JavaScript Objects & Operators

Overview

Netscape began developing a scripting language called LiveScript in 1994. It was renamed JavaScript in December 1995, when Netscape and Sun Microsystems agreed to partner in its development. It provides a method of introducing interactivity to HTML documents that is widely used. It runs in the browser on a client computer that can access a local Java Virtual Machine (JVM). A client-side program can be executed in response to user events, such as mouse clicks. JavaScript may also be used to directly control objects, such as the browser status bar and the browser display window. It provides interactivity between Java applets and plug-ins. JavaScript can verify that information has been entered into a form, interpret the entered text, and alert the user with an appropriate message dialog.

JavaScript objects are defined here, along with their properties, methods, and event handlers. Event handlers are expressions that JavaScript executes when specific types of events occur. Independent functions not connected with any particular object are also listed, as well as operators.

JavaScript is a reduced and redefined version of the Java programming language. JavaScript is case sensitive, as is Java. A dot (.) is placed between an object and its properties. An identifier is a name used in JavaScript for a variable or a function. The first letter of an identifier may be an ASCII letter, an underscore (_), or a dollar sign ($). It may not be a number. Braces ({ }) are placed around function definitions, if-then constructs, and repeat loops. Square brackets ([ ]) are placed around optional items, such as parameters. Optional items are shown in italic. The syntax for defining a function is as follows:

```
function functionName([parameters]) {
[statement 1]
[statement 2]
}
```

Certain words are reserved in JavaScript; they may not be used as identifiers (function names, variable names, or loop labels). The following words are part of the language syntax and are reserved: break, case, continue, default, delete, do, else, export, for, function, if, import, in, new, return, switch, this, typeof, var, void, while, and with.

An additional list of words is reserved by the version of JavaScript standardized by the European Computer Machinery Association (ECMA), known as ECMA-262. The following words are reserved for future expansion in this version of the language: catch, class, const, debugger, enum, extends, finally, super, throw, and try.

A variable is a name associated with a data value. A variable contains an associated value. This statement assigns the value 5 to a variable named x:

```
x = 5
```

This script adds 2 to x and assigns the result to the new variable sum:

```
sum = x + 2
```

Variables in JavaScript are untyped, which means that they can hold data of any kind. Before a variable can be used, it must be declared with the var keyword. A variable may be declared and initialized in the same statement, as in the following lines:

```
var x = 5;
var message = "hello world"
```

A JavaScript expression is any letter, number, array, function, variable, or phrase that the interpreter can evaluate. The expression true is a *Boolean* literal, the phrase "hello world" is a *string* literal, and the number 5 is a *numeric* literal. All are different types of expressions.

A JavaScript program is a collection of statements. Examples of statements are if, else if, switch, while, for, break, return, and with. A complete list of statements and their syntax is included at the end of this appendix. Statements are followed by semicolons (;).

An object in JavaScript is a compound data type that can represent multiple data values with a single unit. It is a collection of properties, and each property has a name and a value. An object is created with the new operator, followed by the name of a constructor function that initializes the object. The following example creates an object and establishes its properties. Note that the double slash is used to distinguish comments from actual code.

```
// Create an object and store a reference to it.
var book = new Object();
// Set a property in the object.
book.title = "The Dictionary of Computing and Digital Media"
// Set another property in the object.
book.appendix.title = "JavaScript Objects and Operators"
```

A method is a JavaScript function that is invoked through an object. Functions are simply values stored in variables; these variables are properties of objects.

A method may be defined by the following procedure:

```
object.method = function;
```

The method may be invoked as follows, with the function surrounded by parentheses:

```
object.method(function);
```

Within the body of a method, the keyword "this" may be used to refer to the related object.

To make JavaScript respond dynamically to user input, event handlers are used. They may be defined as attributes of HTML objects. An event handler contains a piece of code to be executed when a particular event occurs, such as a mouse click. That code might send a message to the status bar or pop up an alert box containing text.

JavaScript can control most of the parts of a browser and responds to user actions such as form input and page navigation. All of the processing commands are written in a script that is embedded in an HTML document and carried out on the client side without reference back to a server.

A script is embedded in HTML with the <SCRIPT> element:

 © 2020 Hansen Media

```
<SCRIPT>..(text of script goes here)..</SCRIPT>
```

The text of a script is placed between the <SCRIPT> tag and its end element. Attributes are specified in the following way:

```
<SCRIPT LANGUAGE="JavaScript">
text of a script
</SCRIPT>
```

When specifying the language, it may be advisable to indicate which version of JavaScript is being used.

If the LANGUAGE attribute is not used, the SRC attribute may be used to identify a URL that will load the text of the script.

```
<SCRIPT LANGUAGE="language" SRC=url>
```

A user agent (browser) that is Java-enabled evaluates and stores the functions of each script it receives. The functions defined by the script are executed only when they are triggered by certain events within a page. Moving the cursor over an object or entering text into a text box may trigger the execution of a script.

The text of a script can be enclosed within comment elements to prevent browsers that are not capable of interpreting JavaScript from including it as text:

```
<SCRIPT LANGUAGE="JavaScript">
<!--"Hide script contents from incompatible browsers.
Complete text of JavaScript.
End hiding script contents."-->
</SCRIPT>
```

The following alphabetized list includes commonly used objects, properties, methods, and event handlers.

Objects

(*with Properties, Methods, and Event Handlers)*

anchor

The target of a hypertext link.

Properties

name: A string value containing the name of an anchor.

applet

Represents a Java applet in a web page.

Properties

name: A string value containing the NAME

 © 2020 Hansen Media

area

Represents a clickable area in an image map.

Properties

hash: An anchor name from a URL.

host: The host and domain name part of a URL.

hostname: The host, domain name, and port number of a URL.

href: The complete URL.

pathname: Just the path part of a URL, without host, domain, or port number.

port: The port number of a URL.

protocol: The protocol part of a URL, including the colon after it.

search: The query part of a URL, which follows a question mark.

target: The TARGET value of an AREA tag.

Methods

getSelection: Returns the current selection value as a string.

Event Handlers

onMouseOut: Specifies the script to be executed when a user moves the mouse outside the area defined.

onMouseOver: Specifies the script to be executed when a user moves the mouse over the area defined.

onDblClick: Specifies the script to be executed when a user double-clicks in the area.

array

A new array is created with the command

arrayName = new Array(*arrayLength*)

Properties

length: An integer defining the number of elements in an array.

prototype: A way to add properties to an Array object.

Methods

sort(): Sorts the elements of an array according to the function defined in parentheses, or alphabetically if no function is defined. (all browsers)

join(): Converts all elements of an array to concatenated strings.

 © 2020 Hansen Media

reverse(): Reverses the order of elements in an array.

concat(): Combines the elements of two arrays into a third.

pop(): Removes an element from the end of an array.

push(): Adds an element to the end of an array.

slice(): Returns a portion of an array, called a subarray.

button

Represents a push button in an HTML form. It is a reflection of an INPUT element with a TYPE attribute of "button".

Properties

Inherits properties, methods, and event handlers from Input and HTMLElement.

name: The name of the button element.

value: The value of the button element.

type: The TYPE attribute of an INPUT tag.

enabled: A Boolean value indicating a button's status.

form: Refers to the form object containing a button.

Methods

click(): Emulates the action of clicking on a button.

focus(): Gives a button focus.

Event Handlers

onClick: Specifies the script to be executed when the button is clicked.

onMouseDown: Specifies the script to be executed when a mouse button is pressed.

onMouseUp: Specifies the script to be executed when a mouse button is released.

onFocus: Specifies the script to be executed when a button is given focus.

checkbox

Makes a checkbox in an HTML form available in JavaScript.

Properties

checked: A Boolean value indicating the status of a checkbox element.

defaultChecked: A Boolean value indicating that a checkbox element reflects the CHECKED attribute.

name: The name of the checkbox element.

 © 2020 Hansen Media

value: The value of the checkbox element.

enabled: A Boolean value indicating whether the checkbox is enabled.

form: Refers to the form object containing the checkbox.

type: Indicates the TYPE attribute of the INPUT tag.

Methods

click(): Emulates the act of clicking the checkbox.

focus(): Gives the checkbox focus.

Event Handlers

onClick: Specifies the script to be executed when the checkbox is clicked.

onFocus: Specifies the script to be executed when focus is on the checkbox.

combo

Represents a combo field in JavaScript.

Properties

listCount: The number of elements in a list.

listIndex: The index of the selected elements in a list.

multiSelect: A Boolean value that indicates whether a combo field is in multiselect mode.

name: The name of a combo field.

value: The value of a combo field.

enabled: A Boolean value indicating whether the combo box is enabled.

form: Refers to the form object containing the combo box.

Methods

click(): Emulates a click on the combo field.

clear(): Clears the contents of the combo field.

focus(): Gives the combo field focus.

addItem(*index*): Adds an item to the combo field just before the item at *index*.

removeItem(*index*): Removes the item at *index* from the combo field.

Event Handlers

onClick: Specifies the script to be executed when the mouse clicks on a combo field.

onFocus: Specifies the script to be executed when focus is given to a combo field.

 © 2020 Hansen Media

Date

Provides a way to use dates and times in JavaScript. Syntax for creating instances is as follows:

```
newObjectName = new Date(dateInfo)
If no dateInfo is specified, the new object returns the current date and
time.
dateInfo may be used to specify a date and time in three ways:
1. "month, day, year, hours;minutes;seconds"
2. "year, month, day" (in integers)
3. "year, month, day, hours, minutes, seconds" (in integers)
```

Properties

prototype: Adds properties to a Date object.

Methods

getTime(): Returns the time of the current Date object in milliseconds, with 00:00:00:00 set at 12:00 am, January 1, 1970.

getDate(): Returns an integer for the day of the month of the current Date (1–31).

getMonth(): Returns an integer for the month in the current Date object (0–11, beginning with January).

getDay(): Returns an integer for the day of the week of the current Date (0–6, beginning with Sunday).

getHours(): Returns an integer for the hour in the current Date object (0–23).

getMinutes(): Returns an integer for the minutes in the current Date object (0–59).

getSeconds(): Returns an integer for the seconds in the current Date object (0–59).

getYear(): Returns a two-digit integer for the year in the current Date object (for example, 68 for 1968).

getTimezoneOffset(): Returns an integer in minutes representing the difference between local time and Greenwich mean time (GMT).

toGMTString(): Returns the value of the current Date object in GMT in the form Day, DD MON YYYY HH;MM;SS GMT.

toLocalString(): Returns the value of the current Date object in local time.

UTC *(yearValue, monthValue, dateValue, hoursValue, minutesValue, secondsValue)*: Returns an integer number of milliseconds since 12:00 am, January 1, 1970, (00:00:00) in GMT.

parse(*dateString*): Returns an integer number of milliseconds between 12:00 am, January 1, 1970, (00:00:00:00) and the date specified in *dateString.*

Other scripts that may be used to set date values, similar to those using the get command, are setDate*(dateValue)*, setHours*(hoursValue)*, setMinutes(*minutesValue*), setMonth*(month Value)*, setSeconds*(secondsValue)*, setTime*(timeValue)*, and setYear*(yearValue).*

 © 2020 Hansen Media

document

Represents the attributes of the currently displayed HTML document. Its properties are derived from the BODY element.

Properties

a1inkColor: The color of active links, defined as a hexadecimal triplet or a string.

anchors: An array of the anchor objects in the order they appear in an HTML document. anchors.length returns the number of anchors in a document.

bgColor: The background color of the document.

cookie: The current document's cookie values.

fgColor: The foreground color of the document.

forms: An array of form objects in the order in which the forms appear in an HTML file. forms.length returns the number of forms in the document.

lastModified: The last date on which the document was modified.

linkColor: The color of links, defined as a hexadecimal triplet or a string.

links: An array of the link objects in the order in which the hypertext links appear in an HTML document. links.length returns the number of links in a document.

location: The URL of the current document, expressed as document.URL. This property is deprecated.

referrer: The URL of the calling document.

title: The name of the current document.

vlinkColor: The color of followed links, defined as a hexadecimal triplet or a string.

applets: An array of the applet objects in the order in which they appear in the HTML document. applets.length returns the number of applets in the document.

embeds: An array of the plugin objects in the order in which they appear in the HTML document. embeds.length returns the number of plug-ins in the document.

images: An array of image objects in the order in which they appear in the HTML document. images.length returns the number of images in the document.

URL: The URL of the current document.

Methods

close(): Closes the current output stream.

open(*mimeType*): Allows write() method and writeln() method to write to the document window. *mimeType* (optional) specifies a document type.

 © 2020 Hansen Media

write(): Writes text and HTML to the specified document.

writeln(): Writes text and HTML to the specified document, and inserts a newline character.

clear(): Clears the document window.

captureEvents(): Specifies that a window with frames will capture all specified events.

releaseEvents(*eventType*): Forces the current window to release events which can then be passed to other objects.

routeEvent(*event*): Sends an event through the standard event hierarchy.

Event Handlers

onMouseDown: Specifies the script to be executed when a mouse button is pressed.

onMouseUp: Specifies the script to be executed when a mouse button is released.

onDblClick: Specifies the script to be executed when a mouse button is double clicked in a particular area. , N3, IE3, Macintosh)

onKeyUp: Specifies the script to be executed when a particular key is released.

onKeyDown: Specifies the script to be executed when a particular key is pressed.

onKeyPress: Specifies the script to be executed when a particular key is held down.

FileUpload

Represents a file upload element in an HTML form.

Properties

name: The name of the file upload element.

value: The field of a file upload element.

form

Represents a single HTML form in JavaScript. It collects input from a user to send to a server.

Properties

action: The URL to which the form data is submitted. This is the form's ACTION attribute.

elements: An array of objects for each form element in sequential order.

encoding: The MIME encoding of the form as specified in the ENCTYPE attribute. This is the form's ENCTYPE attribute.

method: A method in which form data is submitted to a server. This is the form's METHOD attribute.

target: The name of the window that displays responses to form submissions. This is the form's TARGET attribute.

 © 2020 Hansen Media

Methods

submit(): Submits form data to a server.

reset(): Resets a form.

Event Handlers

onSubmit: Specifies the script to be executed when a form is submitted. The value true must be returned to enable the form to be submitted. If a false value is returned, the form is not submitted.

onReset: Specifies the script to be executed when a form is reset.

frame

Represents a frame window in JavaScript. Frames within a browser are instances of the Window object and do not exist separately.

Properties

frames: An array of objects for each frame in a window, sequentially ordered as they appear in the HTML source code.

parent: The name of the window containing the frame set.

self: An alternate name for the current window.

top: An alternate name for the topmost window.

window: An alternate name for the current window.

onBlur: Represents the onBlur event handler for a frame. The event handler may be changed by assigning new values to this property.

onFocus: The onFocus event handler for the frame. The event handler may be changed by assigning new values to this property.

Methods

alert(*message*): Displays *message* in a dialog box.

clearTimeout(*name*): Cancels the timeout identified by *name*.

close(): Closes the window.

confirm(*message*): Displays *message* in a dialog box with OK and Cancel buttons. The user determines the true or false status.

open(*url, name, features*): Opens *url* in the *name* window, or creates a new window with that name. *features* is a list of features for the new window. The feature list may contain name-value pairs as shown below:

width=pixels

 © 2020 Hansen Media

height=pixels

status=[yes,no,1,0]

menubar=[yes,no,1,0]

scrollbars=[yes,no,1,0]

resizable=[yes,no,1,0]

toolbar=[yes,no,1,0]

location=[yes,no,1,0]

directories=[yes,no,1,0]

prompt(*message, response*): Displays *message* in a dialog box with the value of *response* in a text entry field. The user's response in the text entry field is returned.

setTimeout(*expression, time*): Evaluates *expression* after *time*.

blur(): Removes focus from the frame.

focus(): Gives focus to the frame.

clearInterval(*intervalID*): Cancels timeouts created with the setInterval method.

print(): Prints the contents of a frame or window.

setInterval(*expression, msec*): Evaluates *expression* after the duration specified by the *msec* parameter.

setInterval(*function, msec)*: Calls a function after the duration specified by the *msec* parameter.

Event Handlers

onFocus: Specifies the JavaScript code to execute when the frame is given focus.

onBlur: Specifies the script to be executed when focus is removed from the frame.

onMove: Specifies the script to be executed when the frame is moved.

onResize: Specifies the script to be executed when the frame is resized.

Function

A means of indicating the JavaScript code that should be compiled as a function. *functionName* is used as a variable with a reference to the function.

Properties

arguments: An integer that represents the number of arguments in the function.

prototype: A means of adding properties to the Function object.

 © 2020 Hansen Media

hidden

Represents a hidden field in an HTML form. This invisible form element allows arbitrary data to be sent to a server when the form is submitted.

Properties

name: The name of a hidden element in the form of a string.

value: The value of a hidden text element in the form of a string.

type: The TYPE property of the INPUT tag.

history

Allows JavaScript to access the browser's history list. The contents of the list are not reflected in JavaScript for security.

Properties

length: The number of items in the history list, expressed as an integer.

Methods

back(): Refers to the previous document in the list.

forward(): Refers to the next document in the list.

go(*location*): Goes to the document in the history list specified by *location,* which may be a string or integer value. As an integer it indicates the relative position of the document in the list.

HTMLElement superclass

This is the superclass of all classes that represent HTML elements. Its objects, listed below, are used in various contexts in client-side JavaScript.

```
document.all[i]
document.anchors[i]
document.forms[i]
document.forms[i].elements[j]
document.images[i]
document.links[i]
document.elementName
document.formName.elementName
```

Properties

It is advisable to determine whether these properties are implemented in the target browser:

all[]: An array of all elements contained by an element.

children[]: All elements that are direct children of an element.

className: The value of the CLASS attribute.

 © 2020 Hansen Media

document: Refers to the containing document object.

id: The value of the ID attribute.

innerHTML: The HTML text within an element.

innerText: The plain text within an element.

lang: The value of the LANG attribute.

offsetHeight: The height of an element.

offsetLeft: The x-coordinate of an element.

offsetParent: An element that contains offsetLeft and offsetTop.

offsetTop: The y-coordinate of an element.

offsetWidth: The width of an element.

outerHTML: The HTML text of an element, with start and end tags.

outerText: The plain text in a document, with start and end tags.

parentElement: An element that is the direct parent of this one.

sourceIndex: The index of an element in Document.all[].

style: The inline Cascading Style Sheet (CSS) attributes for an element.

tagName: The name of the HTML tag that created an element.

title: The value of the TITLE attribute.

Methods

It is advisable to determine whether these methods are implemented in the target browser:

contains(): Determines whether an element contains the one specified in ().

getAttribute(): Returns the value of a named attribute.

insertAdjacentHTML(): Inserts HTML text into the document nearest an element.

insertAdjacentText(): Inserts plain text into the document nearest an element.

removeAttribute(): Deletes an attribute and its value from an element.

scrollIntoView(): Scrolls the document so an element is visible in the window.

setAttribute(): Sets the value of an attribute of an element.

handleEvent(): Passes an event object to the appropriate event handler.

Event Handlers

onClick: Specifies the script to be executed when a mouse button is clicked.

 © 2020 Hansen Media

onDblClick: Specifies the script to be executed when a mouse button is double clicked on an element.

onHelp: Specifies the script to be executed when a user requests help.

onKeyDown: Specifies the script to be executed when a particular key is pressed.

onKeyPress: Specifies the script to be executed when a particular key is held down.

onKeyUp: Specifies the script to be executed when a particular key is released.

onMouseDown: Specifies the script to be executed when a mouse button is pressed.

onMouseMove: Specifies the script to be executed when a mouse button is moved.

onMouseOut: Specifies the script to be executed when the mouse is moved off of an element.

onMouseOver: Specifies the script to be executed when the mouse is moved over an element.

onMouseUp: Specifies the script to be executed when a mouse button is released

Image

Represents an image in an HTML document.

Properties

border: An integer representing the width of the border of an image in pixels.

complete: A Boolean value that indicates whether an image is done loading.

height: An integer representing the height of an image in pixels.

hspace: The HSPACE attribute of the IMG tag, expressed as an integer.

lowsrc: The URL of a low resolution version of an image.

name: The name of an Image object.

prototype: A means of adding properties to an Image object.

src: The URL of an image.

vspace: The VSPACE attribute of the IMG tag, expressed as an integer.

width: An integer representing the width of an image in pixels.

Event Handlers

onAbort: Specifies the script to be executed when an attempt to load an image is aborted.

onError: Specifies the script to be executed when an error occurs during image loading. If this handler is set to *null*, the error message is suppressed.

onLoad: Specifies the script to be executed when an image is completely loaded.

onKeyDown: Specifies the script to be executed when a specific key is pressed.

 © 2020 Hansen Media

onKeyUp: Specifies the script to be executed when a specific key is released.

onKeyPress: Specifies the script to be executed when a specific key is held down.

Layer

A means of embedding layers of content within a page. Layers allow dynamically positioned elements to be included in a dynamic HTML (DHTML) document.

Properties

above: Refers to a layer on top of the current layer.

background: Specifies the tiled background image of a layer.

below: Refers to a layer below the current layer.

bgColor: Sets the background color of a layer.

clip(*left, right, top, bottom, width, height*): Specifies the visible boundaries of the layer.

height: Specifies the height of a layer, expressed in pixels by an integer or by a percentage of the current layer.

ID: Names a layer for reference by other scripts. (Same as name.)

left: Specifies the horizontal position of the top-left corner of the layer, along with the top property.

page[*X, Y*]: Specifies the horizontal (X) or vertical (Y) position of the top-left corner of the layer relative to the overall enclosing document.

parentLayer: Specifies the layer object that contains the current layer.

siblingAbove: Specifies the layer object directly on top of the present one.

siblingBelow: Specifies the layer object directly under the present one.

SRC: Specifies the source URL of a layer's contents.

top: Specifies the y-coordinate of the top-left corner of the layer, along with the left property.

visibility: Specifies the visibility of the layer. The three variables are *show* (visible), *hidden* visible), and *inherit* (properties inherited from the parent layer).

width: Specifies the width of a layer or a boundary inside of which contents remain confined or wrap.

z-index: Specifies, with an integer, the stacking order (z-order) of a layer. Sets a layer's position within the overall rotational order if there are multiple layers.

Methods

captureEvents(): Specifies that a window with frames will capture all specified events.

 © 2020 Hansen Media

handleEvent(): Dispatches an event to the appropriate handler.

load(*source, width*): Replaces the source with HTML (or JavaScript) from the file specified in *source*. Also passes a width value in pixels.

moveAbove(*layer*): Places the current layer above the *layer* identified in the stack.

moveBelow(*layer*): Places the current layer below the *layer* identified in the stack.

moveBy(*x, y*): Changes the position of the current layer by the pixel values specified.

moveTo(*x, y*): Changes the position of the current layer (within the containing layer) to the coordinates specified in pixels.

moveToAbsolute(*x, y*): Changes the position of the current layer (within a page) to the coordinates specified in pixels.

resizeBy(*width, height*): Changes the size of the current layer by the values specified in pixels.

resizeTo(*width, height*): Changes the size of the current layer to the values specified for width and height in pixels.

releaseEvents(*eventType*): Specifies that the current window should release events on input of *eventType.*

Event Handlers

onBlur: Specifies the script to be executed when the layer no longer has focus.

onFocus: Specifies the script to be executed when the layer gains focus.

onLoad: Specifies the script to be executed when the layer is loaded.

onMouseOut: Specifies the script to be executed when the mouse moves off the current layer. Properties that may be used include type, target, layer[n], page[n], and screen[n].

onMouseOver: Specifies the script to be executed when the mouse enters the current layer. Properties that may be used include type, target, layer[n], page[n], and screen[n].

link

Represents a hypertext link in the body of an HTML document.

Properties

hash: The anchor name in a URL referenced (follows the hash mark #).

host: The host name and port number from a URL referenced.

hostname: The numerical IP address, or domain name, from the URL referenced.

href: The entire URL referenced. (The HREF attribute of the A element.)

pathname: The path portion of a URL referenced.

 © 2020 Hansen Media

port: The port number from the URL referenced (ftp=21; WWW=80; Usenet news=119).

protocol: The protocol from the URL referenced, including the colon but not the slashes.

search: All information passed to a GET CGI-BIN call.

target: The name of a window or frame specified in the TARGET attribute.

text: The plain text between the <A> and </A> tags that created the object.

Event Handlers

onMouseUp: Specifies the script to be executed when the mouse button is released.

onClick: Specifies the script to be executed when a user clicks on a link.

onDblClick: Specifies the script to be executed when the mouse button is double clicked in a particular area.

moveMouse: Specifies the script to be executed when the mouse moves over a link.

onKeyUp: Specifies the script to be executed when a particular key is released.

onKeyDown: Specifies the script to be executed when a particular key is pressed.

onKeyPress: Specifies the script to be executed when a particular key is held down.

onMouseDown: Specifies the script to be executed when the mouse button is pressed.

onMouseOut: Specifies the script to be executed when the mouse cursor moves out of an object. Properties that may be used include type, target, layer[n], page[n], and screen[n].

onMouseOver: Specifies the script to be executed when the mouse moves over a hypertext link. Properties that may be used include type, target, layer[n], page[n], and screen[n].

location

Reflects information about the current URL.

Properties

hash: The anchor name in the URL.

host: The host name and port number from the URL.

hostname: The numerical IP address, or domain name from the URL.

href: The entire URL.

pathname: The path portion of the URL.

port: The port number from the URL.

protocol: The protocol from the URL with the colon but not the slashes.

search: All information passed to a GET CGI-BIN call.

 © 2020 Hansen Media

Methods

reload(): Reloads the current document.

replace(*url*): Loads the *url* specified over the current entry in the history list; after this it is impossible to navigate back to the previous URL by using the Back button.

Math

Provides properties and methods for mathematical calculations.

Properties

E: Euler's constant (approximately 2.718282), which is used as the base for natural logarithms.

LN2: The natural logarithm of two (approximately 0.693147).

LN10: The natural logarithm of 10 (approximately 2.302585).

LOG10E: The base 10 logarithm of e (approximately 0.434294).

LOG2E: The base 2 logarithm of e (approximately 1.442695).

PI: The value of pi for calculating the circumference and area of a circle (approximately 3.14159).

SQRT1_2: The square root of one-half (approximately 0.7071).

SQRT2: The square root of two (approximately 1.4142).

Methods

abs(*number*): Returns the absolute value of *number* without consideration of its sign. abs(8) is equal to abs(-8).

acos(*number*): Returns the arc cosine of *number* in radians.

asin(*number*): Returns the arc sine of *number* in radians.

atan*(number)*: Returns the arc tangent of *number* in radians.

atan2*(number1, number2)*: Returns the angle of the polar coordinate corresponding to the Cartesian coordinate *(number1,number2).*

ceil*(number)*: Returns the smallest integer equal to or greater than *number*.

cos*(number)*: Returns the cosine of *number*.

exp*(number)*: Returns the value of e to the power of *number.*

floor(*number*): Returns the greatest integer equal to or less than *number*.

log*(number)*: Returns the natural logarithm (base e) of *number.*

max*(number1, number2)*: Returns the greater of *number1* and *number2.*

 © 2020 Hansen Media

min*(number1, number2)*: Returns the smaller of *number1* and *number2.*

pow*(number1, number2)*: Returns the value of *number1* to the power of *number2.*

random(): Returns a random number between zero and one.

round(*number*): Returns the closest integer to *number*, rounded off.

sin(*number*): Returns the sine of *number,* which is an angle expressed in radians.

sqrt*(number)*: Returns the square root of *number.*

tan*(number)*: Returns the tangent of *number,* where *number* represents an angle in radians.

mimeType

Represents a MIME type supported by the client browser.

Properties

description: A description of the MIME type.

enabledPlugin: Refers to the plugin object that supports the MIME type.

suffixes: A list of file suffixes for the MIME type, separated by commas.

type: A string that indicates the MIME type itself, as in "video/mpeg".

option

Creates entries in a select list with the syntax shown here:

```
optionName = new Option(optionText, optionValue, defaultSelected, selected)
where
selectName.options[index] = optionName
```

Properties

defaultSelected: Specifies with a Boolean value whether an option is selected by default.

index: Specifies with an integer an option's index in the select list.

prototype: A means of adding properties to an Option object.

selected: Indicates with a Boolean value whether an option is currently selected.

text: The text displayed for an option.

value: The value sent to the server when a form is submitted.

password

Represents a password text field from an HTML form in JavaScript.

Properties

defaultValue: The default value of the password element.

 © 2020 Hansen Media

name: The name of the password element.

value: The value of the password element.

form: Refers to the form object containing the password field.

enabled: Indicates with a Boolean value if a password field is enabled.

Methods

focus(): Gives focus to the password field.

blur(): Removes focus from the password field.

select(): Selects the text in the password field.

Event Handlers

onBlur: Specifies the script to be executed when focus is removed from the password field.

onFocus: Specifies the script to be executed when focus is given to the password field.

plugin

Reflects a plug-in supported by the browser.

Properties

description: The description provided by the plug-in itself.

filename: The file name of the plug-in as it appears on the client's machine.

length: The number of MIME types supported by the plug-in, described as array elements of the plug-in object.

name: The name of the plug-in.

radio

Represents a set of graphical radio buttons in an HTML form in JavaScript. Individual radio buttons in the set are accessed numerically in order, beginning with zero. Example: myButton[0], myButton[1], myButton[2] for three radio buttons.

Properties

checked: Indicates with a Boolean value whether a particular button is checked. Buttons may be selected or deselected with this property.

name: The name of a set of radio buttons.

value: The value of a specific radio button in a set.

defaultChecked: Indicates with a Boolean value whether a particular button reflects the CHECKED attribute.

length: Indicates with an integer the number of radio buttons in a set.

 © 2020 Hansen Media

enabled: Indicates with a Boolean value whether a particular button is enabled.

form: Refers to the form object containing the radio button.

Methods

click(): The act of clicking a radio button.

focus(): Gives focus to a radio button.

Event Handlers

onClick: Specifies the script to be executed when a radio button is clicked.

onFocus: Specifies the script to be executed when a radio button is given focus.

RegExp

Represents a regular expression for pattern matching with strings. Properties identify a series of values that may be accessed in a search. Its arguments are pattern and attributes, constructed as follows: new RegExp(*pattern, attributes*)

Static Properties

$1,$2,$3,. . $9: Identifies the last nine substrings enclosed in parentheses that were matched.

input: The string to which a regular expression is compared.

lastMatch: Identifies the last matched characters.

lastParen: Identifies the last matched string that appeared in parentheses.

leftContext: Identifies the string just before the most recently matched regular expression.

multiline [*true, false*]: Determines whether a search continues beyond line breaks.

rightContext: Identifies the part of a string that continues beyond the most recently matched regular expression.

Flags

i: Ignore upper- and lowercase in search (optional).

g: Perform a global match in the search for a regular expression (optional).

gi: Perform a global match, ignoring case (optional).

Instance Properties

source: The pattern's text in a read-only version.

ignoreCase [*true, false*]: Sets the i (ignore case) flag value.

global [*true, false*]: Sets the g (global) flag value.

 © 2020 Hansen Media

lastIndex: Indicates with an integer the index position at which to begin the next matching procedure.

Methods

compile: Defines a new pattern and attributes for a RegExp object, usually invoked at script startup.

exec(str): Executes the search for a regular expression matching the specified string (str).

test(str): Searches for a regular expression and the string specified (str).

reset

Represents a reset button from an HTML form in JavaScript.

Properties

name: The name of the reset element.

value: The value of the reset element.

enabled: Indicates with a Boolean value whether the reset button is enabled.

form: Refers to the form object containing the reset button.

Methods

click(): The act of clicking the reset button.

focus(): Specifies the script to be executed when the reset button is given focus.

Event Handlers

onClick: Specifies the script to be executed when the reset button is clicked.

onFocus: Specifies the script to be executed when the reset button is given focus.

Screen

Specifies characteristics of the current screen.

Properties

availHeight: Specifies the height of the screen in pixels, after deducting display constraints defined by the OS.

availTop: Specifies the first available vertical pixel, after deducting display constraints defined by the OS.

availWidth: Specifies the width of the current screen in pixels, after deducting display constraints defined by the OS.

colorDepth: Specifies the bit depth, or number of colors for the current screen.

height: Specifies the height of the current screen in pixels.

 © 2020 Hansen Media

pixelDepth: Specifies the resolution in bits per pixel of the current screen.

width: Specifies the width of the current screen in pixels.

select

Represents a selection list from an HTML form in JavaScript.

Properties

length: Identifies with an integer the number of choices in a selection list.

name: The name of a selection list.

selectedIndex: The index of the currently selected option in a selection list.

options: An array that represents the choices in a selection list.

Properties of options include the following:

defaultSelected: Indicates with a Boolean value whether a particular option reflects the SELECTED attribute.

index: The index of an option.

length: The number of options in a selection list.

name: The name of the selection list.

selected: Indicates with a Boolean value whether a particular button is selected. Options may be selected or deselected with this property.

selectedIndex: Indicates with an integer the index of the currently selected option.

text: The text displayed in a selection list for a particular option.

value: The value for a specified option.

Methods

blur(): Takes focus away from a selection list.

focus(): Gives focus to a selection list.

Event Handlers

onBlur: The script code to be executed when focus is removed from a selection list.

onChange: The script to be executed when the selected option in a list changes.

onFocus: The script to be executed when focus is given to a selection list.

String

Provides properties and methods for working with variables and string literals.

Properties

 © 2020 Hansen Media

length: The number of characters in the string, expressed as an integer.

prototype: Means of adding properties to a String object.

Methods

anchor(*name*): Returns a copy of the *name* string in the <A NAME=> format.

big(): Returns a copy of the specified string in the <BIG> format.

blink(): Returns a copy of the specified string in the <BLINK> format.

bold(): Returns a copy of the specified string in the <B> format.

charAt(*index*): Returns the character at the location in a string specified by *index.*

charCodeAt(*index*): Returns the encoded value of a character at the *index* position in a string.

fixed(): Returns a copy of the specified string in the <TT> format.

indexOf(*substring, start*): Returns the index of the first occurrence of *substring,* starting the search at *start*. By default, the search starts at the beginning of the string.

italics(): Returns a copy of the specified string in the <I> format.

lastIndexOf(*substring, start*): Returns the index of the first occurrence of *substring*, beginning at *start* and searching in reverse order. By default, the search starts at the end of the string.

link(*href*): Returns a copy of the *href* string in the <A HREF=> format.

match(*regexp*): Matches the regular expression specified, expressed as a variable or as a literal.

replace(*regexp, replacement*): Locates and replaces a regular expression with *replacement*.

search(*regexp*): Locates a regular expression and matches it to the specified string.

slice(*start, end*): Extracts a substring and derives a new string.

small(): Returns a copy of the specified string in the <SMALL> format.

strike(): Returns a copy of the specified string in the <STRIKE> format.

sub(): Returns a copy of the specified string in the <SUB> format.

sup(): Returns a copy of the specified string in the <SUP> format.

toLowerCase(): Returns a copy of the specified string with lowercase characters.

toUpperCase(): Returns a copy of the specified string with uppercase characters.

fontColor(*color*): Returns a copy of the *color* string in the <FONT COLOR=> format.

fontSize(*size*): Returns a copy of the *size* string in the <FONT SIZE=> format.

split(*delimiter*): Divides a string at every occurrence of *delimiter.*

concat(*string2*): Combines two strings to make a new third one.

 © 2020 Hansen Media

substring(*from, to*): Sets the number of characters within a string. Use *from* to specify the beginning point for the extraction process, ending at *to*.

submit

Represents a submit button from an HTML form in JavaScript.

Properties

name: The name of the submit button element.

value: The value of the submit button element.

enabled: Indicates with a Boolean value whether the submit button is enabled.

form: Represents the form object containing a submit button.

type: The TYPE attribute of the INPUT tag.

Methods

click(): Simulates clicking the submit button.

focus(): Gives focus to the submit button.

Event Handlers

onClick: Specifies the script to be executed when the submit button is clicked.

onFocus: Specifies the script to be executed when the submit button is given focus.

text

Represents a text field from an HTML form in JavaScript.

Properties

defaultValue: The default value of the text element (VALUE attribute).

name: The name of the text field element.

value: The value of the text element.

enabled: Indicates with a Boolean value whether a text field is enabled.

form: Represents the form object containing a text field.

type: The TYPE attribute of the INPUT tag.

Methods

focus(): Gives focus to the text field.

blur(): Removes focus from the text field.

select(): Selects text in the text field.

 © 2020 Hansen Media

Event Handlers

onBlur: Specifies the script to be executed when focus is removed from a text field.

onChange: Specifies the script to be executed when text field content is changed.

onFocus: Specifies the script to be executed when the text field receives focus.

onSelect: Specifies the script to be executed when any text in the field is selected.

textarea

Represents a multiline text field from an HTML form in JavaScript.

Properties

defaultValue: The default value of the textarea element (VALUE attribute).

name: The name of the textarea element.

value: The value of the textarea element.

enabled: Indicates with a Boolean value whether a textarea field is enabled.

form: Represents the form object containing a textarea field.

type: The TYPE attribute of the textarea INPUT tag.

Methods

focus(): Gives focus to the textarea field.

blur(): Removes focus from the textarea field.

select(): Simulates selecting text in the textarea field.

Event Handlers

onBlur: Specifies the script to be executed when focus is removed from a textarea field.

onChange: Specifies the script to be executed when textarea field content is changed.

onFocus: Specifies the script to be executed when the textarea field receives focus.

onSelect: Specifies the script to be executed when any text in the textarea field is selected.

onKeyUp: Specifies the script to be executed when a specific key is released.

onKeyPress: Specifies the script to be executed when a specific key is held down by the user.

onKeyDown: Specifies the script to be executed when a specific key is pressed.

window

The parent object for document, location, and history objects. This is the top-level object for an array of frames or a window. The Window is the "global object," and all expressions are evaluated in the context of the current window object.

 © 2020 Hansen Media

Properties

defaultStatus: The default value displayed in the status bar.

frames: An array of objects representing each frame in the window, ordered as they appear in the HTML source code.

name: The name of a window or frame.

parent: The name of the window containing a frame set.

self: Another name for the current window.

status: Displays a message, consisting of assigned values, in the status bar.

statusbar=[*true,false,1,0*]: Controls whether the status bar in the target window is visible.

toolbar=[*true,false,l,0*]: Controls whether the toolbar in the target window is visible.

top: Another name for the top-most window.

window: Another name for the current window.

length: Indicates with an integer the number of frames in a parent window.

opener: The window object that contains the open() method used to open the current window.

innerHeight(): Specifies the vertical size in pixels of the content area.

innerWidth(): Specifies the horizontal size in pixels of the content area.

pageXOffset: Specifies the current X position in pixels of the viewable window area.

pageYOffset: Specifies the current Y position in pixels of the viewable window area.

scrollbars [*visible=true,false*]: Controls whether scroll bars in the current window are visible.

Methods

alert(*message*): Causes a specified *message* to appear in a dialog box.

captureEvents():Specifies that a window with frames will capture all specified events. Used with enableExternalCapture.

clearTimeout(*name*): Cancels the timeout specified by *name*.

clearTimeout(*timeoutId*): Cancels the deferred execution of the *setTimeout* identified.

confirm(*question*): Displays a *question* in a dialog box, to which a user may respond "OK" or "Cancel." Returns true or false, depending on the button selected by the user.

length(): Specifies the number of frames in a window.

prompt(*message, response)*: Shows *message* in a dialog box, along with a text entry field containing the default value of *response.* Returns a string with data keyed into a text entry field.

 © 2020 Hansen Media

setTimeout(*code, delay*): Defers execution of *expression* until after *delay*, expressed in milliseconds.

stop(): Ends the current download. Simulates pressing the Stop button.

close(): Closes a specified window.

open(*url, name, features*): Opens the *url* specified in a window called *name*. A new window is created, if one does not exist by that name. The string argument "*features*" specifies parameters for the window that is opened. The following names and values may be specified. There are no spaces between values, but each is followed by a comma:

alwaysLowered=[*yes,no,1,2*] (background window)

alwaysRaised=[*yes,no,1,2*] (top-level window)

directories=[*yes,no,1,0*]

height=pixels

location=[*yes,no,1,0*]

menubar=[*yes,no,1,0*]

resizable=[*yes,no,1,0*]

scrollbars=[*yes,no,1,0*]

status=[*yes,no,1,0*]

toolbar=[*yes,no,1,0*]

width=pixels

dependent[*yes,no,1,2*](on parent window)

hotkeys=[*yes,no,1,2*]

innerWidth=pixels

innerHeight=pixels

outerWidth=pixels

outerHeight=pixels

screenX=pixels

screenY=pixels

z-lock=[*yes,no,1,2*]

blur(): Removes focus from the window, in most cases sending it to the background.

focus(): Gives keyboard focus to the top window, in most cases bringing it to the front.

 © 2020 Hansen Media

scrollTo(*x, y*): Scrolls the current window to the specified *x*- and *y*-coordinates.

back(): Calls the previous URL visited, similar to activating the Back browser button.

clearInterval(*intervalID*): Eliminates timeouts created with the setInterval method.

disableExternalCapture(): Prevents a current window with frames from capturing events that occur in pages loaded from another location.

enableExternalCapture(): Allows a current window with frames to capture events that occur in pages loaded from another location.

event(): Describes the most recent event to occur in a window.

find([*string*], [*true, false*], [*true, false*]): Finds the *string* specified in the target window. The first Boolean true/false parameter specifies whether the search is case sensitive. The second true/false parameter specifies whether the search should be performed in reverse order.

forward(): Moves to the next URL listed, similar to activating the Forward browser button.

home(): Directs the browser to the user's home page.

moveBy(*x, y*): Repositions a window to the right (*x*) or down (*y*), according to the specified values.

movoTo(*x, y*): Positions the upper-left corner of a window at the specified location.

print(): Sends the contents of a frame or window to a printer. Simulates pressing the Print button.

releaseEvents(*eventType*): Rather than capturing events in the current window, allows them to be passed on to other objects.

resizeBy(*horizontal, vertical*): Beginning from the lower-right corner, resizes the window.

resizeTo(*outerWidth, outerHeight*): Applies the specified *outerWidth* and *outerHeight* properties to the window.

routeEvent(*event*): Sends an event through the normal hierarchy.

scrollBy(*horizontal, vertical*): Causes the viewing area of the current window to be scrolled by the number of pixels specified by the *horizontal* or *vertical* values.

scrollTo(*x, y*): Beginning at the upper-left corner of the current window, scrolls it to the position specified in the *x*- and *y*-coordinates.

setInterval(*expression, msec*): Evaluates the specified *expression* after the period identified by the *msec* parameter.

setInterval(*function, msec, args*): Continuously calls a *function* after the period specified by the *msec* parameter.

 © 2020 Hansen Media

Event Handlers

onLoad: The script to be executed when the window or frame is done loading.

onUnload: The script to be executed when a user exits the document in a frame or window.

onBlur: The script to be executed when focus is removed from the window.

onError: The script to be executed when a JavaScript error is returned while a document is loading. A JavaScript error message will not be displayed to the user if this event handler is set to null.

onFocus: The script to be executed when the window receives focus.

onResize: The script to be executed when the window is resized by a user.

onMove: The script to be executed when the window is repositioned by a user.

onDragDrop: The script to be executed when an object is dropped into the window by a user.

Symbolic Operators

Arithmetic Operators

+ Adds the operands on the left and on the right. (When used with strings rather than integers, it combines the operands into one string.)

- When used as a binary operator, subtracts the right operand from the left operand.

- Performs unary negation when placed immediately before an operand; converts an operand from positive to negative, and vice versa.

* Multiplies two operands.

/ Divides the left operand by the right operand.

% Returns the remainder of the left operand divided by the right operand.

++ Increases an operand by one (may be applied before or after an operand).

— Decreases an operand by one (may be applied before or after an operand).

Assignment Operators

These operators work with both numbers and strings. For example, with numbers the operator += performs addition and assignment. For strings it performs concatenation and assignment.

= Assigns the value of the right operand to an expression. It causes the variable, element, or property a to refer to the value b.

Example: a = b is equivalent to b

 © 2020 Hansen Media

+= Adds the left and right operands, then assigns the total to the left operand. (When used with strings rather than integers, concatenation of the two operands is performed.)

Example: a += b is equivalent to a = a + b

-= Subtracts the right operand from the left operand, then assigns the result to the left operand.

Example: a -= b is equivalent to a = a - b

*= Multiplies two operands, then assigns the result to the left operand.

Example: a *= b is equivalent to a = a * b

/= Divides the left operand by the right operand, then assigns the value to the left operand.

Example: a /= b is equivalent to a = a / b

%= Divides the left operand by the right operand, then assigns the remainder to the left operand.

Example: a %= b is equivalent to a = a % b

Bitwise Operators

These operators treat their operands as binary numbers, convert them to integers with 32 bits, and return a JavaScript numerical value.

AND(&) Pairs corresponding bits, and returns one for each pair of ones. Returns zero for any other combination.

OR(|) Pairs corresponding bits, and returns one for each pair if either of the bits is a one. Returns zero if both bits are zero.

XOR(^) Pairs corresponding bits, and returns one for each pair where only one bit is a one. Returns zero for any other combination.

<< Shifts bits to the left by the number of bits indicated by the right operand. Bits shifted to the left are discarded, and zeros are shifted over from the right.

>> Shifts bits to the right by the number of bits indicated by the right operand. Bits shifted to the right are discarded. The bit furthest to the left is duplicated and shifted over from the left.

>>> Shifts bits to the right by the number of bits indicated by the right operand. Bits shifted to the right are discarded. Zeros are shifted over from the left.

Logical Operators

These operators perform Boolean algebra. They are particularly useful in complex comparisons that involve more than one variable, combined with if, while, and for statements.

&& (logical AND) Returns true when both operands are true, otherwise returns false.

|| (logical OR) Returns true if either operand is true. Returns false only when both operands are false.

 © 2020 Hansen Media

! (logical NOT) Returns true if the operand is false, and returns false if the operand is true. This unary operator precedes the operand.

Equality and Identity Operators

== Returns true if operands are equal (numbers, strings, or Boolean values) or if they refer to the same object (function, array, or object). This operator compares for equality without comparing type only in Internet Explorer 4.0.

!= Returns true if operands are not equal or they refer to different objects.

=== Returns true only if operands are identical without comparing type.

!== Returns true only if operands are not equal without comparing type.

Comparison Operators

> Returns true if the left operand is greater than the right operand.

< Returns true if the left operand is less than the right operand.

>= Returns true if the left operand is greater than or equal to the right operand.

<= Returns true if the left operand is less than or equal to the right operand.

Conditional Operators

typeof: Returns the type of its single operand. Types that are returned are object, string, number, boolean, function, or undefined.

void: Discards its operand value or returns an undefined value. This unary operator may precede an expression with any value.

(? :): This is a ternary operator, which uses three operands. Returns value1 if the condition is true, otherwise returns value2.

Example: condition ? value1 : value2

Independent Functions

escape(*character*): Returns a string with *character* ASCII-encoded for transmission across networks to all computer platforms. NonASCII characters are encoded in the format %*xx*, where *xx* is two hexadecimal digits representing the character in ISO 8859 (Latin-1) coding.

eval(*code*): Returns the result of evaluating an arithmetic expression, or executes the JavaScript statements represented by *code* and returns the value.

isNaN(*value*): Evaluates *value* to determine whether it is NaN (not a number), and returns a Boolean value.

parseFloat*(string)*: Returns the value of *string* after converting it to a floating-point number. This function returns NaN (zero in MS Windows) when it encounters the first character that cannot be converted to a number.

 © 2020 Hansen Media

parseInt(*string, radix*): Returns the value of *string* after converting it to an integer in base *radix*. This function returns NaN (zero in MS Windows) when it encounters the first character that cannot be converted to an integer.

unescape(*string*): Returns a character based on the ASCII encoding contained in *string.* It decodes a previously escaped string.

Table of Operators

In this table, the first three columns on the left show the operators, types of operands, and operations performed. The associativity is shown in the next column (LTR = left to right, RTL = right to left). The order of precedence that each operator takes is shown in the last column on the right.

Operator	Type of Operand	Operation	Associati-vity	Prece-dence
(,)	Any	Multiple Evaluation	LTR	1
(=)	Variable, Any	Assignment	RTL	2
(+=, -=, *=, /=, %=, <<=, >>=, >>>=, &=, ^=, =)	Variable, Any	Assignment with Operation	RTL	2
(? :)	Boolean, Any	Conditional (ternary)	RTL	3
(||)	Boolean	Logical OR	LTR	4
(&&)	Boolean	Logical AND	LTR	5
(|)	Integer	Bitwise OR	LTR	6
(^)	Integer	Bitwise XOR	LTR	7
(&)	Integer	Bitwise AND	LTR	8
(==, ! =)	Any	Checks for Equality/Identity	LTR	9
(<, <=, >, >=)	Numbers/Strings	Relational	LTR	10
(<<, >>, >>>)	Integer	Shift	LTR	11
(+, -)	IntegerString (+ only)	Addition/Subtraction Concatenation	LTR	12
(*, /, %)	Integer	Multiply/Divide/Remainder	LTR	13
(++,--, -)	Integer	Increment/Decrement/Negate (unary)	RTL	14
(~)	Integer	Bitwise Complement (unary)		

 © 2020 Hansen Media

(!)	Boolean	Logical Complement (unary)		
new	Constructor Call	Create New Object (unary)		
type of	Any	Return Data Type (unary)		
void	Any	Return Undefined Value (unary)		
([ ]) (( )) (.)	Array, Integer	Array Index	LTR	15
	Functions, Arguments	Function Calls		
	Object, Property	Property Access		

Statements

break: Exits from a “for” or a “while” loop and passes program control to the statement following the loop.

case: Labels a statement within a switch statement.

comment: Allows a programmer to comment on code. Comments are enclosed in a /* (start), */ (end) structure.

continue: Continues iteration of a “while” or a “for” loop after the execution of the statements within the loop.

default: Labels the default statement within a switch statement.

do/while: Sets up a continuous loop that executes statements and code until the condition evaluates to false.

export(*expression*): In secure, signed scripts, it allows all of the properties, functions, and variables to be exported to another script. Used in conjunction with an import statement, it makes the *expression* variable or function accessible in other windows or execution contexts.

for(*[expression]; [condition]; [increment];)*: Opens a for loop. Arguments initialize a variable (*expression*), create a condition to test for (*condition*), and specify incrementation.

for/in: Introduces a variable to all the properties of an object and executes a block of code for each one, looping through the properties.

function [name](*[arg1,... argn]*): Declares a function that event handlers or other processes may access.

if/else: Conditionally executes code. Example: If (*expression*) *statement1* [else *statement2*]. Nested statements and functions will be executed depending on whether the condition described by *expression* is true or false.

 © 2020 Hansen Media

import (*variables*): In secure, signed scripts this statement allows all properties, functions, and variables to be imported from another script. Used in conjunction with the export statement, the named variables are imported.

label(*identifier: statement*): Gives *statement* the name *identifier*. Creates a pointer to code located elsewhere in the script. The script is redirected to the labeled statement.

return [*expression*]: Specifies an expression or value to be returned by a given function.

switch (*expression*): Evaluates an expression and attempts to match it to a case or default pattern or label. If the expression matches the case, statements associated with the label are executed.

this: A statement that refers to a specific object. For example, this.width = w passes a specified value to an object.

var [*name*]: Declares and initializes the variable name.

while (*expression*): A statement that constructs and begins a while loop. As long as a condition is true, the specified code is executed.

with (*object*): A statement that sets the value for the default object and adds it to the front of a chain.

 © 2020 Hansen Media