
var v = e.value;
if ((v == null ) || (v == "") || (v.length == 0))
    return true;
// check for whitespace characters in form element
for (var i = 0; i < v.length; i++)
    {
    var c = v.charAt(i);
    if ((c != ' ') && (c != "\n") && (c != ""))
        return false;
    }
return true;
}
```

Perform all edits

```
<FORM onsubmit=”return EditForm(this)”  . . .>
. . .
</FORM>
. . .
function setEditProperties(f)
{
// f is form name; properties are used by function EditForm
// assume all radio buttons are required; they have no initial value
document.soq.product.required = true;
document.soq.contact.required = true;
```

```
document.soq.name.required = true;
document.soq.zip.zip = true;
setBillAddress();
var mailbill = new Array("billstreet", "billcity", "billstate", "billzip");
    // all or none
document.soq.billzip.allornone = mailbill;
document.soq.billzip.groupname = "Complete Mailing/Bill Address";
document.soq.taxid.taxid = true;
document.soq.naics.integer = true;
document.soq.email.email = true;
document.soq.url.url = true;
var geo = new Array(48);
initGeoArray(geo);
document.soq.allca.atleast = geo;                // at least one of a group of
    checkboxes is required
document.soq.allca.groupname = "Geographical Service Area";
document.soq.gross.amount = true;
document.soq.start1.monthyear = true;
document.soq.cphone1.telephone = true;
document.soq.expdate.date = true;
. . .
}

function EditForm(f)
{
// this catches all the errors and reports them back at once
setEditProperties(f);

var ErrorCount = 0;
var msg;
var empty_fields = "";
var errors = "";
var n = 0;
var em = "";
var req = "";
// check for required fields
// isEdit() returns an error message or "N" if no error
for (var i = 0; i < document.soq.length; i++)        // loop through elements
    in the form
    {
    var e = document.soq.elements[i];
    em = isEdit(e);
    if ((em != "undefined") && (em != "N"))         // error
        {
        if ((e.required) || (e.type == "radio") || (e.atleast))
            empty_fields += em;
        else
            errors += em;
        }
    }

if (!empty_fields && !errors)
    return true;

msg  = "_____________________________________________________________________\n\n";
msg += "The form was not submitted because of the following error(s).\n";
msg += "Please correct these error(s) and resubmit. \n";
msg += "_____________________________________________________________________\n\n";
if (empty_fields)
    msg += "- The following required field(s) are empty:" + empty_fields +
    "\n";
if (errors)
    msg += "\n" + errors;
alert(msg);
return false;
}
function isEdit(e)
```

```
{
// find the first error for a field
// return error message or "N" (if no error)
var msg = "";
if (e.required)
    {
    if (!isVisible(e))
        return "N";
    switch (e.type)
        {
        case 'text':
            {
            if (isBlank(e))
                {
                msg = "\n      " + expandName(e.name);
                return msg;
                }
            break;
            }
        case 'textarea':
            {
            if (isBlank(e))
                {
                msg = "\n      " + expandName(e.name);
                return msg;
                }
            break;
            }
        case 'select-one':
            {
            if (e.selectedIndex < 1)
                {
                msg = "\n      " + expandName(e.name);
                return msg;
                }
            break;
            }
        }
    }

if (e.integer)
    {
    msg = isInteger(e);
    if ((msg != "undefined") && (msg != "N"))
        return msg;
    }

if (e.zip)
    {
    msg = isZip(e);
    if ((msg != "undefined") && (msg != "N"))
        return msg;
    }

if (e.type == "radio")
    {
    if (!isVisible(e))
        return "N";
    var radiogroup = document.soq.elements[e.name];      // get the whole set
    of radio buttons
    var itemchecked = false;
    for(var j = 0 ; j < radiogroup.length ; j++)
        {
        if (radiogroup[j].checked)
            itemchecked = true;
        }
```

```
    if ((!itemchecked) && (e == radiogroup[0]))            // do only once for
    group
        {
// hope to replace with generic code - to identify division containing form
    element
        var x = document.getElementById("dod");
        if (e.name != "cpuc")
            {
            msg = "\n      " + expandName(e.name);
            return msg;
            }
        else
        if (x.style.display == "block")
            {
            msg = "\n      " + Name(e.name);
            return msg;
            }
        }
    }

if ((e.atleast) && (e.type == "checkbox"))
    {
    if (!isVisible(e))
        return "N";
    var groupchecked = false;
    for (var k = 0; k < e.atleast.length; k++)
        {
        var en = e.atleast[k];              // element name
        var c = document.soq.elements[en].checked;
        if (c)
            {
            groupchecked = true;
            break;
            }
        }
    if (groupchecked == false)
        {
        msg = "\n      " + e.groupname;
        return msg;
        }
    }

if ((e.atleast) && (e.type == "text"))
    {
    if (!isVisible(e))
        return "N";
    var groupchecked = false;
    for (var k = 0; k < e.atleast.length; k++)
        {
        if (!isBlank(e))
            {
            groupchecked = true;
            break;
            }
        }
    if (groupchecked == false)
        {
        msg = "\n      " + e.groupname;
        return msg;
        }
    }
if (e.allornone)
    {
    if (!isVisible(e))
        return "N";
    var cnterror = 0;                       // count non-bland fields
    for (var k = 0; k < e.allornone.length; k++)
```

```
        {
        var en = e.allornone[k];              // element name
        var ele = document.soq.elements[en];
        if ((ele.type == "text") || (ele.type == "textarea"))
            {
            if (!isBlank(ele))
                {
                cnterror++;
                }
            }
        else
        if (ele.type == "select-one")
            {
            if (ele.selectedIndex > 0)
                {
                cnterror++;
                }
            }

        }
    if ((cnterror > 0) && (cnterror < e.allornone.length))
        return "\n -  " + e.groupname + " must be present or absent";
    }

if (e.telephone)
    {
    msg = isTelephone(e);
    if ((msg != "undefined") && (msg != "N"))
        return msg;
    }

if (e.amount)
    {
    if (!isVisible(e))
        return "N";
    msg = isAmount(e);
    if ((msg != "undefined") && (msg != "N"))
        return msg;
    }

if (e.date)
    {
    if (!isVisible(e))
        return "N";
    msg = isSOQDate(e);
    if ((msg != "undefined") && (msg != "N"))
        return msg;
    }
if (e.taxid)
    {
    msg = isTaxId(e);
    if ((msg != "undefined") && (msg != "N"))
        return msg;
    }

if (e.email)
    {
    msg = isEmail(e);
    if ((msg != "undefined") && (msg != "N"))
        return msg;
    }

if (e.url)
    {
    msg = isURL(e);
    if ((msg != "undefined") && (msg != "N"))
        return msg;
```

```
    }

if (e.monthyear)
    {
    msg = isMonthYear(e);
    if ((msg != "undefined") && (msg != "N"))
        return msg;
    }

return "N";
}
function isBlank(e)
{
var v = e.value;
if ((v == null ) || (v == "") || (v.length == 0))
    return true;
// check for whitespace characters in form element
for (var i = 0; i < v.length; i++)
    {
    var c = v.charAt(i);
    if ((c != ' ') && (c != "\n") && (c != ""))
        return false;
    }
return true;
}
function isInteger(e)
{
// all non-blank characters must be digits; may have trailing blanks; may be
    all blank
if (e.type != "text")
    {
    alert("error! " + expandName(e.name) + " cannot be integer because type = "
    + e.type);
    return "N";
    }
if (isBlank(e))
    return "N";
/*
var digits = "0123456789";
var temp;
for (var i = 0; i < e.value.length; i++)
    {
    temp = e.value.substring(i, i+1);
    alert(e.name + ", " + e.value + ", " + temp + " position in digits = " +
    digits.indexOf(temp));
    if (digits.indexOf(temp) == -1)
        return "\n       " + expandName(e.name) + " should be integer and non-
    zero";
    }
if (e.value > 0)
    return "N";
else
    return "\n       " + expandName(e.name) + " should be integer and non-
    zero";
-- */
var re = /^\d+$/;
if ((!re.test(e.value)) || (e.value == 0) || (e.value == "0"))
    {
    if (e.required)
        return "\n      " + expandName(e.name) + " should be integer and non-
    zero";
    else
        return "\n -  " + expandName(e.name) + " should be integer and non-
    zero";
    }
return "N";
}
```

```
function isZip(e)
{
// all 5 characters must be digits and non-zero
if (e.type != "text")
    {
    alert("error! " + expandName(e.name) + " cannot be zip because type = " +
    e.type);
    return "N";
    }
var re = /^\d+$/;           // match beginning and end
if ((e.value.length != 5) || (!re.test(e.value)) || (e.value == 0))    // if
    not integer, or length not = 5, or zero
   {
    if (e.required)
        return "\n      " + expandName(e.name) + " should be 5 digits and non-
    zero";
    else
        return "\n -  " + expandName(e.name) + " should be 5 digits and non-
    zero";
    }
return "N";
}
function isTelephone(e)
{
if (e.type != "text")
    {
    alert("error! " + expandName(e.name) + " cannot be telephone because type =
    " + e.type);
    return "N";
    }
if (isBlank(e))
    return "N";
var savePhone = e.value;
var phoneDelimiters  = "[()- ";
var normalizedPhone = stripCharsInBag(e.value, phoneDelimiters);            //
    remove blanks, dashes, and parentheses
e.value = normalizedPhone;
if ((e.value.length != 10) || (isInteger(e) != "N"))
   {
    e.value = savePhone;
    if (e.required)
        return "\n      " + expandName(e.name) + " should be 10-digit telephone
    number";
    else
        return "\n -  " + expandName(e.name) + " should be 10-digit telephone
    number";
    }
e.value = reformat(normalizedPhone, "", 3, "-", 3, "-", 4);        //123-456-
    7890
return "N";
}
function isTaxId(e)
{
if (isBlank(e))
    return "N";
var saveid = e.value;
var bag = "-";
e.value = stripCharsInBag(e.value, bag);
if ((!isInteger(e)) || (e.value.length != 9))
    {
    if (e.required)
        return "\n      " + expandName(e.name) + " must be valid tax id";
    else
        return "\n -  " + expandName(e.name) + " must be valid tax id";
    }
e.value = reformat(e.value, "", 2, "-", 7);
return "N";
```

```
}
function isEmail(e)
{
if (isBlank(e))
    return "N";
var saveid = e.value;
var reEmail = /^.+\@.+\..+$/
if (!reEmail.test(e.value))
    {
    if (e.required)
        return "\n      " + expandName(e.name) + " must be valid email
    address";
    else
        return "\n -  " + expandName(e.name) + " must be valid email address";
    }
return "N";
}
function isURL(e)
{
if (isBlank(e))
    return "N";
var saveid = e.value;
var reURL = /\.+/
if (!reURL.test(e.value))
    {
    if (e.required)
        return "\n      " + expandName(e.name) + " must be valid URL";
    else
        return "\n -  " + expandName(e.name) + " must be valid URL";
    }
return "N";
}
function stripCharsInBag(s, bag)
{
//builds return string from characters in s that are not in bag
var returnString = "";
for (var i = 0; i < s.length; i++)
    {
    var c = s.charAt(i);
    if (bag.indexOf(c) == -1)
        returnString += c;
    }
return returnString;
}
function reformat(s)
{
/*  ========
    Takes one string argument and any number of other arguments (integers
    and/or strings).
    The odd-numbered (e.g., 1, 3, 5) other argument must be a string.  It can
    be "".
    reformat processes the other arguments in order one by one.
    If the other argument is an integer, reformat appends that number of
    sequential characters from s to the
        returnString.
    If the other argument is a string, reformat appends that string to the
    returnString.
    =======  */
var returnString = "";
var arg;
var sPos = 0;
for (var i = 1; i < reformat.arguments.length; i++)     // start with second
    argument
    {
    arg = reformat.arguments[i];
    if (i % 2 == 1)             // odd-numbered argument
        returnString += arg;
```

```
    else
        {
        returnString += s.substring(sPos, sPos + arg);
        sPos += arg;
        }
    }
return returnString;
}
function isAmount(e)
{
if (isBlank(e))
    return "N";
var saveamt = e.value;
// strip off $,+
var bag = "$,+";
var result1 = stripCharsInBag(e.value, bag);
// find .
var x = result1.indexOf(".");           // position of . in string
// if . present, strip it and following characters
if (x != -1)
    {
    e.value = result1.substring(0, x);
    if ((isInteger(e) != "N") && (isInteger(e) != "undefined"))
        {
        e.value = saveamt;
        if (e.required)
            return "\n      " + expandName(e.name) + " should be numeric";
        else
            return "\n -  " + expandName(e.name) + " should be numeric";
        }
    }
else
    {
    e.value = result1;
    if ((isInteger(e) != "N") && (isInteger(e) != "undefined"))
        {
        e.value = saveamt;
        if (e.required)
            return "\n      " + expandName(e.name) + " should be numeric and
    non-zero";
        else
            return "\n -  " + expandName(e.name) + " should be numeric and non-
    zero";
        }
    }
return "N";
}
function isSOQDate(e)
{
if (isBlank(e))
    return "N";
// parse value into MM, DD, YY or MM, DD, YYYY; delimiter required
var dmonth;
var dday;
var dyear;
var re = new RegExp ('/', 'gi');
e.value = e.value.replace(re, "-");         // replace / by -
var delim1 = e.value.indexOf("-");
var delim2;
if (delim1 == -1)           // no dash in value
    {
        if (e.required)
            return "\n      " + expandName(e.name) + " should be valid date MM-
    DD-YYYY";
        else
            return "\n -  " + expandName(e.name) + " should be valid date MM-
    DD-YYYY";
```

```
    }
else                            // dash in value
    {
    dmonth = e.value.substring(0, delim1);
    delim2 = e.value.indexOf("-", delim1 + 1);
    if (delim2 == -1)           // no / in value
        {
        if (e.required)
            return "\n      " + expandName(e.name) + " should be valid date MM-
    DD-YYYY";
        else
            return "\n -  " + expandName(e.name) + " should be valid date MM-
    DD-YYYY";
        }
    dday  = e.value.substring(delim1 + 1, delim2);
    dyear = e.value.substring(delim2 + 1, e.value.length);
    }
if (!isDate(dyear, dmonth, dday))
    {
    if (e.required)
        return "\n      " + expandName(e.name) + " should be valid date MM-DD-
    YYYY";
    else
        return "\n -  " + expandName(e.name) + " should be valid date MM-DD-
    YYYY";
    }
return "N";
}
function isMonthYear(e)
{
if (isBlank(e))
    return "N";
// parse value into MM, YYYY; delimiter required
var dmonth;
var dyear;
var delim1 = e.value.indexOf("-");
if (delim1 == -1)           // no dash in value
    {
    delim1 = e.value.indexOf("/");
    if (delim1 == -1)           // no / in value
        {
        if (e.required)
            return "\n      " + expandName(e.name) + " should be valid month-
    year MM-YYYY";
        else
            return "\n -  " + expandName(e.name) + " should be valid month-year
    MM-YYYY";
        }
    dmonth = e.value.substring(0, delim1);
    dyear = e.value.substring(delim1 + 1, e.value.length);
    }
else                            // dash in value
    {
    dmonth = e.value.substring(0, delim1);
    dyear = e.value.substring(delim1 + 1, e.value.length);
    }
if ((!isMonth(dmonth)) || (!isYear(dyear)))
    {
    if (e.required)
        return "\n      " + expandName(e.name) + " should be valid month-year
    MM-YYYY";
    else
        return "\n -  " + expandName(e.name) + " should be valid month-year MM-
    YYYY";
    }
return "N";
}
```

Special Techniques: Change Images

Images can be changed once displayed.

Special Techniques: Floating Menu

This was copied from www.asianinc.org. This code appears just before the </BODY> tag. It creates a vertical menu that floats (so it appears on the screen in the same place) and that can be dragged to a different location.

```
<!-- vmenu  -->
<img name='awmMenuPathImg-vMenu' id='awmMenuPathImg-vMenu'
    src='scripts/awmmenupath.gif' alt=''>
<script type='text/javascript'>var MenuLinkedBy='AllWebMenus [2]', awmBN='494';
    awmAltUrl='';</script>
<script src='scripts/vMenu.js' language='JavaScript1.2'
    type='text/javascript'></script>
<script type='text/javascript'>awmBuildMenu();</script>
<!-- vMenu ends -->
```

The following code comprises vMenu.js:

```
//----------vMenu---------------
var awmMenuName='vMenu';
var awmLibraryPath='/vMenu';
var awmImagesPath='/vMenu';
var awmSupported=(navigator.appName +
    navigator.appVersion.substring(0,1)=="Netscape5" || document.all ||
    document.layers || navigator.userAgent.indexOf('Opera')>-1)?1:0;
if (awmAltUrl!='' && !awmSupported) window.location.replace(awmAltUrl);
if (awmSupported){
var awmMenuPath;
if (document.all) mpi=document.all['awmMenuPathImg-vMenu'].src;
if (document.layers) mpi=document.images['awmMenuPathImg-vMenu'].src;
if (navigator.appName + navigator.appVersion.substring(0,1)=="Netscape5" ||
    navigator.userAgent.indexOf('Opera')>-1)
    mpi=document.getElementById('awmMenuPathImg-vMenu').src;
awmMenuPath=mpi.substring(0,mpi.length-16);
var nua=navigator.userAgent,scriptNo=(nua.indexOf('Gecko')>-
    1)?2:((document.layers)?3:((nua.indexOf('Opera')>-
    1)?4:((nua.indexOf('Mac')>-1)?5:1)));
document.write("<SCRIPT
    SRC='"+awmMenuPath+awmLibraryPath+"/awmlib"+scriptNo+".js'><\/SCRIPT>");
var n=null;
awmzindex=1000;
}

var awmSubmenusFrame='';
var awmSubmenusFrameOffset;
var awmOptimize=1;
function awmBuildMenu(){
if (awmSupported){
awmCreateCSS(1,2,1,'#FFFFFF','#000000',n,'bold 12pt
    Verdana',n,'outset',2,'#FF0000',0,4)
awmCreateCSS(0,1,0,n,'#BFBFBF',n,n,n,'outset',0,n,0,0);
awmCreateCSS(1,2,1,'#0000FF','#EFEFEF',n,'bold 10pt
    Verdana',n,'none',0,'#0000FF',3,1)
awmCreateCSS(0,2,1,'#FF0000','#EFEFEF',n,'bold 10pt
    Verdana',n,'solid',1,'#FF0000',3,1)
awmCreateCSS(1,2,1,'#0000FF','#DFDFDF',n,'bold 10pt
    Verdana',n,'none',0,'#0000FF',3,1)
```

```
awmCreateCSS(0,2,1,'#FF0000','#DFDFDF',n,'bold 10pt
    Verdana',n,'solid',1,'#FF0000',3,1)
awmCreateCSS(1,2,1,'#0000FF','#DFDFDF',n,'bold 9pt
    Verdana',n,'none',0,'#0000FF',3,1)
awmCreateCSS(0,2,1,'#FF0000','#DFDFDF',n,'bold 9pt
    Verdana',n,'solid',1,'#FF0000',3,1)
awmCreateCSS(1,2,1,n,n,n,'9pt Verdana',n,'none',0,n,0,1)
awmCreateCSS(0,1,0,n,n,n,n,n,'outset',0,n,0,0);
awmCreateCSS(0,1,0,n,n,n,n,n,'none',0,n,0,0);
awmCreateCSS(1,2,1,'#0000FF','#EFEFEF',n,'bold 10pt
    Verdana',n,'none',0,'#FF0000',3,1)
awmCreateCSS(0,2,1,'#FF0000','#EFEFEF',n,'bold 10pt Verdana',n,'solid',1,n,3,1)
awmCreateCSS(1,2,1,n,n,n,'9pt Verdana',n,'outset',0,n,0,1)
var s0=awmCreateMenu(0,0,0,0,1,0,1,2,0,2,147,0,0,1,0,"Main Menu","You can drag
    this menu to a new location",n,1,0,1,0,n,n);
it=s0.addItem(2,3,3,"Home",n,n,"","",n,n,n,"../index.html",n);
it=s0.addItem(4,5,5,"News/Events",n,n,"","",n,n,n,"../newsEvents.html",n);
it=s0.addItem(2,3,3,"Census Info Center",n,n,"","",n,n,n,"../census.html",n);
var s1=it.addSubmenu(0,0,1,0,0,0,0,9,8,n,"",n,1,0,1,1,n,n);
it=s1.addItem(2,3,3,"CIC Home",n,n,"","",n,n,n,"../census.html",n);
it=s1.addItem(4,5,5,"Publications",n,n,"","",n,n,n,"../census_pub.html",n);
it=s1.addItem(2,3,3,"Census Links",n,n,"","",n,n,n,"../census_links.html",n);
it=s1.addItem(4,5,5,"Contact",n,n,"","",n,n,n,"../census_contact.html",n);
it=s0.addItem(4,5,5,"Housing",n,n,"","",n,n,n,"../housing.html",n);
var s1=it.addSubmenu(0,0,1,0,0,0,0,10,8,n,"",n,1,0,1,1,n,n);
it=s1.addItem(4,5,5,"About Housing",n,n,"","",n,n,n,"../housing.html",n);
it=s1.addItem(2,3,3,"Affordable
    Housing",n,n,"","",n,n,n,"../aHousing.html","new");
it=s1.addItem(4,5,5,"Single-Family
    Housing",n,n,"","",n,n,n,"../singleFamilyHousing.html","new");
it=s1.addItem(11,3,12,"First Time
    Homebuyer",n,n,"","",n,n,n,"../homebuyers.html","new");
it=s0.addItem(2,3,3,"Small Business",n,n,"","",n,n,n,"../business.html",n);
it=s0.addItem(6,7,7,"WMBE<br>Clearinghouse",n,n,"","",n,n,n,"../clearinghouse.h
    tml",n);
var s1=it.addSubmenu(0,0,1,0,0,0,0,10,8,n,"",n,1,0,1,1,n,n);
it=s1.addItem(4,5,5,"Background
    Info",n,n,"","",n,n,n,"../clearinghouse.html",n);
it=s1.addItem(2,3,3,"Application
    Form",n,n,"","",n,n,n,"../wmbe_application_form.html",n);
it=s1.addItem(4,5,5,"Printable
    Brochure",n,n,"","",n,n,n,"../wmbe_brochure.html",n);
it=s1.addItem(2,3,3,"FAQs",n,n,"","",n,n,n,"../wmbe_faq.html",n);
it=s1.addItem(4,5,5,"Contact Us",n,n,"","",n,n,n,"../wmbe_contact.html",n);
it=s0.addItem(2,3,3,"Social  Programs",n,n,"","",n,n,n,"../social.html",n);
it=s0.addItem(4,5,5,"Company Info",n,n,"","",n,n,n,"../contact.html",n);
var s1=it.addSubmenu(0,0,1,0,0,0,0,10,13,n,"",n,1,0,1,1,n,n);
it=s1.addItem(4,5,5,"Contact Info",n,n,"","",n,n,n,"../contact.html",n);
it=s1.addItem(2,3,3,"Board",n,n,"","",n,n,n,"../board.html",n);
it=s1.addItem(4,5,5,"Jobs",n,n,"","",n,n,n,"../jobs.html",n);
s0.pm.buildMenu();
}}
```

Special Techniques: Changing Content Dynamically

Display a series of images one at a time with “x of y”

The design here is to present one image with [Previous] and [Next] buttons and text “x of y” where x is the relative position of the image in the series and y is the total number of images. To do this the images are placed in a particular directory, the filenames are placed in an array, and JavaScript handles the Next and Previous events. CSS centers the “x of y” text between the two buttons. In this example the image is changed and the text that presents its position in the series is changed.

```
<head>
. . .
<style>
.photo     { border: 2px solid red;  }
#of        { margin-left: 100px; margin-right: 100px; }
button     { width: 90px; }
</style>

<script type="text/javascript">
var picDirectory = "Photos/";
var cntPics = 5;
var a = new Array(cntPics);
a[0] = "DSC00783.JPG";
a[1] = "DSC00784.JPG";
a[2] = "DSC00785.JPG";
a[3] = "DSC00786.JPG";
a[4] = "DSC00787.JPG";
var i = 0;
var from = 0;

function fullpicURI(j)
{
var URI = picDirectory + a[j];
return URI;
}
function prev()
{
if (i == 0) return;
i--;
var fn = fullpicURI(i);
document.images["pic"].src = fn;
replaceFrom();
}
function next()
{
if (i == (cntPics - 1)) return;
i++;
var fn = fullpicURI(i);
document.images["pic"].src = fn;
replaceFrom();
}
function writemofn()
{
document.write("1 of " + cntPics);
}
function replaceFrom()
{
document.getElementById("of").innerHTML = (i + 1) + " of " + cntPics;
}
</script>
</head>

<body>
<h1>Test of images</h1>
<div class="photo">
<button onclick="prev()">Previous</button>

<span id="of">
<script type="text/javascript">
writemofn();
</script>
</span>

<button onclick="next()">Next</button>
<p>
<img name=pic src="Photos/DSC00783.JPG">
</p>
```

```
</div>
```

Special Techniques: Miscellaneous

Using mailto as a URL

mailto can have one or more parts:

- "mailto:" which is like a protocol
- the recipient(s) email address
- one or more "headers," one for each part of the message (e.g., cc, bcc, subject, and body)

Multiple addresses are separated by a comma. The first header is separated from the recipient's email address by "?". Subsequent headers are separated from each other by a "&".

Link to send email with mailto

```
<a href="mailto:you@here.com">Email Us</a>
<a href="mailto:you@here.com,someoneelse@here.com">Email Us</a>
<a href="mailto:you@here.com?subject=Mail from Our Site">Email Us</a>
<a href="mailto:you@here.com?cc=someoneelse@theirsite.com">Email Us</a>
<a href="mailto:you@here.com?subject=testing&body=yeah">Email Us</a>
```

This approach causes the email message to be presented to the user for completion and sending, it is not sent invisibly. The user can change anything on the email before sending it manually.

It is possible to initiate the body of the email, but this is trickier. You must encode special characters like interword blanks (%20) and new lines (%0D). You can use the JavaScript function **encodeURIComponent()** to make this easier, but it must be used with the **document.write** method. For example:

```
<script language="JavaScript"><document.write("<a
    href=\"mailto:you@here.com?subject=" + encodeURIComponent("When, when is
    now? (if \"now\" is here)") + "\">mail me!</a>")</script>
```

Link to email while keeping email address obscure in HTML (to avoid spam)

```
<script type="text/javascript">
var td = "mailto:sjda@pge.com"
var tdx = "mai" + "lto:" + "sjda@p" + "ge.com"
</script>
<p><a href="" onclick="this.href=td">click to initiate email with
    JavaScript</a>. This works!</p>
<p><a href="" onclick="this.href=tdx">click to initiate email with
    JavaScript</a>. This works!</p
```

Using location.href to send email

```
<script type="text/javascript">
var td = "mailto:sjda@pge.com;mmo7@pge.com?subject=try this?body=blah blah
    blah";
parent.location.href=td;
```

Link to other page

```
<script type="text/javascript">
function jump()
{ location="http://www.xyz.com/" }
</script>
</head>
<body>
<form><input type="button" onclick="jump()" value="New location"></form>
```

Tie a behavior to an event

```
window.onblur = function() { window.close() }  // close window when loses focus
```

Display date like Tuesday, August 07, 2001

```
<BODY>
<SCRIPT LANGUAGE="Javascript"><!--
// ***********************************************
// AUTHOR: WWW.CGISCRIPT.NET, LLC
// URL: http://www.cgiscript.net
// Use the script, just leave this message intact.
// Download your FREE CGI/Perl Scripts today!
// ( http://www.cgiscript.net/scripts.htm )
// ***********************************************
// Get today's current date.
var now = new Date();
// Array list of days.
var days = new
    Array('Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday
    ');
// Array list of months.
var months = new
    Array('January','February','March','April','May','June','July','August','Se
    ptember','October','November','December');
// Calculate the number of the current day in the week.
var date = ((now.getDate()<10) ? "0" : "")+ now.getDate();
// Calculate four digit year.
function fourdigits(number)      {
    return (number < 1000) ? number + 1900 : number;
    }
// Join it all together
today =  days[now.getDay()] + ", " +
             months[now.getMonth()] + " " +
              date + ", " +
               (fourdigits(now.getYear())) ;
// Print out the data.
document.write("Today\'s date is " +today+ ".");
//--></SCRIPT>
```

Mouseover changes image and displays text

from www.dwellmag.com

```
function changeImages() {
  if (document.images) {
    for (var i=0; i<changeImages.arguments.length; i+=2) {
        document[changeImages.arguments[i]].src =
eval(changeImages.arguments[i+1] + ".src");
    }  } }
...
    document.write('<td height=23 width=144 bgcolor=#ffffff>
<a href="curr_01.html?noflash" onmouseover="changeImages(' + "'image1',
'image1on', 'infoimage', 'infoimage1'" + ')" onmouseout="changeImages(' +
"'image1', 'image1off', 'infoimage', 'infoimagedefault'" + ')"><img
src=imgs/button_mainnav.gif width=24 height=23 border=0 name=image1></a>
<a href="event_01.html?noflash" onmouseover="changeImages(' + "'image2',
'image2on', 'infoimage', 'infoimage2'" + ')" onmouseout="changeImages(' +
"'image2', 'image2off', 'infoimage', 'infoimagedefault'" + ')"><img
src=imgs/button_mainnav.gif width=24 height=23 border=0 name=image2></a>
<a href="diary1_01.html?noflash" onmouseover="changeImages(' + "'image3',
'image3on', 'infoimage', 'infoimage3'" + ')" onmouseout="changeImages(' +
"'image3', 'image3off', 'infoimage', 'infoimagedefault'" + ')"><img
src=imgs/button_mainnav.gif width=24 height=23 border=0 name=image3></a>
```

Function indent is used for index entries where each lower level is indented a little more.

It builds a line of HTML that starts with the indentation (in the form of non-breaking space) continues with the text of the entry with a link to the document and ends with a line break.

```
<head>
<script type="text/javascript">
<!--
/*
If there is no linked document, the a tag is ommitted.
The html that precedes the calls to this function must include the paragraph
tag.
The html that follows the calls to this function must include the end paragraph
tag
(so the style will work).
Three parameters:  1) number of spaces to indent; 2) text for entry; 3)
document to link to
(optional).
*/

function indent(w, t, l) {
    var c
    for (c=0; c<=w; c=c+1) { document.write(" ") }
    if (l == null) {document.write(t, "<br>")}
    else {document.write("<a href=", l, ">", t, "</a><br>")}
    }
//-->
</script>
```

code that calls the above function:

```
<p class=e>
<script type="text/javascript">
indent(0,  "Adjustments",  "adjustments.pdf" )
indent(5,  "Automatically created adjustments" )
```

Get IP address of client

```
<HEAD>
<SCRIPT>
var ip = '<!--#echo var="REMOTE_ADDR"-->'
function ipval() {
document.myform.ipaddr.value=ip;
}
</SCRIPT>
</HEAD>
<BODY  onLoad="ipval()">
```

Block copy of files from client

Dan, This web site somehow has protected its content from being copied. After you look at it, try right-clicking on an image - a "no you don't" message box appears. How do they do that?

http://members.teleweb.at/tuscaloosa-maine-coons/default.html

From: Niranjan Upadhayay <NiranjanUpadhayay@Mortgage.com>
This is what they have done!

```
<script language="javascript">
function prevent(e) {
        if (document.all) {
                if (event.button == 2) {
                        alert(message);
                        return false;
                }
```

```
        }
}
document.onmousedown=prevent()
</script>
```

This guys were not smart enough to prevent it from the menu
View-Source
Go ahead and use it from menu!!
Niranjan

Evaluate an expression or call a function after a period of time

```
setInterval(expression, milliseconds)   // evaluates expression every interval
    until cancelled by clearInterval
setInterval(function, milliseconds)     // executes function every interval
    until cancelled by clearInterval
setTimeout(expression, milliseconds)   // evaluates expression once after time
    period
setTimeout(function, milliseconds)   // executes function once after time
    period
```

Updating copyright notice

This copyright notice places the current year on the page so that copyright notices are always current. Once it's on your page ther is no need to update the script.

```
<script language = 'JavaScript'>
<!—
var today = new Date()
var year = today.getFullYear());
document.write('© '+year+' all rights reserved');
//-->
</script>
```

Date last modified

Use this script to show automatically when your page was last modified. This only works if the web server provides the date-time, otherwise the value is zero. The **lastModified** property is a date and time string like M/D/Y h:m:s.

```
<script language="javascript">
<!—
document.write('Last modified '+document.lastModified);
//--
</script>
```

Error handling

NOTE: untested
At beginning of first embedded script:

```
self.error = reportError;
. . .
function reportError(msg, line, url)
{
var w = window.open();
var d = w.document;
d.write('<HTML><HEAD><TITLE>Error Handler</TITLE></HEAD><BODY>');
d.write('<P>A JavaScript error has occurred.<>BR>Message: ' + msg + '<BR>Line
    number: ' + line');
d.write('</BODY</HTML>');
d.close();
}
```

Add this page to favorites

```
<span style='cursor:hand;text-
    decoration:underline;'onclick='window.external.AddFavorite(location.href,do
    cument.title);'>
Click here to add this page to your favorites</span>
```

Dynamically set link href to same-named file on different website

```
<script type="text\javascript" langualge'javascript">
function newURL()
{
//old URL is in format "http://pages.sbcglobal.net/sjdorey/"
var tp = location.pathname;   // format /sjdorey/...
var tl = tp.length;
var np = tp.slice(8);            // strip off /sjdorey
var nu = "http://www.susandoreydesigns.com" + np;
location.href = nu;
}
</script>
</head>

<body>
<p>ATTENTION.<br>
Effective immediately, this website has moved to <br>
<a href="javascript: newURL()">www.susandoreydesigns.com</a>.<br>
Click on the link to go there.</p>

</body>
```

Special Techniques: Change Text Size

Include on the page text and/or image to increase text size and to decrease text size. Each of these has a hyperlink with an onclick event that changes the text size. The following code uses a function resident in dw_sizerdx.js which I purloined from
http://www.newenglandancestors.org/common/js/dw_sizerdx.js
http://www.newenglandancestors.org/common/js/dw_cookies.js

```
<a href="" onclick="dw_fontSizerDX.adjust(0.1); return false" title="Incease
    Text Size">
<a href="" onclick="dw_fontSizerDX.adjust(-0.1); return false" title="Decrease
    Text Size">
. . .
<script type="text/javascript">
dw_fontSizerDX.setDefaults('em', 1, 0.7, 2.2, ['#secondary-content','#homepage-
    content'] );
dw_fontSizerDX.init();
</script>
```

Special Techniques: Viewing Series of Images on a Web Page

This was developed with file HTML/Test/test-images.html
Relevant design elements:

- page elements:
  controls: [Previous] x of y [Next]
  one image IMG tag
- JavaScript changes content of IMG tag when user clicks [Previous] or [Next]
- image file names are stored in an array, done with JavaScript
- the number of images in the array, the "y" in "x of y", is determined by JavaScript as the number of entries in the array

- first image in the array is shown when page opens
- CSS defines styling

```
.photo     { border: 2px solid red;  }
#of        { margin-left: 100px; margin-right: 100px; }
button     { width: 90px; }

<script type="text/javascript">   --- in HEAD section
var picDirectory = "Photos/";
var cntPics = 5;
var a = new Array(cntPics);
a[0] = "DSC00783.JPG";
a[1] = "DSC00784.JPG";
a[2] = "DSC00785.JPG";
a[3] = "DSC00786.JPG";
a[4] = "DSC00787.JPG";
var i = 0;
var from = 0;

function fullpicURI(j)
{
var URI = picDirectory + a[j];
return URI;
}

function prev()
{
if (i == 0) return;
i--;
var fn = fullpicURI(i);
document.images["pic"].src = fn;
replaceFrom();
}

function next()
{
if (i == (cntPics - 1)) return;
i++;
var fn = fullpicURI(i);
document.images["pic"].src = fn;
replaceFrom();
}

function writexofy()
{
document.write("1 of " + cntPics);
}

function replaceFrom()
{
document.getElementById("of").innerHTML = (i + 1) + " of " + cntPics;
}

</script>
</head>

<body>
<h1>Test of images</h1>
<div class="photo">
<button onclick="prev()">Previous</button>

<span id="of">
<script type="text/javascript">
writexofy();
</script>
</span>
```

```
<button onclick="next()">Next</button>
<p>
<img name=pic src="Photos/DSC00783.JPG">
</p>
</div>

</body>
</html>
```

Special Techiniques: Asynchronicity

AJAX is the label coined to refer to for Asynchronous JavaScript and XML. Asynchronous loading of content first became practical when Java applets were introduced in the first version of the Java language in 1995. These allow compiled client-side code to load data asynchronously from the web server after a web page is loaded. In 1996, Internet Explorer introduced the IFrame element to HTML, which also enabled asynchronous loading. In 1999, Microsoft created the XMLHTTP ActiveX control in Internet Explorer 5, which was later adopted by Mozilla, Safari, Opera and other browsers as the native **XMLHttpRequest** object. Microsoft has adopted the native XMLHttpRequest model as of Internet Explorer 7, though the ActiveX version is still supported.

The term "Ajax" was coined in 2005 by Jesse James Garrett. On April 5, 2006 the World Wide Web Consortium (W3C) released the first draft specification for the object in an attempt to create an official web standard http://www.w3.org/TR/XMLHttpRequest/

Special Techniques: Copy Text to Clipboard

It can be helpful to provide a tool to the user that makes it easy to copy text from a web page to the user's Clipboard after which they can easily paste it somewhere else. Most HTML elements have onClick and onDbleClick events. While I've seen this done with a button, like [Copy to Clipboard] it could also be done with an INPUT element (text box).

Note: There are many scripts for copying text from a Web page that work fine in Internet Explorer but not in Firefox, Netscape, or Opera. An IE-only script is very short and simple as it uses the proprietary **clipboardData** variable. For other browsers there is still no common solution with pure Javascript but there is one successful approach that uses a small Flash file embeded in inviibly current page.

I got the following script of the web, have not yet tried it. Mozilla-based browsers will ask the user for permission before allowing this script to access the clipboard.

Here's the HTML:

```
<textarea id='testText'>#COPYTOCLIPBOARD CODE#</textarea><br>
<button onclick='copyToClipboard(document.getElementById('testText').value);'>
```

Here's the JavaScript to make the copy:

```
function copyToClipboard(s)
{
    if( window.clipboardData && clipboardData.setData )
    {
        clipboardData.setData("Text", s);
    }
    else
    {
```

```
        // You have to sign the code to enable this or allow the action in
    about:config by changing
        user_pref("signed.applets.codebase_principal_support", true);

        netscape.security.PrivilegeManager.enablePrivilege('UniversalXPConnect'
    );
        var clip
    Components.classes['@mozilla.org/widget/clipboard;[[[[1]]]]'].createInstanc
    e(Components.interfaces.nsIClipboard);
        if (!clip) return;
        // create a transferable
        var trans =
    Components.classes['@mozilla.org/widget/transferable;[[[[1]]]]'].createInst
    ance(Components.interfaces.nsITransferable);
        if (!trans) return;
        // specify the data we wish to handle. Plaintext in this case.
        trans.addDataFlavor('text/unicode');
        // To get the data from the transferable we need two new objects
        var str = new Object();
        var len = new Object();
        var str = Components.classes["@mozilla.org/supports-
    string;[[[[1]]]]"].createInstance(Components.interfaces.nsISupportsString);
        var copytext=meintext;
        str.data=copytext;
        trans.setTransferData("text/unicode",str,copytext.length*[[[[2]]]]);
        var clipid=Components.interfaces.nsIClipboard;
        if (!clip) return false;
        clip.setData(trans,null,clipid.kGlobalClipboard);
    }
}
```