ActiveXObject – Object

Activates and returns a reference to an **Automation** object.

Syntax

```
new ActiveXObject(application.type[, server])
```

`application` is the name of the application that supplies the object.
`type` represents the type of object that is going to be created.
`server` represents the name of the server where the object is created. This argument is optional.

Anchor – Object

Properties **See also**

A place in a document that is the target of a hypertext link. In an HTML document, all **<A> elements** that have the attribute **NAME** require an entry. These entries are kept in an array. For example, if a document has three elements: **<A NAME=..>**, these will be represented as follows: **`document.anchor[0]`**, **`document.anchor[1]`** and **`document.anchor[2]`**.

Syntax

To define the **Anchor** element in a document, use the standard **HTML** syntax.

```
<A NAME="anchorName"> text </A>
```

To use the anchor array:

```
document.anchors[index]
document.anchors.length
```

`index` is an integer representing element **<A>** .
To obtain the number of elements **<A>** in a document, use the property length (Array.

Example

```
<SCRIPT LANGUAGE="JavaScript">
window2=open("file.htm","Subscribe","scrollbars=yes,width=250, height=400")
function GoToWindow(num) {
   if (window2.document.anchors.length > num)
      window2.location.hash=num
   else
      alert("Error")
}
</SCRIPT>
<FORM>
Click on the button to  go to URL <P>
<INPUT TYPE="button" VALUE="Subscribe" NAME="subscribe_button"
onClick="GoToWindow(this.value)">
</FORM>
</BODY>
</HTML>
```

Applet – Object

See also

This object is used to insert a Java Applet in an HTML document .

The applets array

Reference can be made to different applets that are indexed, by using an applets array. This table contains an entry for every **Applet** object (**<APPLET>** element) according to the source order. For example, if a documents contains three applets, they can be referenced in the following manner:

```
document.applets[0]
document.applets[1]
document.applets[2]
```

Syntax

To define a Java Applet in an HTML document:

```
<APPLET>
 ..
</APPLET>
```

To use an applet object

```
AppletName.PropertyName
document.applets[index].propertyName
```

appletName is the value of attribute **NAME** of an **Applet** object.
index is an integer representing an applet in a document or a character string containing the name of an **Applet** object.

To use the applets table

```
document.applets[index]
document.applets.length
```

The length property is used to obtain the number of Applets in a document.

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.write('This document contains',document.applets.length,' applets')
</SCRIPT>
```

Area – Object

Events **See also**

This object is used to define hotpots in an image map.
To reference an **Area** object, it is necessary to use a links array. **Area** objects are a type of Link object.

Syntax:

To define an **Area** object, use standard HTML syntax in order to handle onMouseOut or onMouseOver events:

```
<MAP NAME="ImageMapName">
   <AREA>
</MAP>
```

To use the Area object properties

```
areaName.propertyName
document.links[index].propertyName
```

areaName is the value of attribute **NAME** of an **Area** object.
index is an integer representing a region in a document or a character string containing the name of an **Area** object.

Array – Object

Properties Methods See also

A built-in object that in Javascript language, used to create arrays and handle them. The initial length of the array can be specified.

The following code creates an array with six elements:

```
 = new Array(6)
```

It is also possible to assign an initial value to certain elements in the array:

```
ATable = new Array(20)
ATable[0] = "contentA"
ATable[1] = "contentB"
ATable[2] = "contentC"
```

Syntax

To create an **Array** object:

```
arrayObjectName = new Array()
arrayObjectName = new Array(TableLength)
```

To use the **Array** object:

```
arrayObjectName.propertyName
arrayObjectName.methodName(parameters)
```

`arrayObjectName` is the name of a new object or a property of an existing object.
`TableLength` is the table's initial length.

Example

```
Line=new Array (8)
Line[7]=0
```

Boolean – Object

Properties **Methods** **See also**

This is used to create a new Boolean value.

Syntax

```
new boolean(Value)
```

Note

`Value` is an optional argument. If it's absent, **NaN**, **null**, **0**, **false** or an empty string, then the value is **False**. The initial value is **True**.

Boolean is a wrapper.

Button – Object

Properties Methods Events See also

This object is the button found on HTML forms.

Syntax

To define a button in an HTML document:

```
<INPUT TYPE="button" NAME="ButtonName" ..>
```

To use the properties and methods of the **Button** object

```
buttonName.propertyName
buttonName.methodName(parameters)
formName.elements[index].propertyName
formName.elements[index].methodName(parameters)
```

`formName` is the value of attribute **NAME** of the **<FORM>** element.
`index` is an integer representing the **Button** object in a form.

Example

```
<SCRIPT LANGUAGE="JavaScript">
<FORM>
<INPUT TYPE="button" VALUE="Subscribe" NAME="subscribe_button"
onClick="window.open('suscribe.html', 'Subscribe',
'scrollbars=yes,status=yes,width=500,height=300')">
</FORM>
</SCRIPT>
```

Checkbox – Object

Properties Methods Events See also

This object represents a checkbox in an HTML form.

Syntax

To define a checkbox in an HTML document:

```
<INPUT TYPE="checkbox" NAME="CheckboxName"..>
```

To use the properties and methods of the **Checkbox** object:

```
checkboxName.propertyName
checkboxName.methodName(parameters)
formName.elements[index].propertyName
formName.elements[index].methodName(parameters)
```

`formName` is the value of attribute **NAME** of the **<FORM>** element.
`index` is an integer representing a **Checkbox** object in a form.

Example

```
<SCRIPT LANGUAGE="JavaScript">
<FORM><INPUT TYPE="checkbox" VALUE="Help" NAME="Help"
onClick="alert ('pay attention to quotes in the alert notice') ">
</FORM>
</SCRIPT>
```

Date – Object

Properties **Methods** **See also**

This object allows the user to work with the date and time.

Syntax

To create the **Date** object:

1. `dateObjectName = new Date()`
2. `dateObjectName = new Date("day of the month, year, hour:minutes:seconds")`
3. `dateObjectName = new Date(year, month, day)`
4. `dateObjectName = new Date(year, month, day, hour, minutes, seconds)`

`dateObjectName` is the name of a new object or the property of an existing object.
`year`, `month`, `day`, `hour`, `minutes` and `seconds` are character string values for points 1 and 2, while points 3 and 4 are integer values.

To use the **Date** object:

```
dateObjectName.methodName(parameters)
```

Exceptions

Methods parse and UTC are methods that you must use as follows:

```
Date.parse(parameters)
Date.UTC(parameters)
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
DisplayTime function () {
Today = new  Date  ()
document.write (Today.getHours() + "h" +   Today.getMinutes () )
}
DisplayHour ()
</SCRIPT>
```

Document – Object

Properties Methods Events See also

This object contains information on the current document and provides the methods for displaying HTML output to the user. You can load a new document by using the Location object.

Syntax

To define the **document** object, use standard HTML syntax:

```
<BODY>
  ..
</BODY>
```

To use the properties and methods of the **Document** object:

```
document.propertyName
document.methodName(parameters)
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.write ("The background color is ", document.bgColor,
"<BR>The text (foreground) color is ",document.fgColor)
</SCRIPT>
```

Enumerator – Object

Methods

This object is used to enumerate the elements in a collection.

Syntax

```
new Enumerator(collection)
```

The `collection` argument represents any collection object `collection` object.

Notes

Members of a collection are not directly accessible. In this case, indexes are not useful. It suffixes to point on the first element or the following element in a collection.

The `Enumerator` object allows accessing each of the members of a collection.

Error – Object

Properties **See also**

Contains information on errors.

Syntax

```
var newErrorObj = new Error()
var newErrorObj = new Error(number)
var newErrorObj = new Error(number, description)
```

The syntax of the **Error** object includes the following elements:

Element	Description
number	Number related to the error. Zero if absent.
description	Short description of error. Empty string if absent.

Notes

An **Error** object is created when a mistake occurs in order to inform of its occurrence. The object has two intrinsic properties that contain a description of the error (description property) and the error number (number property).

Event – Object

Properties

The **`event`** object contains properties describing a JavaScript event and it is used as an argument for an **event** when the latter takes place.

For example, when a onMouseDown event takes place, the **`event`** object contains the event type (in this case, onMouseDown), positions x and y of the mouse pointer at the time the event takes place, a number indicating the mouse button used and a field indicating the modifying keys (Ctrl, Alt, etc.) that were released when the event took place.

FileUpload – Object

Properties Methods Events See also

This object allows the user to specify the access path and file name as input in a HTML form field. This file is sent together with the other information on the form.

Syntax

To define the **FileUpload** object, use standard HTML syntax:

```
<INPUT TYPE="file" NAME="fileUploadName">
```

To use the properties of the **FileUpload** object:

```
fileUploadName.propertyName
```

`fileUploadName` is the value of the **NAME** attribute of a **FileUpload** object.

Example

```
<FORM NAME="fileup">
File to send: <INPUT TYPE="file" NAME="test">
<INPUT TYPE="button" VALUE="name"
onClick="alert('name: ' +document.fileup.test.name)">
<INPUT TYPE="button" VALUE="value"
onClick="alert('value: ' +document.fileup.test.value)"><BR>
</FORM>
```

Form – Object

Properties Methods Events See also

Lets users input text and make choices from Form elements. You can also use a form to post data to a server. Each form in a document represents a different object. You can refer to each element in a form using the element's name (attribute **NAME** of the **<FORM>** element), using the **elements array** object.

The forms array

Reference can be made to a form that is indexed, by using a forms table. This array has an entry for every form object (**<FORM>** element) in the document in source order. For example, if a document contains three forms, these are represented in the following manner:

```
document.forms[0]
document.forms[1]
document.forms[2]
```

Syntax

To define the **Form** object, use standard HTML syntax:

```
<FORM>
   ..
</FORM>
```

To use the properties and methods of the Form object

```
formName.propertyName
formName.methodName(parameters)
forms[index].propertyName
forms[index].methodName(parameters)
```

formName is the value of attribute **NAME** of the **<FORM>** element.
index is an integer representing the **Form** object

To use a forms array:

```
document.forms[index]
document.forms.length
```

index is an integer representing a form in a document.
To obtain the number of forms in a document, use the length. property.

Example

```
<SCRIPT LANGUAGE="JavaScript">
function testInput (){
if (document.form1.fourchars.value.length != 4)
{
     alert("Enter four characters " +
document.form1.fourchars.value + " is not valid")
 }
}
</SCRIPT>

<FORM NAME=form1 onSubmit="testInput()" >
<INPUT TYPE="text" VALUE="1" SIZE=4  NAME=fourchars >

</FORM>
```

Frame – Object

See also

This object can display multiple, independent frames on a single window, each with its own distinct URL. Each frame is a window object.

A frame can be referenced by using the statement **`parent.frameName`** or **`parent.frames[index]`**.For example, if the third frame has the attribute **`NAME="FrameOne`**, then it can be referenced using the statement **`parent.FrameOne`** or **`parent.frames[2]`**.

To reference the current fame, use the self or window property. You can also use the top and parent properties instead of frame names.

The frames array:

You can reference a **Frame** object in your code using the frames array. This array contains an entry for each child frame (**<FRAME>** elements) of an **<FRAMESET>** element, and the latter, in the source order.
For example, if a window has three child frames, these are represented as follows:

```
parent.frames[0]
parent.frames[1]
parent.frames[2]
```

Elements in the frames array are read-only.
For example, the statement **`fourcharsframes[0]="frame1"`** produces no effect.

Syntax

To define the **Frame** object, use standard HTML syntax:

```
<FRAMESET>
...<FRAME SRC="URL" NAME="FrameName">
</FRAMESET>
```

To use the properties of the Frame object

```
[fourchars] frameName.propertyName
[fourchars]frames[index].propertyName
window.propertyName
self.propertyName
parent.propertyName
```

`windowReference` is variable of the **`windowVar`** type of a window definition (refer to the window object, or to one of the top or parent properties).
`frameName` is the value of attribute **NAME** of the **<FRAME>** element.
`index` is an integer representing the **Frame** object

To use frames array

```
[FrameReference.]frames[index]
[FrameReference.]frames.length
[windowReference.]frames[index]
[windowReference.]frames.length
```

`FrameRef` is the correct way to reference a frame.
`windowReference` is a variable of the **`windowVar`** type of a window definition (refer to the window object) or to one of the top or parent properties.
`index` is an integer representing a frame in a parent window.
To obtain the names of child frames of a window or a frame, use the length property.

Example

```
<SCRIPT LANGUAGE="JavaScript">document.write ("Number of active frames: " +
window.frames.length)</SCRIPT>
```

Function – Object

Properties **Methods** **See also**

This object indicates a character string of JavaScript code to be compiled as a function. The **Function** objects are evaluated each time they are used.

Syntax

```
TargetFunction = new Function ([arg1, arg2, ... argn], FunctionBody)
```

TargetFunction is the name of a variable or a property of an existing object or an object followed by an event, such as **window.onError**.
arg1, **arg2**, ... **argn** are character string type of arguments.
functionBody is a character string that indicates the JavaScript code to be complied as the function body.

Global – Object

Properties Methods See also

`Global` is one of the intrinsic values of JScript5 (objects that are part of the standard JScript language). It's objective is to cover all the global methods in a unique object.

Syntax

There is no syntax. Methods are called directly.

Note

The `Global` object is never used directly. It cannot be created with the new operator. Its methods and properties are available once the script engine starts.

Hidden – Object

Properties See also

This object is used to associate a text type value to a form field. The field and its value are not visible to the user and therefore cannot be modified. However, it is possible to change the **Hidden** object value using JavaScript.

Syntax

To define the **hidden** object, use standard HTML syntax:

```
<INPUT TYPE="hidden" NAME="hiddenName"..>
```

To use the properties of the "hidden" object:

```
hiddenName.propertyName
frameName.elements[index].propertyName
```

hiddenName is the value of attribute **NAME** of a **Hidden** object.
formName is the value of attribute **NAME** of the Form object or an element in a forms array.
index is an integer representing the **Hidden** object in a form.

Example

```
<FORM NAME=form3>
<INPUT TYPE="hidden" NAME="any" VALUE="Hidden">
<INPUT TYPE="text" NAME="surprise">
<INPUT TYPE="button" Value="See hidden field" NAME="button"
onClick="document.form3.surprise.value=document.form3.any.value">
</FORM>
```

History – Object

Properties Methods See also

This object memorizes the URL's addresses the user visits. The URL's are saved as a list and are usually accessible from the browser's button **Go**.

Syntax

To use the **History** object:

```
history.propertyName
history.methodName(parameters)
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
function Reload () {
history.go(0)
}
</SCRIPT>
<INPUT TYPE="image" BORDER=0 SRC="gif/reload.gif"  onClick="Reload()"
```

Image – Object

Properties **Methods** **Events** **See also**

This object represents an image in an HTML form.

The Image() constructor:

The objective of using an image objet with the **Image()** constructor is to load an image from the network and decode it before there is need to display it. Then, when the image is to be displayed, the image's src property can be set to the same value it will use later. In this way the image will be displayed from the cache memory which is capable of displaying faster than from network itself. You can also use this method to create an animation or to display one of the images stored immediately after the entries in a form.

Example

```
myImage = new Image()
myImage.src = "anImage.gif"
...
document.images[0].src = myImage.src
```

The images array

You can reference an image using an images array. This array has an entry for every **Image** object (**<IMG>** element) in the document in source order. Images created with the Image() constructor are not included in the images array. For example, if the document contains three images, these are referred to in the following manner:

```
document.images[0]
document.images[1]
document.images[2]
```

Elements in the images array are read-only.

Syntax

To define the an image object, use standard HTML syntax:

```
<IMG SRC="URL" ..>
```

To create the Image object:

```
ImageName new Image([length,height])
```

imageName is the name of the new object or the property of an existing object.
width is the image width in pixels.
height is the image height in pixels.

To se the properties of an Image object

```
imageName.propertyName
document.images[index].propertyName
formName.elements[index].propertyName
```

imageName is the value of attribute **NAME** of an **Image** object.
formName is the value of attribute **NAME** of the Form object or an element in a forms array.
index, when used in an images array is an integer or a character string representing an **Image** object. When used in an elements array, **index** is an integer representing an **Image** object in a form.

To define an event handler for an Image object created with the Image() constructor:

```
imageName.onabort = handlerFunction
imageName.onerror = handlerFunction
imageName.onload = handlerFunction
```

imageName is the name of the new object or the property of an existing object.

The **handlerFunction** can be the **null** keyword, the name of a function, a variable, a property containing the **null** keyword or a valid reference to a function.

To use the images array

```
document.images[index]
document.images.length
```

index is an integer representing an image in a document or a character string containing the name of an **Image** object.
To obtain the number of images in a document, use the length property.

Example

```
<SCRIPT LANGUAGE="JavaScript">
delay = 100
Number = 0
myImage = new Array()
for(i = 0; i < 10; i++) {
   myImage[i] = new Image()
   myImage[i].src = "gif/" + i + ".gif"
}
function animate() {
   document.animation.src = myImage[Number].src
   Number++
   if(Number>= 10) {
      Number = 0
   }
}
</SCRIPT>
<IMG NAME="animation" SRC="gif/0.gif""
   onLoad="setTimeout('animate()', delay)">
```

Java – Object

See also

A top-level, built-in JavaScript object used to access any Java class of a *java.** package using JavaScript. You can access this object without using the constructor or without calling a method.

JavaArray – Object

Properties **Methods**

An instance of a Java array that can be accessed with JavaScript. Every reference to a Java array must use the **JavaArray** object.

Created by:

Whatever the method used, Java outputs a list.

JavaClass – Object

See also

A JavaScript reference to a Java class.

A **JavaClass** object is a reference to a class of a (*Java package*), such as *netscape.javascript.JSObject*. Moreover, **JavaPackage** is a reference to a (*Java package*), such as *netscape.javascript*.

In **JavaScript**, the **JavaPackage** and **JavaClass** hierarchy is a consequence of the Java package and class hierarchy.

Properties

The properties of the **JavaClass** object are defined by the Java class static fields.

Methods

The methods of the **JavaClass** object are defined by the Java class static methods.

JavaObject – Object

See also

This object is used as a container for a Java object accessed using JavaScript code. Whatever the method, Java returns a type of object. You can also create a **JavaObject** object using a constructor of Java objects by employing the **Packages** keyword:

```
new Packages.JavaClass(parametersList)
```

where **Javaclass** is the full name of the Java class.

Properties

It inherits (*public data members*) of the Java class. It also inherits (*public data members*) of every super-class, including properties.

Methods

It inherits the Java class public methods. The **JavaObject** object also inherits the method issued of a **java.lang.Object** and any other super-class.

JavaPackage – Object

See also

A Javascript reference to a (*Java package*).

Properties

The properties of the **JavaPackage** object are defined by the JavaClass objects and any other **JavaPakage** object contained therein.

Layer – Object

Properties **Methods** **Events**

The **Layer** object corresponds to the **<LAYER>** element of an HTML document and provides different means to handle said element.

Created by:

The HTML **<LAYER>** and **<ILAYER>** elements or cascading Style Sheets (CSS).

Link – Object

Properties **Methods** **Events** **See also**

This object is a hyperlink in a document. When the user activates the link, the reference is loaded in the (**window target**).

Each **Link** object is a Location object and possesses the same properties as the latter.

If the **Link** object is also an Anchor object, then it has an entry in the bookmarks array and the links array.

The links array

You can reference a **Link** object in your document using the links array. This array contains an entry for every **Link** object (**<A href="...">** element) found in the document. For example, if a document has three **Link** objects, these will be represented as follows:

```
document.links[0]
document.links[1]
document.links[2]
```

Syntax

To define the **Link** object, use standard HTML syntax:

```
<A HREF="URL"..> hypertext link </A>
```

Your can also define the **Link** object using the link method.

To use the Link method:

```
document.links[index].propertyName
```

index is an integer representing the **Link** object.

To use the links array:

```
document.links[index]
document.links.length
```

index is an integer representing a link in a document.

To obtain the number of links in a document, use the length property.

Example

```
<A HREF=annonce.htm onMouseOver ="alert('click')">announcement.htlm</A>
```

Location – Object

Properties Methods See also

This object contains information on the current URL. It represents the complete URL. Each property of the **Location** object represents a different portion of the URL.

The following description of a URL shows the relationships between the location properties:

```
protocol//host:port/pathname#hash?search
```

protocol (represents the beginning of the URL, up to and including the first colon)
host (represents the host and domain name, or IP address, of a network host)
port (represents the communications port that the server uses)
pathname (represents the path portion of the URL and the filename indicated in the URL)
search (represents any query information)
hash (represents an anchor name in the URL)

Do not confuse the **Location** object with the location property of the document object. The value of property location (**document.location**) cannot be changed, however, you can change the value of the **Location** value (**window.location.propertyName**).

Syntax

To use the **Location** object:

```
[windowReference.]location[.propertyName]
[windowReference.]location.methodName(parameters)
```

windowReference is variable of the **windowVar** type of a window definition (refer to the window object, or to one of the top or parent properties).

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.write("<FONT COLOR=Green>The protocol is: </FONT><BR>" +
window.location.protocol)
</SCRIPT>
```

Math – Object

Properties **Methods** **See also**

A built-in object in JavaScript language. It has properties and methods for mathematical constants and functions. For example, the Math object's PI property has the value of pi. For example, the **Math** object's PI property has the value of pi.

Syntax

```
Math.Property
Math.Method
```

Example

```
<FORM><INPUT TYPE="button" VALUE="root"
onClick="form.result.value = Math.sqrt(form.number.value)"><INPUT
TYPE="text" NAME="number" SIZE=6><INPUT TYPE="text" NAME="result"></FORM>
```

MimeType – Object

Properties **See also**

The **MimeType** (*Multipart Internet Mail Extension*) are built-in objects in JavaScript. Users cannot create **MimeType** objects.

They are only accessible using navigator or Plugin objects:

```
browser.mimeTypes[index]
```

As it is, the MimeType object becomes a property of navigator or Plugin objects.

For example the following table indicates the values required to display JPEG images:

Expression	Value
`navigator.mimeTypes["image/jpeg"].type`	image/jpeg
`navigator.mimeTypes["image/jpeg"].description`	JPEG Image
`navigator.mimeTypes["image/jpeg"].suffixes`	jpeg, jpg, jpe, jfif, pjpeg, pjp
`navigator.mimeTypes["image/jpeg"].enabledPlugins`	null

navigator – Object

Properties **Methods**

This object contains information about the version of the user's Web browser.

Syntax

To use the **navigator** object:

```
navigator.propertyName
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.write (  "<FONT COLOR=green>Your browser is</FONT><BR>"
+ navigator.appName + navigator.appVersion )
</SCRIPT>
```

netscape – Object

See also

A top-level, built-in JavaScript object used to access any Java class of a *netscape.** package using JavaScript. You can access this object without using the constructor or without calling a method.

Number – Object

Properties **Methods** **See also**

This object is used to create object using numeric constants.

Syntax

```
new Number(value)
```

`value` is a numeric value.

Object – Object

Properties **Methods** **See also**

The **Object** is the primitive type of JavaScript object. All JavaScript objects are descendants of **Object** and possess the methods defined by this super-object.

Created by:

The `Object constructor:`

```
new Object()
```

Option – Object

Properties

This object is an option of the Select object. This option is created using the **`Option()`** constructor. For more information, refer to the Select object.

Example

```
<FORM Name=form4>
<SELECT NAME="choice"
onChange="form4.results.value+=form4.choice.options[form4.choice.selectedIn
dex].value"
<OPTION VALUE="Go "> Go
<OPTION VALUE="Return "> Return
<OPTION VALUE="Stay ">Stay
</SELECT>
<INPUT TYPE=TEXT SIZE=15 NAME="result">
</FORM>
```

Packages – Object

Properties

Top-level, built-in object used to access Java classes using JavaScript code. You can access this object without using the constructor or without calling a method.

Description

The **Packages** object is used to access methods and public fields of a Java class using JavaScript. The **java**, **netscape**, and **sun** properties represent the *java.*, netscape.*,* and *sun.** packages.

Use the usual Java syntax to access classes, methods and fields of these packages.

Password – Object

Properties Methods Events See also

This object is a text box for passwords found on HTML forms. The characters entered in this box are hidden by asterisks.

Syntax

To define the **Password** object, use standard HTML syntax:

```
<INPUT TYPE="password" NAME="Password"..>
```

To use the properties and methods of the **Password** object:

```
passwordName.propertyName
passwordName.methodName(parameters)
formName.elements[index].propertyName
formName.elements[index].methodName(parameters)
```

passwordName is the value of attribute **NAME** of a P**assword** object.
formName is the value of attribute **NAME** of the Form object or an element in a forms array.
index is an integer representing the **Password** object in a form.

Example

```
<FORM>
<INPUT TYPE="password" NAME="password" onBlur="alert('Your password is ' +
this.form.password.value)">
</FORM>
```

Plugin – Object

Properties **See also**

This object generates output using a *plug-in* application (an embedded object).

The embeds array:

You can reference embedded modules (*plug-ins*) found in HTML code using the embeds array. This array contains an entry for each **Plugin** object (**<EMBED>** element) found in the document. For example, if the document contains three embedded modules, these are referred to in the following manner:

```
document.embeds[0],
document.embeds[1],
document.embeds[2].
```

The embeds array is read-only.

The plugins array:

This array is used to determined if the user has installed a particular embedded object (*plug-in*).

Syntax

To define the **Plugin** object, use standard HTML syntax:

```
<EMBED SRC=source  NAME=AppletName..>
..
</EMBED>
```

To use the embeds array:

```
document.embeds[index]
document.embeds.length
```

index is an integer representing an embedded module in a document or a character string containing the name of a **Plugin** object.
To obtain the number of embedded modules in a document, use the length property.

Example

```
<SCRIPT LANGUAGE="JavaScript">
for (i=0; i < navigator.plugins.length; i++) {
               document.writeln(navigator.plugins[i].name,
     navigator.plugins[i].description,
                 navigator.plugins[i].length)
          }
</SCRIPT>
```

Radio – Object

Properties **Methods** **Events** **See also**

This object represents a selection box (radio button type) found on HTML forms. These buttons allow the user to select an option in a list.

Syntax

To define the **Radio** object, use standard HTML syntax:

```
<INPUT TYPE="radio" NAME="radioName"..> text to be displayed
```

To use the properties and methods of the Radio object

```
radioName[index1].propertyName
radioName[index1].methodName(parameters)
formName.elements[index2].propertyName
formName.elements[index2].methodName(parameters)
```

radioName is the value of attribute **NAME of a Radio** object.
index is an integer representing a radio button in a **Radio** object.
formName is the value of attribute **NAME** of the Form object or an element in a forms array.
index is an integer representing the radio button in a form. The elements array has an entry for each radio button in a **Radio** object.

Example

```
<FORM NAME=form5>
<INPUT TYPE="text" Name="ch1" SIZE=4>
<INPUT TYPE="radio" NAME="ch2" value=one
onClick="this.form.ch1.value=this.value"><INPUT TYPE="radio" NAME="ch2"
value=two onClick="this.form.ch1.value=this.value">
</FORM>
```

RegExp – Object

Properties **Methods** **See also**

The **RegExp** object contains a pattern of a regular expression. These properties and methods allow using regular expressions to find and replace matches in strings.

Created by:

A text literal form or the **RegExp** construction function.

The **literal form** is used as follows:

/pattern/flags

The **construction function** is used as follows:

```
new RegExp("pattern"[, "flags"])
```

Parameters

`pattern`
Text or regular expression.
`flags`
If specified, **`flags`** can have one of the following values:
`g`: global match
`i`: ignore case
`gi`: a combination of the first two parameters

Table containing some of the special characters that can be used with expressions:

Character	Description
`\`	For characters that are usually treated literally, indicates that the next character is special and not to be interpreted. **`/b/`** corresponds to character «b«. With the backslash , **`/\b/`**, the character is considered special. On the contrary, **`/a*/`** equal to 0 or more like a **`. /a\*/`** corresponds to «a*».
`^`	Matches the beginning of a line or an entry. **`/^A/`** does not correspond to 'A' found in «ancestor A,» but to «Ancestor A».
`$`	Matches the end of a line or an entry. **`/t$/`** does not correspond to 't' in «later», but to «sept».
`*`	Matches the preceding character 0 or more times. **`/bo*/`** matches 'boooo' in «The boy booooed».
`+`	Matches the preceding character 1 or more times. **`/a+/`** matches «bandit» and all the a's in «baaaaandit»
`?`	Matches the preceding character 0 or 1 time. **`/e?le?/`** matches the 'el' in «angel» and the 'le' dans «angle».
`.`	The decimal point matches any single character except the newline character. For example, **`/.n/`** matches 'an' and 'on' in "nay, an apple is on the tree", but not 'nay'.
`(x)`	Matches 'x', where x can be any expression, and remembers the match. **`/(foo)/`** matches and remembers 'foo' in «foo bar.» The matched substring can be recalled from the resulting array's elements. See property of objet RegExp.
`x|y`	Matches either 'x', or 'y' où x and they can represent any expression.
`{n}`	Where **`n`** is a positive integer. Matches exactly **`n`** occurrences of the preceeding character. **`/o{2}/`** matches the two o's in «cocoon».
`[xyz]`	A character set. Matches any of the enclosed characters. **`[abcd]`** or **`[a-d]`**.
`[\b]`	Matches a backspace.
`\b`	Matches a word boundary (before or after a word).

\cX	Where *X* is a Ctrl character.
\d	Matches a digit character. `[0-9]`.
\D	Matches any non-digit character.
\f	A form-feed
\n	A line feed.
\r	A carriage return.
\s	Equivalent to [`\f\n\r\t\v]`.
\S	Equivalent to `[^ \f\n\r\t\v]`.
\t	A tab.
\v	A vertical tab.
\w	Matches any alphanumeric character including the underscore.
\W	Any non-word character. %, 50%
\n	Where **n** is a positive integer. A back reference to the last substring matching the *n* parenthetical in the regular expression.
\ooctal **\xhex**	Where **\ooctal** is an octal escape value and **\xhex** is a hexadecimal value. Allows including ASCII code into regular expressions.

Regular Expression – Object

Properties Methods See also

The **Regular Expression** contains a model of a regular expression. Its properties and methods allow the use of a regular expression to carry out a search and replace of corresponding character strings.

Syntax

```
var regularexpression = /pattern/[switch]
or
var regularexpression = new RegExp("pattern",["switch"])
```

Parameters

pattern
Text or regular expression.
With the first syntax, it is necessary to use / character to limit the expression pattern.
With the second, it is necessary to use quotation marks.

switch
If specified, **switch** can have one of the following values
g: global match
i: ignore case
gi: a combination of the first two parameters

Reset – Object

Properties Methods Events See also

This object is a **Reset** button found on HTML forms. This button allows the user to reset all the fields in the form to their initial values.

Syntax

To define the **Reset** object, use standard HTML syntax:

```
<INPUT TYPE="reset" NAME="NomReset"..>
```

To use the properties and methods of the Reset object

```
resetName.propertyName
resetName.methodName(parameters)
formName.elements[index].propertyName
formName.elements[index].methodName(parameters)
```

resetName is the value of attribute **NAME of a Reset** object.
formName is the value of attribute NAME of the Form object or an element in a forms array.
index is an integer representing the **Reset** object in a form.

Example

```
<FORM>
<INPUT TYPE="Reset" onClick="alert('no question')">
</FORM>
```

screen – Object

Properties

Contains the properties for the screen's display and colors. The JavaScript engine creates the **screen** object by itself. These properties are accessed automatically.

Example:

```
function screen_properties() {
   document.examples.results.value = "("+screen.width+" x
    "+screen.height+") pixels, "+
    screen.pixelDepth +" bit depth, "+
    screen.colorDepth +" bit color palette depth.";
} // end function screen_properties
```

Select – Object

Properties Methods Events See also

This object is a list box or a drop-down box found on HTML forms.

- List box allows selecting one or several given elements.
  The drop-down lists limit the selection to one choice.

Syntax

To define the **Select** object, use standard HTML syntax:

```
<SELECT NAME="selectName"..> text
  <OPTION> text
</SELECT>
```

To use the properties and methods of the Select object

```
selectName.propertyName
selectName.methodName(parameters)
formName.elements[index].propertyName
formName.elements[index].methodName(parameters)
```

selectName is the value of attribute **NAME of a Select** object.
formName is the value of attribute NAME of the Form object or an element in a forms array.
index is an integer representing the **Select** object in a form.

The options array

You can reference an option of a **Select** object in your code using an options array. This array has an entry for every **Select** object (**<OPTION>** element) in the document in source order.

To use the Select object option:

```
selectName.options{index1].propertyName
formName.elements[index2].options[index1].propertyName
selectName.options.length
```

index1 is an integer representing an option of a **Select** object.
index2 is an integer representing the **Select** object in a form.

To obtain the number of elements of a **Select** object, use the **length** property.

To create an option:

```
optionName = new Option([optionText, optionValue, defaultSelect, select])
```

To add a new option to the existing Select object:

```
selectName.options[index]=
```

To delete an option of the Select object:

```
selectName.options[index] = null
```

optionName is the name of a new object or a property of an existing object.
optionText specifies the text to be displayed in the list box.
optionValue is the value of an option in the form while it is selected. This value is returned to the server when the form is submitted.
defaultSelected indicates if the option was selected initially (true or false).
Select indicates the current status of the option selection (true or false).
selectName is the name of an existing **Select** object.
index is an integer representing an option of a **Select** object.

Example

```
<FORM>
<SELECT NAME="list" SIZE=1>
  <OPTION SELECTED VALUE="">Selection
  <OPTION VALUE="index.htm">Index
  <OPTION VALUE="suscribe.htm">Inscription
  <OPTION VALUE="document1.htm">Appear
</SELECT>
<INPUT TYPE="button" VALUE="Go" onClick="if (form.list.selectedIndex != 0)
window.open(form.list.options[form.list.selectedIndex].value, 'Subscribe',
'scrollbars=yes,status=yes,width=500,height=300'); else alert('Would you
like to make a selection?')">
</FORM>??
```

String – Object

Properties Methods See also

This is a built-in object in the Javascript language. It represents a character string. A character string can be inserted in single or double quotes: `'Text'` or `"Text"`. Character strings can be created using the `String()` constructor.

Syntax

To create a **String** object:

```
stringObjectName = new String(characterString)
```

`stringObjectName` is the name of a new **String** object.

To use the String object:

```
stringName.propertyName
stringName.methodName(parameters)
```

`stringName` is the name of a **String** type variable.

Example

```
<FORM><INPUT TYPE="button" VALUE="uppercase" onClick="form.result.value =
form.string.value.toUpperCase() "><INPUT TYPE="text" NAME="string"
VALUE=gilles SIZE=6><INPUT TYPE="text" NAME="result" ></FORM>??
```

Style – Object

Properties **Methods** **See also**

This object is used to specify a style for HTML elements. This is useful in order to use DHTML style sheets (*Dynamic HTML*) with Javascript.

The object's methods and properties allow implementing CSS style properties with JavaScript.

Submit – Object

Properties Methods Events See also

This object is a *Submit* button found on HTML forms. This button allows the user to submit the values indicated in the form's fields.

Syntax

To define the **Submit** object, use standard HTML syntax:

```
<INPUT TYPE="submit" NAME="submitName">
```

To use the properties and methods of the Submit object

```
submitName.propertyName
submitName.methodName(parameters)
formName.elements[index].propertyName
formName.elements[index].methodName(parameters)
```

resetName is the value of attribute **NAME of a Submit** object.
formName is the value of attribute NAME of the Form object or an element in a forms array.
index is an integer representing the **Subject** object in a form.

Example

```
<SCRIPT LANGUAGE="JavaScript">
function Calcul (form)
{
i=eval(form.op1.value) + eval( form.op2.value)
i="Results: " + I
alert ( i)
}
</SCRIPT>
<FORM>
<INPUT NAME=op1 VALUE=1>
<B>+</B>
<INPUT NAME=op2 VALUE=2>
<INPUT NAME=result TYPE=SUBMIT VALUE="=" onClick="Calcul(this.form)">
</FORM>
```

sun – Object

See also

A top-level, built-in JavaScript object used to access any Java class of a *sun.** package using JavaScript. You can access this object without using the constructor or without calling a method.

Text – Object

Properties **Methods** **Events** **See also**

This object is a text box found on HTML forms. This box allows the user to input a line of text in a form's text box..

Syntax

To define the **Text** object, use standard HTML syntax:

```
<INPUT TYPE="text" NAME="textName">
```

To use the properties and methods of the Text object

```
textName.propertyName
textName.methodName(parameters)
formName.elements[index].propertyName
formName.elements[index].methodName(parameters)
```

textName is the value of attribute NAME of a T**ext** object.
formName is the value of attribute NAME of the Form object or an element in a forms array.
index is an integer representing the **Text** object in a form.

Example

```
<FORM NAME="fast">
<INPUT TYPE="text" NAME="field " SIZE="15" onFocus="this.value='I am a fast typist!'">
</FORM>?
```

Textarea – Object

Properties **Methods** **Events** **See also**

This object is a textarea box found on HTML forms. This box allows the user to input several lines of text in a form's text box.

Syntax

To define the **Textarea** object, use standard HTML syntax:

```
<TEXTAREA NAME="textareaName"..>
   text to be displayed
</TEXTAREA>
```

To use the properties and methods of the Textarea object

```
textareaName.propertyName
textareaName.methodName(parameters)
formName.elements[index].propertyName
formName.elements[index].methodName(parameters)
```

textareaName is the value of attribute NAME of a T**extarea** object.
formName is the value of attribute NAME of the Form object or an element in a forms array.
index is an integer representing the **Textarea** object in a form.

Example

```
<FORM>
<TEXTAREA NAME="text" COLS=27 ROWS=4 onFocus="this.value='This help is
truly well done, I think I got a good bargain!'">
</TEXTAREA>
</FORM>?
```

VBArray – Object

Methods **See also**

The **VBArray** object is used to access Visual Basic's **safe arrays** objects.

Please note that **VBArray** objects are read-only. These objects cannot be directly created. Before submitting a **VBArray** value to the **VBArray** constructor, the value must be attributed to the **safeArray** argument. To obtain this value it must first be extracted from an existing ActiveX object or another object.

Syntax

```
new VBArray(safeArray)
```

The **safeArray** argument is a **VBArray** value.

window – Object

Properties **Methods** **Events** **See also**

The top-level object for each document, Location and History object group.

The self and window properties are synonyms for the current window. You can use either property to reference the current window. For example, you can close the current window by calling either **`window.close()`** or **`self.close()`**.

The top and parent properties are also synonyms that can be used in place of the window name.

The top property refers to the browser's main window and the parent property refers to a window containing a **<FRAMESET>**element.

Syntax

To define a window, use the open method:

```
windowVar = window.open("URL", "windowName" [,"windowFeatures"])
```

`windowVar` is the name of a new window. You must use this variable to reference the window's properties, methods and contents.

`windowName` is the name of window which is useful for the **TARGET** attribute of a **<FORM>** or **<A>** element.

To use the properties and methods of the **window** object

```
window.propertyName
window.methodName(parameters)
self.propertyName
self.methodName(parameters)
top.propertyName
top.methodName(parameters)
parent.propertyName
parent.methodName(parameters)
windowVar.propertyName
windowVar.methodName(parameters)
propertyName
methodName(parameters)
```

`windowVar` is a variable that references a **window** object

Example

```
<FORM><INPUT TYPE="button" VALUE="Subscribe" NAME="subscribe_button"
onClick="window.open
   ('subscribe.htm', 'Subscribe',
'scrollbars=yes,status=yes,width=500,height=300')">
</FORM>
```

Dictionary - Object

Properties Methods See also

This object keeps in memory pairs of data known as «key-elements». The element values can be of different types and are kept in an array. Each element in the array is associated with a key that distinguishes it from other elements. The key is used to extract an individual element.

Syntax

To use the **Dictionary** object:

```
y = new ActiveXObject
"Scripting.Dictionary")
```

Example

Creation a **Dictionary** object:

```
var y = new
ActiveXObject("Scripting.Dictionary");
y.add ("a", "test");
if (y.Exists("a"))
  document.write("true");
...
```

Drive - Object

Properties See also

This object is used to access the properties of a particular disk drive or an element of a shared network.

Example

```
function ShowFreeSpace(drvPath)
{
    var fso, d, s;
    fso = new ActiveXObject("Scripting.FileSystemObject");
    d = fso.GetDrive(fso.GetDriveName(drvPath));
    s = "Drive + drvPath + " - " ;
    s += d.VolumeName + "<br>";
    s += "Available space: " + d.FreeSpace/1024 + " Kilobytes";
    return(s);
}
```

Drives - Collection

Properties See also

This collection contains a list of all the drives available. The collection is ready-only.

Example

Obtain the **Drives** collection using the Drives property and access it using the Enumerator object.

```
function ShowDriveList()
{
  var fso, s, n, e, x;
  fso = new ActiveXObject("Scripting.FileSystemObject");
  e = new Enumerator(fso.Drives);
  s = "";
  for (; !e.atEnd(); e.moveNext())
  {
    x = e.item();
    s = s + x.DriveLetter;
    s += " - ";
    if (x.DriveType == 3)
      n = x.ShareName;
    else if (x.IsReady)
      n = x.VolumeName;
    else
      n = "[Drive not ready]";
    s += n + "<br>";
  }
  return(s);
}
```

File - Object

Properties Methods See also

This object is used to access all file properties.

Example:

```
function ShowFileInfo(filespec)
{
  var fso, f, s;
  fso = new ActiveXObject("Scripting.FileSystemObject");
  f = fso.GetFile(filespec);
  s = f.DateCreated;
  return(s);
}
```

Files - Collection

Properties See also

This collection is made up of a Folder's File objects.

Example:

```
function ShowFolderFileList(folderspec)
{
  var fso, f, f1, fc, s;
  fso = new ActiveXObject("Scripting.FileSystemObject");
  f = fso.GetFolder(folderspec);
  fc = new Enumerator(f.files);
  s = "";
  for (; !fc.atEnd(); fc.moveNext())
  {
      s += fc.item();
      s += "<br>";
  }
  return(s);
}
```

FileSystemObject - Object

Properties Methods See also

This object is used to access a computer's file system.

Syntax:

```
y = new ActiveXObject("Scripting.FileSystemObject")
```

Example:

```
var fso = new ActiveXObject("Scripting.FileSystemObject");
var a = fso.CreateTextFile("c:\file-test.txt", true);
a.WriteLine("This is a test.");
a.Close();
```

In the previous example, a file (file-test.txt) is created in the C drive root directory: (C:\file-test.txt) with a short text inserted in the file (This is a test). The file is then closed and the buffer memory cleared.

Folder - Object

Properties Methods See also

This object is used to access all folder properties.

Example:

```
function ShowFolderInfo(C:\folder)
{
  var fso, folder, s;
  fso = new ActiveXObject("Scripting.FileSystemObject");
  folder = fso.GetFolder(C:\folder);
  s = folder.DateCreated;
  return(s);
}
```

Folders - Collection

Properties Methods See also

This collection is made up of all the **Folder** objects contained in a **Folder** object.

Example:

```
function ShowFolderList(folderspec)
{
  var fso, f, fc, s;
  fso = new ActiveXObject("Scripting.FileSystemObject");
  f = fso.GetFolder(folderspec);
  fc = new Enumerator(f.SubFolders);
  s = "";
  for (; !fc.atEnd(); fc.moveNext())
  {
    s += fc.item();
    s += "<br>";
  }
  return(s);
}
```

TextStream - Object

Properties Methods See also

This object is used to facilitate sequential access to a file.

Syntax

```
TextStream.{property | method( )}
```

Example

```
var fso = new ActiveXObject("Scripting.FileSystemObject")
var a = fso.CreateTextFile("c:.txt", true)
a.WriteLine("This is a test.")
a.Close();
```

In this example, **a** is the **TextStream** object resent by the CreateTextFile method over the FileSystemObject object. .

$1...$9 – Property of object: RegExp

See also

Properties that contain substring matches, if any. More exactly, corresponding parenthesized substrings.
The number of possible parenthesized substrings is unlimited, but the predefined RegExp object can only hold 9.

Example

```
text = "Peter Smith";
group = /(\w+)\s(\w+)/;
newtext=text.replace(group, "$2, $1");
document.write(newtext)
```

The result is: Smith, Peter

$_ – Property of object: RegExp

See also

For an array created through a match with a regular expression, this property represents the original text string for which a match was found with the regular expression.

$* – Property of object: RegExp

See also

This property determines whether or not to search in strings across one or multiple lines.

$& – Property of object: RegExp

See also

This property refers to the last corresponding characters.

$+ – Property of object: RegExp

See also

This property corresponds to the last parenthesized substring match.

$` – Property of object: RegExp

See also

This property represents the substring preceding the most recent match.

$' – Property of object: RegExp

See also

This property represents the substring following the most recent match.

above – Property of object: Layer

See also

The **`layer`** object above the current **`layer`** object or **Null** if the current **`layer`** is the first one in the stack.

action – Property of object: Form

See also

This property represents a text string specifying a destination URL for form data that is submitted.

Syntax

```
formName.action
```

formName is the name of a form or an element in a tables array.

Example

```
<FORM NAME=courrier>
Address:<BR>
<INPUT TYPE="text" NAME=email VALUE="webmaster@ungi.com" MAXLENGTH=50
SIZE=25><BR>
Subject:<BR>
<INPUT TYPE="text" NAME=subject VALUE="hello"MAXLENGTH=50 SIZE=25><BR>
Email body:<BR>
<TEXTAREA NAME="textareaName" ROWS=5 COLS=40>Text contained in the input
box</TEXTAREA><BR>
<INPUT TYPE="submit" NAME="Send" VALUE="send mail"
onClick="sendMail();return true;">
</FORM>

<SCRIPT LANGUAGE="JavaScript">

function sendMail()

{
address="mailto:"+document.mail.email.value+"?
subject="+document.mail.subject.value;
document.mail.action=address;
document.mail.submit();
alert(document.mail.action);
}
</SCRIPT>
```

align – Property of object: Style

See also

Specifies the alignment of an HTML element within its parent.

Syntax

```
styleObject.align = {left | right | none}
```

Parameters

styleObject represents a Style object.
left specifies a left alignment.
right specifies a right alignment.
none no alignment specified. The element inherits the alignment value from its parent element.

Note

Do not confuse the **align** property with textAlign, the latter specifies the alignment of an element's content and not of the element itself.

alinkColor – Property of object: document

See also

This property defines the color of an active link, that is, the color of the text hyperlink when the user activates the link using the mouse.

Syntax

```
document.alinkColor
```

Example

```
<A HREF="http://www.imaginet.fr/ime/">UNGI !</A>
<BR><BR>
<FORM>
Color of links:<BR>
<INPUT TYPE="button" NAME="Red" VALUE="link turns red when the user clicks
on it" onClick="changeLink('red')">
<INPUT TYPE="button" NAME="Green" VALUE="link turns green when the user
clicks on it" onClick="changeLink('green')">
<INPUT TYPE="button" NAME="Blue" VALUE="link turns blue when the user
clicks on it" onClick="changeLink('blue')">
</FORM>

<SCRIPT LANGUAGE="JavaScript">
function changeLink(color)
{
if(color=="red")

document.alinkColor="FF0000";
if(color=="green")
     document.alinkColor="00FF00";
if(color=="blue")
     document.alinkColor="0000FF";
}
</SCRIPT>
```

anchors – Property of object: document

See also

This property represents an objects array corresponding to the names of **<A>** elements found in HTML code. Refer to the Anchor object.

Example

```
<A NAME="anchor1"></A>
<A NAME="anchor2"></A>
<A NAME="anchor3"></A>

<SCRIPT LANGUAGE="JavaScript">
document.write("number of hypertext links:
"+document.anchors.length+"<BR>");
</SCRIPT>
```

appCodeName – Property of object: navigator

See also

A string specifying the code name of the browser. Used as read-only.

Syntax

```
navigator.appCodeName
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.writeln("The codename of your browser  (navigator.appCodeName)  is
" + navigator.
appCodeName);
</SCRIPT>
```

applets – Property of object: document

See also

Contains the objects array indicating the Java Applets found in the document, according to their source order.
You can reference the applets in your code by using the applets array of the **applets** property. This array contains an entry for every **Applet** object (**<APPLET>** element) in a document.
For example, if a document contains four applets whose attributes **NAME** are **`applet1`**, **`applet2`**, **`applet3`** and **`applet4`**, you can refer to these objects as follows:

```
document.applets["applet1"]
document.applets["applet2"]
document.applets["applet3"]
```

or

```
document.applets[0]
document.applets[1]
document.applets[2]
```

To obtain the number of applets found in a document, you must use the length property:

```
document.applets.length
```

appName – Property of object: navigator

See also

A string specifying the name of the browser. Used as read-only.

Syntax

```
navigator.appName
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.writeln("You are using (navigator.appName): " +navigator.appName);
</SCRIPT>
```

appVersion – Property of object: navigator

See also

This property specifies the browser's version. Used as read-only.

Syntax

```
navigator.appVersion
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.writeln("Your browser's version (navigator.appVersion) is: "
+navigator.
appVersion+ "<BR> and the format is: version [language] (platform;
country)");
</SCRIPT>
```

arguments – Property of object: Function

See also

Returns an array corresponding to the arguments passed to a Function object.

Syntax

```
functionName.arguments[argument1, argument2, …, argumentn]
```

functionName represents the Function object.
argument1 to **agumentn** represent the function's arguments.

arguments.caller – Property of object: Function

See also

Returns a text string indicating the instructions found in the function being executed.

Note
This property is used exclusively within a function.

arguments.caller – Property of object: Function

See also

Specifies the name of the function that invoked the currently executing function.

Note
This function is not used anymore.

arguments.length – Property of object: Function

See also

Specifies the number of arguments used to call the Function object.

arity – Property of object: Function

See also

Specifies the number of arguments expected by the Function object.

Syntax

```
functionName.arity
```

`functionName` represents the Function object.

availHeight – Property of object: screen

See also

Specifies the height of the screen, in pixels, minus permanent or semi-permanent user interface features displayed by the operating system, (ex. the Taskbar on Windows).

availLeft – Property of object: screen

See also

Specifies the *x-coordinate* of the first pixel that is not allocated to permanent or semi-permanent user interface features.

availTop – Property of object: screen

See also

Specifies the *y-coordinate* of the first pixel that is not allocated to permanent or semi-permanent user interface features.

availWidth – Property of object: screen

See also

Specifies the width of the screen, in pixels, minus permanent or semi-permanent user interface features displayed by the operating system, (ex. the Taskbar on Windows).

background – Property of object: Layer

See also

This property is used to indicate the image source that must be displayed in the background of the *layer*).

Syntax

```
layer.background.src = "imageName.gif";
```

backgroundColor – Property of object: Style

See also

Specifies a background color for an HTML element.

Syntax

```
styleObject.backgroundColor = colorValue
```

Parameters

styleObject represents a Style object.
colorValue is a string representing a color value (the name of a color or the color's hexadecimal value).

backgroundImage – Property of object: Style

See also

Specifies a background image for an HTML element.

Syntax

```
styleObject.backgroundImage = url
```

Parameters

styleObject represents a Style object.
url is a string representing the URL of a style sheet.

below – Property of object: Layer

See also

The layer object below the current layer object or **Null** if the current layer is the last in the stack.

bgColor – Property of objects: document and Layer

See also

This property defines the color of the document background or that of a Layer object.

Syntax

```
document.bgColor
```

Example

```
<FORM>
Change the background color:<BR>
<INPUT TYPE="button" NAME="red" VALUE="red background"
onClick="changeFond('red')">
<INPUT TYPE="button" NAME="green" VALUE="green background"
onClick="changeFond('green')">
<INPUT TYPE="button" NAME="blue" VALUE="blue background"
onClick="changeFond('blue')">
</FORM>

<SCRIPT LANGUAGE="JavaScript">
function changeBackground(color)
{
if(color=="red")
     document.bgColor="FF0000";
if(color=="green")
     document.bgColor="00FF00";

if(color=="blue")
     document.bgColor="0000FF";
}
</SCRIPT>
```

border – Property of object: Image

See also

This property is a text string specifying the width in pixels of an image border. This is read-only.

Syntax

```
imageName.border
```

imageName is the name of an Image object or an element in an images array.

Example

```
<IMG SRC="ungi.gif" NAME="ungi" WIDTH=120 HEIGHT=80 BORDER=4>
<BR>
<SCRIPT LANGUAGE="JavaScript">
document.writeln("The above image has the following width:
"+document.ungi.border);
</SCRIPT>
```

borderBottomWidth – Property of object: Style

See also

Specifies the width of the bottom border of an HTML element.

Syntax

```
styleObject.borderBottomWidth = length
```

Parameters

styleObject represents a Style object.
length is a string indicating a number followed by a unit of measurement, for example, 10px.

borderColor – Property of object: Style

See also

Specifies the color of the border of an HTML element.

Syntax

```
styleObject.borderColor = {none | colorValue}
```

Parameters

styleObject represents a Style object.
colorValue is a string representing a color value (the name of a color or the color's hexadecimal value).

borderLeftWidth – Property of object: Style

See also

Specifies the width of the left border of an HTML element.

Syntax

```
styleObject.borderLeftWidth = length
```

Parameters

styleObject represents a Style object.
length is a string indicating a number followed by a unit of measurement, for example, 10px.

borderRightWidth – Property of object: Style

See also

Specifies the width of the right border of an HTML element.

Syntax

```
styleObject.borderRightWidth = length
```

Parameters

styleObject represents a Style object.
length is a string indicating a number followed by a unit of measurement, for example, 10px.

borderStyle – Property of object: Style

See also

Specifies the style of border around a block-level HTML element.

Syntax

```
styleObject.borderStyle = styleType
```

Parameters

styleObject represents a Style object.
styleType is a string representing the following values: none, solid, double, inset, outset, groove, ridge.

Note

A border width (borderWidth) must be defined for the border to be visible.

borderTopWidth – Property of object: Style

See also

Specifies the width of the top border of an HTML element.

Syntax

```
styleObject.borderTopWidth = length
```

Parameters

styleObject represents a Style object.
length is a string indicating a number followed by a unit of measurement, for example, 10px.

caller – Property of object: Function

See also

Specifies the name of the function that invoked the currently executing function.

Note
This function is no longer in use.

checked – Property of objects: Checkbox and Radio

See also

A Boolean value specifying the selection state of a checkbox or a radio button (checked or not checked).

Syntax

```
checkboxName.checked
radioName[index].checked
```

checkboxName is the value of attribute NAME of the Checkbox object or an element in an elements array.
radioName is the value of attribute NAME of a Radio object.
index is an integer representing a radio button in a Radio object.

Example

```
<FORM NAME=form>
<INPUT TYPE="checkbox" NAME="case1" VALUE="Selection" onClick=checktest()>
</FORM>

<SCRIPT LANGUAGE="JavaScript">
function checktest()
{
if(document.form.case1.checked=="1")
     alert('box checked)
else
     alert('box not checked)
}
</SCRIPT>
```

classes – Property of object: document

See also

Creates a Style object that specifies the styles to apply to HTML tags with a specific **CLASS** attribute.

Syntax

```
document.classes.className.tagName
```

Parameters

className is the value assigned to the **CLASS** attribute of the HTML tag.
tagName is the name of the HTML tag (**<H1>**, **<P>**, **<BLOCKQUOTE>**).

className – Property of object: Packages

See also

The fully qualified name of a Java class, accessible to JavaScript, in a package other than *netscape, java* or *sun*.

Syntax

```
Packages.className
```

className is the fully qualified name of a Java class.

clear – Property of object: Style

See also

Specifies the sides of an HTML element that allow floating elements.

Syntax

```
styleObject.clear = {left | right | both | none}
```

Parameters

styleObject represents a Style object.
left allows floating elements to the left side.
right allows floating elements to the right side.
both allows floating elements on both sides.
none does not allow floating elements on either side.

clip.bottom – Property of object: Layer

See also

Represents the bottom edge of the rectangle of a Layer. Any part of a Layer that is outside the rectangle (content overflow) is not displayed.

clip.height – Property of object: Layer

See also

Represents the height of the rectangle of a Layer. Any part of a Layer that is outside the rectangle (content overflow) is not displayed.

clip.left – Property of object: Layer

See also

Represents the left edge of the rectangle of a Layer. Any part of a Layer that is outside the rectangle (content overflow) is not displayed.

clip.right – Property of object: Layer

See also

Represents the right edge of the rectangle of a Layer. Any part of a Layer that is outside the rectangle (content overflow) is not displayed.

clip.top – Property of object: Layer

See also

Represents the top edge of the rectangle of a Layer. Any part of a Layer that is outside the rectangle (content overflow) is not displayed.

clip.width – Property of object: Layer

See also

Represents the width of the rectangle of a Layer. Any part of a Layer that is outside the rectangle (content overflow) is not displayed.

closed – Property of object: window

See also

Specifies whether a window is closed. If so, the value returned is **true**. This information could be useful so that a script does not attempt to manipulate a closed window (the user might have clicked on the window's close box before the script has finished executing).

color – Property of object: Style

See also

Specifies the color of text in an HTML element.

Syntax

```
styleObject.color = colorValue
```

Parameters

styleObject represents a Style object.
colorValue is a string representing a color value (the name of a color or the color's hexadecimal value).

colorDepth – Property of object: screen

See also

Represents the number of bits of the color palette used.
For example, the value 8 indicates a 8 bit color palette (256 colors).

complete – Property of object: Image

See also

A Boolean value that indicates whether the browser has completed its attempt to load an image. Used as read-only.

Syntax

```
imageName.complete
```

imageName is the name of an Image object or an element in an images array.

Example

```
<IMG SRC="birds.jpg" NAME="ungi" WIDTH=120 HEIGHT=80 BORDER=4>
<BR>
<SCRIPT LANGUAGE="JavaScript">

document.writeln("The complete value of
document.ungi.is:"+document.ungi.complete);
/*
if(document.ungi.complete=="false")
     {
     document.writeln("While this line is displayed, the image is being
loaded.");
     }
else
     {
     document.writeln("While this line is displayed, the image has been
loaded.");
}
*/
function completetest()
{
if(document.ungi.complete=="false")
     {
     alert(document.ungi.complete+': The         image is being loaded.');
     }
else
     {
     alert(document.ungi.complete+': The image is loaded.');
     }
}
</SCRIPT>

<BR>Testing!<BR>
<FORM NAME=form>
<INPUT TYPE="button" NAME="finished" VALUE="Is the display completed?"
onClick=completetest()>
</FORM>
```

constructor – Property of objects: Array, Boolean, Date, Function, Number, Object, RegExp and String

See also

Specifies the function that creates an object.

Syntax

```
Object.constructor
```

`Objet` represents the name of a function or an object.

cookie – Property of object: document

See also

A cookie is a small piece of information stored by the browser in the **cookies.txt** file. This property can be used with methods substring, charAt, indexOf and lastIndexOf to determine the value stored in the **cookie**.

Syntax

```
document.cookie
```

Example

```
<SCRIPT LANGUAGE="JavaScript">

var expression = "mmmm_the_good_chocolate_bits";
today = new Date()
var time = today.toGMTString()
document.cookie = "@" + time + "=" + expression + ";"
</SCRIPT>
```

crypto – Property of object: window

See also

An object used as a property of the: window object which provides to access Netscape Navigator's encryption features.

current – Property of object: History

See also

A string specifying the complete URL of the current history entry (History object).
Obtaining the value of this property requires the *UniversalBrowserRead* privilege. It has no value unless you have access to this privilege.

data – Property of object: event

See also

Used with the onDragDrop event, this property creates an array of a string containing the URLs (addresses) of dropped objects.

Getting the value of this property requires the *UniversalBrowserRead* privilege. It has no value unless you have access to this privilege.

defaultChecked – Property of objects: Checkbox and Radio

See also

A Boolean value specifying the default selection state of a checkbox or a radio button (checked or not checked).

Syntax

```
checkboxName.defaultchecked
radioName[index].defaultchecked
```

checkboxName is the value of attribute NAME of the Checkbox object or an element in an elements array.
radioName is the value of attribute NAME of a Radio object.
index is an integer representing a radio button in a Radio object.

Example

```
<FORM NAME=form>
box1:<INPUT TYPE="checkbox" NAME="box1" CHECKED><BR>
box2:<INPUT TYPE="checkbox" NAME="box2"><BR>
</FORM>

<SCRIPT LANGUAGE="JavaScript">
if(document.form.box1.defaultChecked==true)
     document.writeln("Box1 is checked by default
document.form.box1.defaultChecked
="+document.form.box1.defaultChecked+"<BR>");
if(document.form.box1.defaultChecked==false)
     document.writeln("Box1 is not checked by
defaultdocument.form.box1.defaultChecked
="+document.form.box1.defaultChecked+"<BR>");

if(document.form.box1.defaultChecked==true)
     document.writeln("Box2 is checked by default
document.form.box2.defaultChecked
="+document.form.box2.defaultChecked+"<BR>");
if(document.form.box2.defaultChecked==false)
     document.writeln("Box2 is not checked by default
document.form.box2.defaultChecked
="+document.form.box2.defaultChecked+"<BR>");
</SCRIPT>
```

defaultSelected – Property of object: Option

See also

This property has a Boolean value and indicates the default selection state of an option of a Select or Option object (selected or not selected).

Syntax

```
selectName.options[index].defaultSelected
optionName.defaultSelected
```

selectName is the value of attribute NAME of the Select object or an element in an elements array.
index is an integer representing an option of a Select object.
optionName is the name of a Select object created with the Option() constructor.

Example

```
<FORM NAME=form>
<SELECT NAME=list MULTIPLE SIZE=6>
     <OPTION SELECTED>selection1
     <OPTION SELECTED>selection2
     <OPTION>selection3
     <OPTION SELECTED>selection4
     <OPTION SELECTED>selection5
     <OPTION>selection6
</SELECT>
</FORM>

List of elements selected by default:<BR>

<SCRIPT LANGUAGE="JavaScript">
for (i=0; i < document.form.list.length; i++)
     {
     if(document.form.list[i].defaultSelected==true)

{
           document.writeln(document.form.list[i].text+" is selected by
default<BR>");
           }
     }
</SCRIPT>
```

defaultStatus – Property of object: window

See also

This property displays a message in the status bar (at the bottom of the window).

Syntax

```
windowReference.defaultStatus
```

windowReference is a valid way to reference a window, as it is described in the window object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
window.defaultStatus = "Status bar";
<SCRIPT LANGUAGE="JavaScript">
```

defaultValue – Property of objects: Password, Text and Textarea

See also

This property is a string representing the default value of a Password, Text or Textarea object.

Syntax

```
passwordName.defaultValue
textName.defaultValue
textareaName.defaultValue
```

passwordName is the value of attribute NAME of the Password object or an element in an elements array.
textName is the value of attribute NAME of the Text object or an element in an elements array.
textareaName is the value of attribute NAME of the Textarea object or an element in an elements array.

Example

```
<FORM NAME=form>
<INPUT TYPE="password" NAME="passwordField" VALUE="password" SIZE=15>
</FORM>

<SCRIPT LANGUAGE="JavaScript">
document.writeln("The default password for this field is:
"+document.form.passwordField.defaultValue+"<BR>");
</SCRIPT>
```

description – Property of objects: MimeType and Plugin

See also

This property is used to describe a PlugIn.

Syntax

```
navigator.mimeTypes[index].description
navigator.mimeTypes[mimeTypeName].description
navigator.plugins[index].description
navigator.plugins[pluginName].description
```

Example

```
Description of Mime Types:<BR>

<SCRIPT LANGUAGE="JavaScript">

for (i=0; i < navigator.mimeTypes.length; i++)
           document.writeln(navigator.mimeTypes[i].description+" <BR>");
</SCRIPT>
```

description (Error) – Property of object: Error

See also

This property gives a description of an error.

Syntax

```
errorName.description[=text]
```

errorName represents the Error object.
text represents the text associated with the Error object.

display – Property of object: Style

See also

Overrides the usual display of an element and specifies how it is displayed in the screen: in line, as a block-level element, or as a block-level list item.

Syntax

```
styleObject.display = styleType
```

Parameters

`styleObject` represents a Style object.
`styleType` is a string representing the following values: **none** (no changes), **block** (block level element), **inline** (inline element), **list-item** (block-level list item).

document – Property of objects: Layer and window

See also

Represents the document object associated with the Layer object.

domain – Property of object: document

See also

Specifies the domain name of the server that supplied the document.

E – Property of object: Math

See also

Represents Euler's constant. The base of natural logarithms of this constant is approximately 2,718.

Syntax

```
Math.E
```

Attributes

DontEnum, DontDelete, ReadOnly

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.write("Euler's constant Math.E ="+Math.E+" .<BR>");
</SCRIPT>
```

elements – Property of object: Form

See also

This property represents an array of objects corresponding to form elements such as Checkbox, Radio and Text objects.

Example

```
<FORM NAME=form>
     <INPUT TYPE="text" NAME=textfield1 MAXLENGTH=50 SIZE=25 VALUE="this
is"><BR>
     <INPUT TYPE="text" NAME=textfield2 MAXLENGTH=50 SIZE=25
VALUE="an"><BR>
     <INPUT TYPE="text" NAME=textfield3 MAXLENGTH=50 SIZE=25
VALUE="elements"><BR>
     <INPUT TYPE="text" NAME=textfield4 MAXLENGTH=50 SIZE=25
VALUE="array"><BR>
</FORM>

<SCRIPT LANGUAGE="JavaScript">
for (i=0; i < document.form.elements.length; i++)
     {
     document.write(document.form.elements[i].value+" ");

}
</SCRIPT>
```

embeds – Property of object: document

See also

This property represents an array containing an entry for each embedded object (*plugin*) in the document.
You can reference (*plugin*) modules in your code by calling the array of the **`embeds`** property. This array contains an entry for every **embed** object (**EMBED>** element) in a document.
For example, in a document containing four embedded objects whose **NAME** attributes are **`plug1`**, **`plug2`**, **`plug3`** and **`plug4`**, you can reference these objects as follows:

```
document.embeds["plug1"]
document.embeds["plug2"]
document.embeds["plug3"]
```

or

```
document.embeds[0]
document.embeds[1]
document.embeds[2]
```

To obtain the number of embedded objects in a document, use the length property:

```
document.embeds.length
```

enabledPlugin – Property of object: MimeType

See also

This property returns a reference to a Plugin object for the (*plugin*) embedded object configured for the MIME type. If the MIME type does not have a (*plugin*) configured, the returned reference is **NULL**.

Example

```
<SCRIPT LANGUAGE="JavaScript">
mimetype = navigator.mimeTypes["audio/wav"]
if (MimeType)
      {
      plugin = MimeType.enabledPlugin
      if (plugin)
            document.writeln("Play sound! <BR><EMBED SRC=cyborg.wav>")
      else
      document.writeln("Get your plugin..")
      }
else
      {
      document.writeln("MIME Type is unavailable.")
      }
</SCRIPT>
```

encoding – Property of object: Form

See also

This property specifies the MIME encoding of the form. The encoding property initially reflects the ENCTYPE attribute of the **<FORM>** tag.

Syntax

```
formName.encoding
```

formName is the name of a form or an element in a tables array.

Example

```
<FORM
  NAME=form
  METHOD=POST
  ACTION="mailto:webmaster@ungi.com subject=hello"
  ENCTYPE="text/plain"
  TARGET="_top">
     <TEXTAREA NAME="textareaName" ROWS=5 COLS=40>Text found in the input
box</TEXTAREA>
</FORM>

<SCRIPT LANGUAGE="JavaScript">
document.write("Encoding type:"+document.form.encoding);
</SCRIPT>
```

fgColor – Property of object: document

See also

This property specifies the color of the document (foreground) text.

Syntax

```
document.fgColor
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.fgColor="#FF0000";
document.write("This is RED text<BR>");
</SCRIPT>
```

filename – Property of object: Plugin

See also

This property is the name of a (*plugin*) file on disk.

Syntax

```
navigator.plugins[index].filename
navigator.plugins[PluginName].filename
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
for (i=0; i < navigator.plugins.length; i++)
{
     document.writeln(navigator.plugins[i].name,
          navigator.plugins[i].description,
     navigator.plugins[i].filename+"<BR>");
}
</SCRIPT>
```

fontFamily – Property of object: Style

See also

Specifies the font family, such as Helvetica, Times or Arial, for an HTML text element.

Syntax

```
styleObject.fontFamily = {specificFamily | genericFamily}
```

Parameters

styleObject represents a Style object.
specificFamily is a string representing a list separated by commas, containing the names of font families (ex.: Helvetica, Times, Arial…).
genericFamily is a string representing one of the following values: **serif, sans-serif, cursive, monospace, fantasy.**

fontSize – Property of object: Style

See also

Specifies the font size for an HTML text element.

Syntax

```
styleObject.fontSize = {absoluteSize | relativeSize | length | percentage}
```

Parameters

styleObject represents a Style object.
absoluteSize is a string representing one of the following values: **xx-small**, **x-small**, **small**, **medium**, **large**, **x-large**, **xx-large**.
relativeSize is a string evaluating to a size relative to the size of the parent element, indicated by either of the following values: **smaller**, **larger**.
length is a string presenting a value (number) followed by a unit of measurement, for example, 14pt.
percentage A string evaluating to a percent of the size of the parent element. For example, 125%.

fontStyle – Property of object: Style

See also

Specifies the style of the font of an HTML element.

Syntax

```
styleObject.fontStyle = styleType
```

Parameters

styleObject represents a Style object.
styleType is a string representing the following values: **normal**, **italic**.

fontWeight – Property of object: Style

See also

Specifies the weight of the font of an HTML element.

Syntax

```
styleObject.fontWeight = {absolute | relative | numeric}
```

Parameters

styleObject represents a Style object.
absolute is a string representing one of the following values: **normal, bold**
relative A string evaluating to a weight relative to the weight of the parent element. Possible values are: **bolder** (width increased by one step), **lighter** (width decreased by one step).
numeric is a number representing a font width. The numeric values are between 100 and 900, using 100 increments. Value 100 displays thin characters while 900 displays thick characters.

form – Property of objects: Button, Checkbox, FileUpload, Hidden, Password, Radio, Reset, Select, Submit, Text and Textarea

See also

This property is used to indicate a reference to a form containing the object.
All the objects in a form have a **`form`** property indicating the reference to the form that contains them.

formName – Property of object: document

See also

All document objects have a property separated by each document form. The name of this property is the value of its **name** attribute.

forms – Property of object: document

See also

This property represents an objects array corresponding to forms (**<FORM>** elements) found in a document. Refer to the Form object.

Example

```
<FORM NAME=form>
<SELECT NAME=list>
<OPTION SELECTED>selection1
<OPTION SELECTED>selection2
</SELECT>
<BR>
</FORM>
<FORM NAME=form2>
<INPUT TYPE="button" VALUE="Button"><BR>
</FORM NAME=form>
<FORM NAME=form3>
<TEXTAREA NAME="box" ROWS=3 COLS=40></TEXTAREA><BR>
</FORM>

<SCRIPT LANGUAGE="JavaScript">
document.write("There are document.forms.length ="+document.forms.length+"
forms in this document");
</SCRIPT>
```

frames – Property of object: window

See also

This property represents an objects array corresponding to forms (**<FRAME>** elements) found in a document. Refer to the Frame object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.write("The name of the frame of the parent window is
parent.frames.length = "+parent.frames.length+"<BR>");
</SCRIPT>
```

global – Property of object: RegExp

See also

Determine if the "**g**" (global search) parameter is used or not in the regular expression. Its value is **true** if used, and **false** if not.

hash – Property of objects: Link and Location

See also

This property represents a text string beginning with the "#" hash mark. This mark specifies an anchor name in the URL.

Syntax

```
links[index].hash
location.hash
areaName.hash
```

index is an integer representing the Link object.
areaName is the value of attribute **NAME of an** Area object.

Example

```
<A HREF="http://www.labas.com:2000/directory/file.htm#interesting">This is
a link</A>
<BR>
<SCRIPT LANGUAGE="JavaScript">
document.write("The anchor of link zero is:"+document.links[0].hash);
</SCRIPT>
```

height – Property of objects: document, event, Image and screen

See also

This property is a text string specifying the height of an image, in pixels, or as a percentage of the window's height. It is the same as the **HEIGHT** attribute of the **<IMG>** element. This property is read-only.

Syntax

```
imageName.height
```

imageName is the name of an Image object or an element in an images array.

Example

```
<IMG SRC="ungi.gif" NAME="ungi" WIDTH=120 HEIGHT=80 BORDER=4>Logo UNGI!
<BR>
<SCRIPT LANGUAGE="JavaScript">
document.writeln("Height around the image: "+document.ungi.height);
</SCRIPT>
<BR>
```

history – Property of object: window

See also

Contains information of the URL sites the user has visited using the current window.

host – Property of objects: Link and Location

See also

This property represents a string specifying the server name, subdomain, and domain name. The **host** string specifies a portion of the URL and it is a substring of the hostname property.

hostname – Property of objects: Link and Location

See also

A string containing the full hostname of the server, including the server name, subdomain, domain, and port number This hostname property is the concatenation of the host and port separated by a colon(:).

href – Property of objects: Link and Location

See also

This property represents a string specifying the entire URL.

Syntax

```
links[index].href
location.href
areaName.href
```

index is an integer representing the Link object.
areaName is the value of attribute **NAME of an** Area object.

Example

```
<A HREF="http://www.networkname.com:2000/directory/file.htm#interesting">
This is a link </A>
<BR>
<SCRIPT LANGUAGE="JavaScript">
document.write("The URL of links [0] is:"+document.links[0].href);
</SCRIPT>
```

hspace – Property of object: Image

See also

This property is a string specifying the horizontal spacing between the edges of an image and the surrounding text. The value is expressed in pixels. It is the same as the **HSPACE** attribute of the **<IMG>** element. This property is read-only.

Syntax

```
imageName.hspace
```

imageName is the name of an Image object or an element in an images array.

Example

```
<IMG SRC="ungi.gif" NAME="ungi" WIDTH=120 HEIGHT=80 BORDER=4 HSPACE=40>Logo
UNGI !
<BR>
<SCRIPT LANGUAGE="JavaScript">
document.writeln("Horizontal spacing around an image:
"+document.ungi.hspace);
</SCRIPT>
```

ids – Property of object: document

See also

This property creates a Style object that can specify the style of individual HTML tags. It is necessary that the HTML element have a specific **id** attribute.

ignoreCase – Property of object: RegExp

See also

Indicates whether the "**i**" (ignore case) is used with the regular expression. Its value is **true** if used, and **false** if not.

images – Property of object: document

See also

Represents an array containing an entry for each image in the document according to the source order.
You can reference the images in your code by using the array of the **images** property. This table contains an entry for each Image **object** (<**IMG**> element) in a document.
For example, in a document containing attributes **NAME** **image1** and **image2**, you can reference these objects in the following manner:

```
document.images["image1"]
document.images["image2"]
```

or

```
document.images[0]
document.images[1]
```

To obtain the number of images found in a document, you must use the length property:

```
document.images.length
```

index – Property of objects: Array, Option and RegExp

See also

Returns the position of a character where the first match is found in a string.

Syntax

```
selectName.options[indexValue].index
optionName.index
```

selectName is the value of attribute NAME of the Select object or an element in an elements array.
indexValue is an integer representing an option of a Select object.
optionName is the name of a Select option created with the Option() constructor.

Example

```
<FORM NAME=form>
<SELECT NAME=list>
     <OPTION NAME="sel1">selection1
     <OPTION NAME="sel2">selection2
     <OPTION NAME="sel3">selection3
</SELECT>
</FORM>
Index list of list:<BR>
<SCRIPT LANGUAGE="JavaScript">
for (i=0; i < document.form.list.length; i++)
     {


document.writeln("document.form.list["+i+"].index="+document.form.list[i].
index+" <BR>");
     }
</SCRIPT>
```

Infinity – Property

See also

A numeric value representing infinity. **`Infinity`** is a top-level property and is not associated with any object.

Syntax

```
Infinity
```

innerHeight – Property of object: window

See also

Specifies the vertical dimension, in pixels, of the window's content area.

innerWidth – Property of object: window

See also

Specifies the horizontal dimension, in pixels, of the window's content area.

input – Property of objects: Array and RegExp

See also

For an array created by a regular expression match, this property represents the original string against which the regular expression was matched.

java – Property of object: Packages

See also

Use the java property to access any class in the *java* package within JavaScript.

language – Property of object: navigator

See also

Indicates the language of the Navigator being used. The value is expressed by two letters (for example, fr, en, es) and sometimes by 5 letters (zh_CN).

lastIndex – Property of object: RegExp

See also

This property is read/write. It allows specifying the index at which to start the next search in the text. It can only be used in global searches (*g* parameter).

lastMatch – Property of object: RegExp

See also

This property refers to the last corresponding characters. **$&** is another name for the same property.

Note

lastMatch is static, it is not a property of an individual regular expression object. This property is always used as follows: **RegExp lastMatch**

lastModified – Property of object: document

See also

This property represents a string containing the date that a document was last modified. This is read-only.

Syntax

```
document.lastModified
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.writeln("Date this document was last modified:
"+document.lastModified);
</SCRIPT>
<BR>
```

lastParen – Property of object: RegExp

See also

This property corresponds to the last parenthesized substring match. $+ is another name for the same property.

Note

lastParen is static, it is not a property of an individual regular expression object. This property is always used as follows: **RegExp. lastParen**.

layers – Property of object: document

See also

Represents an array containing an entry for each *layer* in the document according to the source order.
You can reference the *layers* in your code by using the array of the **`layers`** property. This array contains an entry for every Layer **object** (<**LAYER>** or <**ILAYER>** elements) in a document.
For example, in a document containing two *layers* where the attributes **NAME** are **`layer1`** and **`layer2`**, you can reference these objects in the following manner:

```
document.layers["layer1"]
document.layers["layer2"]
```

or

```
document.layers[0]
document.layers[1]
```

To obtain the number of *layers* in a document, use the length property:

```
document.layers.length
```

layerX – Property of object: event

See also

Number specifying either the object width when passed with the onResize event, or the cursor's horizontal position in pixels relative to the *layer* in which the event occurred.

layerY – Property of object: event

See also

Number specifying either the object height when passed with the onResize event, or the cursor's vertical position in pixels relative to the *layer* in which the event occurred.

left – Property of object: Layer

See also

Represents the horizontal position of the left edge of a *layer* with relation to its parent *layer* . The value is expressed in pixels.

leftContext – Property of object: RegExp

See also

The substring preceding the most recent match. $` is another name for the same property.

Note

leftContext is static, it is not a property of an individual regular expression object. This property is always used as follows: **RegExp. leftContext**.

length – Property of objects: Form, History, JavaArray, Option, Plugin, Select and window

See also

With respect to the object for which this property is used, it represents the number of these type of objects found in the document.

length (Array) – Property of object: Array

See also

Returns the number of elements contained in the array.

Syntax

```
arrayName.length
```

arrayName corresponds to the Array object created.

length (Function) – Property of object: Function

See also

Specifies the number of arguments expected by the Function object.

Syntax

```
functionName.length
```

`functionName` represents the Function object.

Note

To avoid confusions, it is best to use the arity property.

length (String) – Property of object: String

See also

Returns the value corresponding to the number of characters found in a text string.

Syntax

```
varText.length
```

varText refers to a text string.

lineHeight – Property of object: Style

See also

Specifies the distance between the baselines of two adjacent lines of a block-level element.

Syntax

```
styleObject.lineHeight = {number | length | percentage | normal}
```

Parameters

styleObject represents a Style object.
number is a string evaluating to a size without a unit of measurement. For example, 1.5.
length is a string representing a size followed by a unit of measurement. For example, 10pt.
percentage is a string representing a percentage of the parent element's height. For example, 20%.
normal indicating that the line height is determined automatically by the browser.

linkColor – Property of object: document

See also

This property specifies the color of the document hypertexts.

Syntax

```
document.linkColor
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.linkColor="#FF0000";
</SCRIPT>
Defines the color red for <A HREF="http://www.imaginet.fr/ime/">UNGI  link!
</A>.<BR>
```

links – Property of object: document

See also

This property represents an object array corresponding to Link objects found in a document.

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.linkColor="#FF0000";
</SCRIPT>
<A HREF="http://www.imaginet.fr/ime/">UNGI link</A>.<BR>
<A HREF="http://www.altavista.com/">AltaVista link</A>.<BR>
<A HREF="http://www.netscape.com/">Netscape link</A>.<BR>
List of links found in this document:<BR>
<SCRIPT LANGUAGE="JavaScript">
for (i=0; i < document.links.length; i++)
     {

document.writeln("document.links["+i+"].href="+document.links[i].href+"
<BR>");

}
</SCRIPT>
```

listStyleType – Property of object: Style

See also

Specifies the style of bullet or numbering used for list items.

Syntax

```
styleObject.listStyleType = styleType
```

Parameters

styleObject represents a Style object.
styleType is a string representing the following values: **disc** (normal bullets, black dot), **circle** (empty circles), **square** (black squares), **decimal** (decimal numbers), **lower-roman** (lowercase Roman numbers), **upper-roman** (uppercase Roman numbers), **lower-alpha** (lowercase), **upper-alpha** (uppercase), **none** (no bullets or numbering).

LN2 – Property of object: Math

See also

Represents the natural logarithm of 2, approximately 0.693.

Syntax

```
Math.LN2
```

Attributes

DontEnum, DontDelete, ReadOnly

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.write("LN2 Math.LN2 ="+Math.LN2+" .<BR>");
</SCRIPT>
```

LN10 – Property of object: Math

See also

Represents the natural logarithm of 10, approximately 2.302.

Syntax

```
Math.LN10
```

Attributes

DontEnum, DontDelete, ReadOnly

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.write("LN10 Math.LN10 ="+Math.LN10+" .<BR>");
</SCRIPT>
```

location – Property of object: window

See also

Contains information on the current URL.

locationbar – Property of object: window

See also

Represents the browser window's location bar. The **locationbar** property has a **visible** property. If the value of this last property is **true**, the location bar is visible, if the value is **false**, the bar is hidden.

Example

```
self.locationbar.visible=true
```

Note

Setting this property requires the *UniversalBrowserWrite* privilege.

LOG2E – Property of object: Math

See also

Represents the base 2 logarithm of **E** (approximately 1.442) .

Syntax

```
Math.LOG2E
```

Attributes

DontEnum, DontDelete, ReadOnly

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.write("The base 2 logarithm of  <B>E</B> Math.LOG2E
="+Math.LOG2E+" .<BR>");
</SCRIPT>
```

LOG10E – Property of object: Math

See also

Represents the base 10 logarithm of **E** (approximately 0.434).

Syntax

```
Math.LOG10E
```

Attributes

DontEnum, DontDelete, ReadOnly

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.write("The base 10 logarithm of  <B>E</B> Math.LOG10E
="+Math.LOG10E+" .<BR>");
</SCRIPT>
```

lowsrc – Property of object: Image

See also

This property is a string specifying the URL of a low-resolution version of an image to be displayed in a document. This property represents the **LOWSRC** attribute of the **<IMG>** tag.

Syntax

```
imageName.lowsrc
Parameters
```

imageName is the name of an Image object or an element in an images array.

Example

```
<IMG SRC="birds.jpg" LOWSRC="ungi.gif" NAME="ungi" WIDTH=120 HEIGHT=80
BORDER=4>
<BR>
<SCRIPT LANGUAGE="JavaScript">
document.writeln("The image's lowsrc property is
document.ungi.lowsrc="+document.ungi.lowsrc);
</SCRIPT>
<BR>
```

marginBottom – Property of object: Style

See also

Specifies the minimal distance between the bottom of an HTML element and the top of an adjacent element.

Syntax

```
styleObject.marginBottom = {length | percentage | auto}
```

Parameters

styleObject represents a Style object.
length is a string indicating a number followed by a unit of measurement, for example, 10px.
percentage is a string representing a percentage of the parent element's width, for example, 10%.
auto indicates that the margin is determined automatically by the Web browser.

marginLeft – Property of object: Style

See also

Specifies the minimal distance between the left side of an HTML element and the right side of an adjacent element.8

Syntax

```
styleObject.marginBottom = {length | percentage | auto}
```

Parameters

styleObject represents a Style object.
length is a string indicating a number followed by a unit of measurement, for example, 10px.
percentage is a string representing a percentage of the parent element's width, for example, 10%.
auto indicates that the margin is determined automatically by the Web browser.

marginRight – Property of object: Style

See also

Specifies the minimal distance between the right side of an HTML element and the left side of an adjacent element.

Syntax

```
styleObject.marginBottom = {length | percentage | auto}
```

Parameters

styleObject represents a Style object.
length is a string indicating a number followed by a unit of measurement, for example, 10px.
percentage is a string representing a percentage of the parent element's width, for example, 10%.
auto indicates that the margin is determined automatically by the Web browser.

marginTop – Property of object: Style

See also

Specifies the minimal distance between the top of an HTML element and the bottom of an adjacent element.

Syntax

```
styleObject.marginBottom = {length | percentage | auto}
```

Parameters

styleObject represents a Style object.
length is a string indicating a number followed by a unit of measurement, for example, 10px.
percentage is a string representing a percentage of the parent element's width, for example, 10%.
auto indicates that the margin is determined automatically by the Web browser.

MAX_VALUE – Property of object: Number

See also

Returns the value 1,79E+308.

Syntax

```
number.MAX_VALUE
```

`number` corresponds to the Number object created.

menubar – Property of object: window

See also

Represents the browser window's menu bar. The **menubar** property has a **visible** property. If the value of this last property is **true**, the menu bar is visible, if the value is **false**, the bar is hidden.

Example:

```
self.menubar.visible=true
```

Note

Setting this property requires the *UniversalBrowserWrite* privilege.

method – Property of object: Form

See also

A string specifying how form field input information is sent to the server. It is the same as the **METHOD** attribute of the **<FORM>** element. The property should evaluate to either "get" or "post".

Syntax

```
formName.method
```

formName is the name of a form or an element in a tables array.

Example

```
<FORM NAME=form METHOD=post>
     <INPUT TYPE="password" NAME="passwordField" VALUE="password" SIZE=15>
     <INPUT TYPE="submit">
</FORM>
<BR>
<SCRIPT LANGUAGE="JavaScript">
document.writeln("Submission method of form document.form.method
="+document.form.method);
</SCRIPT>
<BR>
```

mimeTypes – Property of object: navigator

See also

An array of all MIME types supported by the browser. Each element in the array corresponds to a MimeType object.

MIN_VALUE – Property of object: Number

See also

Returns the value 2,22E-308.

Syntax

```
number.MIN_VALUE
```

`number` corresponds to the Number object created.

modifiers – Property of object: event

See also

String specifying the modifier keys associated with a mouse or key event. Possible values are: ALT_MASK, CONTROL_MASK, SHIFT_MASK, and META_MASK.

Note

The use of this property requires the *UniversalBrowserWrite* privilege.

multiline – Property of object: RegExp

See also

This property determines whether or not to search in strings across one or multiple lines. $* is another name for the same property.

Note

multiline is static, it is not a property of an individual regular expression object. This property is always used as follows: **RegExp.multiline**.

name – Property of objects: Anchor, Button, Checkbox, FileUpload, Form, Hidden, Image, Layer, Password, Plugin, Radio, Reset, Select, Submit, Text, Textarea and window

See also

This property represents a text string specifying the name of an object.

Syntax

```
objectName.name
frameReference.name
frameReference.frames.name
radioName[index].name
selectName.options.name
windowReference.name
windowReference.frames.name
fileUploadName.name
imageName.name
navigator.plugins[index].name
navigator.plugins[PluginName].name
```

objectName is the value of the NAME attribute of an object mentioned below or an element in an arrays table.
frameReference is a valid way to reference a frame, as it is described in the Frame object.
radioName is the value of attribute NAME of a Radio object.
selectName is the value of attribute NAME of the Select object or an element in an elements array.
windowReference is a valid way to reference a window, as it is described in the window object.
fileUploadName is the value of attribute NAME of the FileUpload object or an element in an elements array.
imageName is the value of attribute NAME of the Image object or an element in an elements array.
index is an integer representing an embedded module (*plugin*) in a document.
PluginName represents a text string containing the name of a Plugin object.

Example

```
<IMG SRC="birds.jpg" LOWSRC="ungi.gif" NAME="ungi" WIDTH=120 HEIGHT=80
BORDER=4>
<BR>
<SCRIPT LANGUAGE="JavaScript">
document.writeln("NAME property of the first image document.images[0].name
="+document.images[0].name);
</SCRIPT>
<BR>
```

NaN – Property

See also

This property is a top-level property and is not associated with any object, it indicates a value that is not associated with any number (Not-A-Number).

NaN (Number) – Property of object: Number

See also

Specifies a numeric expression that is not a numeric value.

Syntax

```
number.NaN
```

number corresponds to the Number object created.

NEGATIVE_INFINITY – Property of object: Number

See also

Returns a special numeric value representing negative infinity (`-Infinity`).

Syntax

```
number.NEGATIVE_INFINITY
```

`number` corresponds to the Number object created.

next – Property of object: History

See also

Represents a string specifying the complete URL of the next history entry. This property reflects the URL that would be used if the user chose Forward from the browser menu.

Note

The use of this property requires the *UniversalBrowserWrite* privilege.

netscape – Property of object: Packages

See also

Use the netscape property to access any class in the *netscape* (*netscape package*) from within JavaScript.

number – Property of object: Error

See also

Returns or defines a numeric value corresponding to an error.

Syntax

```
errorName.number[=errorNumber]]
```

offscreenBuffering – Property of object: window

See also

Specifies whether window updates are performed in an offscreen (*buffer*) or not, before updating the screen.
The **yes** value authorizes the user of the buffer memory and the **no** value updates the window immediately without using the buffer memory.

opener – Property of object: window

See also

This property specifies the window name of the calling document when a window is opened using the open method . Using this method, the newly created window can use the variables and methods of the parent windows.

Syntax

```
window.opener
```

Example

Source in the first window:

```
<SCRIPT LANGUAGE="JavaScript">
this.name="principal"
window2=open("window2.htm","Subscribe","scrollbars=no,width=250,height=400"
);
</SCRIPT>
```

Source in the second window 'window2.htm':

```
This is a second window.<BR>
<SCRIPT LANGUAGE="JavaScript">
document.writeln("Window opened by
window.opener.name="+window.opener.name);
</SCRIPT>
```

options – Property of object: Select

See also

This property represents an array corresponding to the options (**<OPTION>** element) of a Select object found in a document. Refer to the Select object.

Example

```
<FORM NAME=form>
<SELECT NAME=list>
     <OPTION NAME="sel1">selected1
     <OPTION NAME="sel2">selected2
     <OPTION NAME="sel3">selected3
</SELECT>
</FORM>
Contents of list:<BR>
<SCRIPT LANGUAGE="JavaScript">

for (i=0; i < document.form.list.options.length; i++)
     {
document.writeln("document.form.list.options["+i+"].text="+document.form.li
st.options[i].text+" <BR>");
     }
</SCRIPT>
<BR>
```

outerHeight – Property of object: window

See also

Specifies the window's vertical external size in pixels. To create a window smaller than 100 x 100 pixels, set property in a signed script.

outerWidth – Property of object: window

See also

Specifies the window's horizontal external size in pixels. To create a window smaller than 100 x 100 pixels, set property in a signed script.

paddingBottom – Property of object: Style

See also

indicates the spacing value that must be inserted between the bottom of an element and its content (text, image).

Syntax

```
styleObject.paddingBottom = {length | percentage}
```

Parameters

styleObject represents a Style object.
length is a string indicating a number followed by a unit of measurement, for example, 10px.
percentage is a string evaluating to a percentage of the parent element's width, for example 10%.

paddingLeft – Property of object: Style

See also

Specifies the spacing value that must be inserted between the left side of an element and its content (text, image).

Syntax

```
styleObject.paddingBottom = {length | percentage}
```

Parameters

styleObject represents a Style object.
length is a string indicating a number followed by a unit of measurement, for example, 10px.
percentage is a string evaluating to a percentage of the parent element's width, for example 10%.

paddingRight – Property of object: Style

See also

Specifies the spacing value that must be inserted between the right side of an element and its content (text, image).

Syntax

```
styleObject.paddingBottom = {length | percentage}
```

Parameters

styleObject represents a Style object.
length is a string indicating a number followed by a unit of measurement, for example, 10px.
percentage is a string evaluating to a percentage of the parent element's width, for example 10%.

paddingTop – Property of object: Style

See also

Indicates the spacing value that must be inserted between the top of an element and its content (text, image).

Syntax

```
styleObject.paddingBottom = {length | percentage}
```

Parameters

styleObject represents a Style object.
length is a string indicating a number followed by a unit of measurement, for example, 10px.
percentage is a string evaluating to a percentage of the parent element's width, for example 10%.

pageX – Property of objects: event and Layer

See also

Number specifying the horizontal position relative to the *layer* page, for the object Layer or the mouse pointer for the event object.

With regards to the event object, the use of this property requires *UniversalBrowserWrite* privileges.

pageXOffset – Property of object: window

See also

Provides the current x-position (x-coordinate) , in pixels, of a window. The coordinate refers to the top left corner of the window as point of reference. This property is useful to determine the current position of the window before using the methods scrollBy or scrollTo.

pageY – Property of objects: event and Layer

See also

Number specifying the vertical position relative to the *layer* page, for the object Layer or the mouse pointer for the event object.

With regards to the event object, the use of this property requires *UniversalBrowserWrite* privileges.

pageYOffset – Property of object: window

See also

Provides the current y-position (y-coordinate) , in pixels, of a window. The coordinate refers to the top left corner of the window as point of reference. This property is useful to determine the current position of the window before using the methods scrollBy or scrollTo.

parent – Property of object: window

See also

This property makes reference to the window or frame (**<FRAMESET>** element) containing the current frame. Used as read-only.

Syntax

```
parent.propertyName
parent.methodName
parent.frameName
parent.frames[index]
```

Parameters

propertyName is the defaultStatus, status, length, name or parent property when the call refers to a window object.
propertyName is the length, name or parent property when the call refers to a Frame object.
methodName is any method associated with a window object.
frameName and **frames[index]** are ways to reference a frame.

Examples

```
<SCRIPT LANGUAGE="JavaScript">
document.write("The name of the parent window is window.parent.name =
"+window.parent.name+"<BR>");
</SCRIPT>
```

parentLayer – Property of object: Layer

See also

Represents the Layer object containing the current *layer* or the window object if the current *layer* is not nested in another Layer.

pathname – Property of objects: Link and Location

See also

This property represents the portion of the URL indicating the name of the preceding file in the access path.

Syntax

```
links[index].pathname
location.pathname
areaName.pathname
```

index is an integer representing the Link object.
areaName is the value of attribute **NAME of an** Area object.

Example

```
<A HREF="http://www.labas.com:2000/directory/file.htm#interesting">This is
a link</A>
<BR>
<SCRIPT LANGUAGE="JavaScript">
document.write("The path of the file specified in the links  [0]
is:"+document.links[0].pathname);
</SCRIPT>
```

personalbar – Property of object: window

See also

Represents the browser window's personal bar which displays folder buttons with the user's personal bookmarks. The **personalbar** property has a **visible** property. If the value of this last property is **true**, the personalbar is visible, if the value is **false**, the bar is hidden.

Example:

```
self.personalbar.visible=true
```

Note

Setting this property requires the *UniversalBrowserWrite* privilege.

PI – Property of object: Math

See also

The ratio of the circumference of a circle to its diameter (approximately 3.14159).

Syntax

```
Math.PI
```

Attributes

DontEnum, DontDelete, ReadOnly

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.write("Pi Math.PI ="+Math.PI+" .<BR>");
</SCRIPT>
```

pixelDepth – Property of object: screen

See also

Screen resolution: number of colors expressed in bits per pixel.

platform – Property of object: navigator

See also

Indicates the machine type for which the browser was compiled. Possible values are: Win32, Win16, Mac68k, MacPPC, and various Unix platforms, Linux.

plugins – Property of objects: document and navigator

See also

Represents an array corresponding to the embedded modules (*plugins*) found in the document, according to their source order.
You can reference (*plugin*s) modules in your code by calling the **`plugins`** property array.

For example, in a document containing two embedded modules, you can reference them as follows:

```
document.plugins[0]
document.plugins[1]
```

port – Property of objects: Link and Location

See also

This property specifies the communications port that the server uses.

Syntax

```
links[index].port
location.port
areaName.port
```

index is an integer representing the Link object.
areaName is the value of attribute **NAME of an** Area object.

Example

```
<A HREF="http://www.labas.com:2000/directory/file.htm#interesting">This is
link</A>
<BR>
<SCRIPT LANGUAGE="JavaScript">
document.write("The port associated to the link links[0]
is:"+document.links[0].port);
</SCRIPT>
```

POSITIVE_INFINITY – Property of object: Number

See also

Returns a positive infinite value.

Syntax

```
number.POSITIVE_INFINITY
```

number corresponds to the Number object created.

previous – Property of object: History

See also

Represents a string specifying the complete URL of the preceding history entry. This property reflects the URL that would be used if the user chose BACK from the browser menu.

Note

Setting this property requires the *UniversalBrowserWrite* privilege.

protocol – Property of objects: Link and Location

See also

This property is a string specifying a URL portion that indicates the protocol used (for example, http:, ftp:, telnet:).

Syntax

```
links[index].protocol
location.protocol
areaName.protocol
```

index is an integer representing the Link object.
areaName is the value of attribute **NAME of an** Area object.

Example

```
<A HREF="http://www.labas.com:2000/directory/file.htm#interesting">This is
a link</A>
<BR>
<SCRIPT LANGUAGE="JavaScript">
document.write("The protocol of the file specified in the links is [0]
is:"+document.links[0].protocol);
</SCRIPT>
```

prototype – Property of objects: Array, Boolean, Date, Function, Number, Object, RegExp and String

See also

This property defines a property that is shared by all the objects in a particular class. You must use this property to add properties to objects created with the new operator.

Syntax

```
objectType.prototype.propertyName = value
```

objectType is the name of the constructor specified for the object type.
propertyName is the name of the property that will be created.
value is the initial value of the property for all specified **objectType** objects.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Adding a property to the Date object:
Date.prototype.Millenium;
now = new Date();
now.millenium = "third";
document.writeln("millennium: "+now.millenium+"<BR>");
</SCRIPT>
```

referrer – Property of object: document

See also

Specifies the URL of the calling document when a user activates a link.

Syntax

```
document.referrer
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.write("This document was opened from a link on URL
document.referrer = "+document.referrer+"<BR>");
</SCRIPT>
```

rightContext – Property of object: RegExp

See also

The substring following the most recent match. $' is another name for the same property.

Note

rightContext is static, it is not a property of an individual regular expression object. This property is always used as follows: **RegExp. rightContext**.

screenX – Property of objects: event and window

See also

Number specifying the cursor's horizontal position relative to the screen. The value is expressed in pixels.

Note
The use of this property requires the *UniversalBrowserWrite* privilege.

screenY – Property of objects: event and window

See also

Number specifying the cursor's vertical position relative to the screen. The value is expressed in pixels.

Note
The use of this property requires the *UniversalBrowserWrite* privilege.

scrollbars – Property of object: window

See also

Represents the window's vertical and horizontal scroll bars. The **scrollbar** property has a **visible** property. If the value is set to **true** the scrollbars are visible, if the value is set to **false**, the bars are hidden.

Example:

```
self.scrollbars.visible=true
```

Note

Setting this property requires the *UniversalBrowserWrite* privilege.

search – Property of objects: Link and Location

See also

A string beginning with a question mark that specifies any query information in the URL. The string has a question mark at the beginning, followed by a request.

Syntax

```
links[index].search
location.search
areaName.search
```

index is an integer representing the Link object.
areaName is the value of attribute **NAME of an** Area object.

Example

```
<A HREF="http://www.labas.com:2000/directory/file.cgi?
this_is_the_request">This is a link</A>
<BR>
<SCRIPT LANGUAGE="JavaScript">
document.write("The request specified in the link links[0]
is:"+document.links[0].search);
</SCRIPT>
```

selected – Property of object: Option

See also

Boolean value indicating an option selection state of a Select or Option object (whether the option is selected or not).

Syntax

```
selectName.options[index].selected
optionName.selected
```

selectName is the value of attribute NAME of the Select object or an element in an elements array.
index is an integer representing an option of a Select object.
optionName is the name of a Select option created with the Option() constructor.

Example

```
<FORM NAME=form>
<SELECT NAME=list MULTIPLE SIZE=6>
     <OPTION SELECTED>selected1
     <OPTION SELECTED>selected2
     <OPTION>selected3
     <OPTION SELECTED>selected4
     <OPTION SELECTED>selected5
     <OPTION>selected6
</SELECT>
<BR>
<INPUT TYPE="button" VALUE="Show selections" onClick="selections()");
<BR>
List of elements selected:<BR>
<TEXTAREA NAME="box" ROWS=5 COLS=40></TEXTAREA>
</FORM>

<SCRIPT LANGUAGE="JavaScript">

function selections()
{
var list = "";
for (i=0; i < document.form.list.length; i++)
     {
     if(document.form.list[i].selected==true)
           {
           list +=document.form.list[i].text+" is selected";
           }
     }
document.form.box.value=list;
}
</SCRIPT>
```

selectedIndex – Property of object: Select

See also

This property is an integer specifying the index of the selected option in a Select object.

Syntax

```
selectName.selectedIndex
selectName.options.selectedIndex
```

selectName is the value of attribute NAME of the Select object or an element in an elements array.

Example

```
<FORM NAME=form>
<SELECT NAME=list>
     <OPTION>selection1
     <OPTION>selection2
     <OPTION>selection3
     <OPTION>selection4
     <OPTION>selection5
     <OPTION>selection6
</SELECT>
<BR>
<INPUT TYPE="button" VALUE="Show selection" onClick="selectionindex()");
<BR>
Index of selected elements;:<BR>
<TEXTAREA NAME="box" ROWS=5 COLS=40></TEXTAREA>
</FORM>

<SCRIPT LANGUAGE="JavaScript">
function selectionindex()
{
document.form.box.value=document.form.list.selectedIndex;
}
</SCRIPT>
```

self – Property of object: window

See also

The self property is a synonym for the current window.

Syntax

```
self.propertyName
self.methodName
```

propertyName is the defaultStatus, status, length, name or parent property when the call refers to a window object.
propertyName is the length, name or parent property when the call refers to a Frame object.
methodName is any method associated with a window object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
this.name="principal"
document.writeln("'self' is a synonym of 'window':<BR>");
document.writeln("Name returned by window.name="+window.name+"<BR>");
document.writeln("Name returned by  self.name="+self.name+"<BR>");
</SCRIPT>
```

siblingAbove – Property of object: Layer

See also

Represents the *layer* above the current *layer* according to the **z-order**, among all *layers* that share the same parent *layer*. **Null** if the layer has no sibling above.

siblingBelow – Property of object: Layer

See also

Represents the *layer* below the current *layer* according to the **z-order**, among all *layers* that share the same parent *layer*. **Null** if the layer has no sibling below.

source – Property of objects: RegExp and Regular Expression

See also

A read-only property that contains the text of the pattern, excluding the forward slash "**/**" and the "**g**" or "**i**" flags. You cannot change this property directly. However, calling the compile method changes the value of this property.

SQRT1_2 – Property of object: Math

See also

Represents the square root of ½ (approximately 0.707), equivalent to 1 over the square root of 2.

Syntax

```
Math.SQRT1_2
```

Attributes

```
DontEnum, DontDelete, ReadOnly
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.write("Square root of 2 Math.SQRT1_2 ="+Math.SQRT1_2+" .<BR>");
</SCRIPT>
```

SQRT2 – Property of object: Math

See also

Represents the square root of 2 (approximately 1.414).

Syntax

```
Math.SQRT2
```

Attributes

`DontEnum`, `DontDelete`, `ReadOnly`

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.write("Square root of 2 Math.SQRT2 ="+Math.SQRT2+" .<BR>");
</SCRIPT>
```

src – Property of objects: Image and Layer

See also

This property represents a string specifying the URL of an image to be displayed in a document. It represents the **SRC** attribute of the **<IMG>** tag.

Syntax

```
imageName.src
```

imageName is the name of an Image object or an element in an images array.

Example

```
<IMG SRC="ungi.gif" NAME="ungi" WIDTH=120 HEIGHT=80 BORDER=4>
<BR>
<SCRIPT LANGUAGE="JavaScript">
document.writeln("The src property of the image is
document.ungi.src="+document.ungi.src);
</SCRIPT>
<BR>
```

status – Property of object: window

See also

This property specifies a priority or a message in the status bar, such as a URL address, when the user moves the mouse pointer over a hyperlink.

Syntax

```
windowReference.status
```

windowReference is a valid way to reference a window, as it is described in the window object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
function gozone()
{
this.status="You are in the picture!";
// the function should return true when it is called by onMouseOver
return "true";
}
function outzone()
{
this.status="You are not in the picture!";
}
</SCRIPT>

<IMG SRC="ungi.gif" WIDTH=120 HEIGHT=80 BORDER=0 USEMAP="#ungimap">
<MAP NAME="ungimap">
  <AREA NAME="zone0" SHAPE="RECT" COORDS="0,0,118,78" TARGET="_top"
HREF="http:\\www.galaxy.spc\sun\terre\france.htm#normandy"
onMouseOver="gozone()" onMouseOut="outzone()">
</MAP>
```

statusbar – Property of object: window

See also

Represents the status bar at the bottom of the window. The **statusbar** property has a **visible** property. If the value of this last property is **true**, the location bar is visible, if the value is **false**, the bar is hidden.

Example

```
self.statusbar.visible=true
```

Note

Setting this property requires the *UniversalBrowserWrite* privilege.

suffixes – Property of object: MimeType

See also

A list of possible file extensions for a MIME type. For example, the extensions for the MIME types "audio/x-midi" are: mid, midi.

sun – Property of object: Packages

See also

Use this property to access any type of *sun package* within JavaScript.

tags – Property of object: document

See also

Allows specifying a style for any HTML tag.

Example:

```
<STYLE TYPE="text/javascript">
   tags.P.color="green"
</STYLE>
```

target – Property of objects: event, Form and Link

See also

This property is a string that operates the same as the **TARGET** attribute of the **<FORM>** and **<A>** elements. In case of a form, a string specifying the name of the window that responses go to after a form has been submitted In case of a hypertext link, it specifies the name of the window that will display the contents of the document called.

Syntax

```
formName.target
links[index].target
areaName.target
```

formName is the name of a form or an element in a tables array.
index is an integer representing the Link object.
areaName is the value of attribute **NAME of an** Area object.

Example

```
<A HREF="http://www.labas.com:2000/directory/file.cgi?ThisIsTheRequest"
target="_top">This is a link</A>
<BR>
<SCRIPT LANGUAGE="JavaScript">
document.write("The URL specified by link links[0] will be displayed
in:"+document.links[0].target);
</SCRIPT>
```

text – Property of objects: Link and Option

See also

This property specifies the text following an **<OPTION>** element in a **select** field. You can change the text.

Syntax:

```
selectName.options[index].text
optionName.text
```

selectName is the value of attribute NAME of the Select object or an element in an elements array.
index is an integer representing an option of a Select object.
optionName is the name of a Select option created with the Option() constructor.

Example

```
<FORM NAME=form>
<SELECT NAME=list>
     <OPTION>selection1
     <OPTION>selection2
     <OPTION>selection3
     <OPTION>selection4
     <OPTION>selection5
     <OPTION>selection6
</SELECT>
<BR>
<INPUT TYPE="text" NAME=textfield1 MAXLENGTH=10 SIZE=10 VALUE="selection1">
<INPUT TYPE="text" NAME=textfield2 MAXLENGTH=10 SIZE=10 VALUE="selection2">
<INPUT TYPE="text" NAME=textfield3 MAXLENGTH=10 SIZE=10 VALUE="selection3">
<INPUT TYPE="text" NAME=textfield4 MAXLENGTH=10 SIZE=10 VALUE="selection4">
<INPUT TYPE="text" NAME=textfield5 MAXLENGTH=10 SIZE=10 VALUE="selection5">
<INPUT TYPE="text" NAME=textfield6 MAXLENGTH=10 SIZE=10 VALUE="selection6">
<INPUT TYPE="button" VALUE="Change list" onClick="selectionindex()");
<BR>
</FORM>

<SCRIPT LANGUAGE="JavaScript">
function selectionindex()
{
document.form.list[0].text=document.form.textfield1.value;
document.form.list[1].text=document.form.textfield2.value;

document.form.list[2].text=document.form.textfield3.value;
document.form.list[3].text=document.form.textfield4.value;
document.form.list[4].text=document.form.textfield5.value;
document.form.list[5].text=document.form.textfield6.value;
}
</SCRIPT>
```

text (Anchor) – Property of object: Anchor

See also

Represents a text string where the text is a link for an anchor.

textAlign – Property of object: Style

See also

Specifies the alignment of an HTML block-level text element.

Syntax

```
styleObject.textAlign = alignment
```

Parameters

styleObject represents a Style object.
alignment is a string specifying one of the following values: **left** (left aligned), **right** (right aligned), **center** (centered), **justify** (justified).

Note

Not to be confused with the align property which specifies the alignment of an HTML element within its parent element.

textDecoration – Property of object: Style

See also

Specifies special effects added to an HTML text element.

Syntax

```
styleObject.textDecoration = decoration
```

Parameters

styleObject represents a Style object.
decoration is a string specifying one of the following values: **none** (no special effect), **underline**, **line-through**, **blink**.

textIndent – Property of object: Style

See also

Specifies the indentation value appearing before the first line of a block-level HTML text element.

Syntax

```
styleObject.textIndent = {length | percentage}
```

Parameters

styleObject represents a Style object.
length is a string indicating a number followed by a unit of measurement, for example, 10px.
percentage is a string evaluating to a percentage of the parent element's width, for example 10%.

textTransform – Property of object: Style

See also

Transforms the case of an HTML text element..

Syntax

```
styleObject.textTransform = transformation
```

Parameters

styleObject represents a Style object.
transformation is a string representing one of the following values: **none** (no transformation), **capitalize** (the first letter of each word is capitalized), **uppercase** (all the words are in uppercase), **lowercase** (all the words are in lowercase).

title – Property of object: document

See also

This property represents the title of a document, much the same as the **<TITLE>** tag. This property is read-only.

Syntax

```
document.title
```

Example

```
<HTML>
<HEAD>
<title>Methods date</title>
</HEAD>

<BODY>

<SCRIPT LANGUAGE="JavaScript">
document.write("The document title is document.title
="+document.title+"<BR>");
</SCRIPT>

</BODY>
</HTML>
```

toolbar – Property of object: window

See also

Represents the window's standard toolbar. The **toolbar** property has a **visible** property. If the value of this last property is **true**, the toolbar is visible, if the value is **false**, the bar is hidden.

Example:

```
self.toolbar.visible=true
```

Note

Setting this property requires the *UniversalBrowserWrite* privilege.

top – Property of objects: Layer and window

See also

Represents the vertical position of the left edge of a *layer* with relation to its parent *layer* . The value is expressed in pixels.

type – Property of objects: Button, Checkbox, event, FileUpload, Hidden, MimeType, Password, Radio, Reset, Select, Submit, Text and Textarea

See also

This property specifies the type of elements of a form. Type is the value of the **TYPE** attribute found in **<INPUT>**, **<SELECT>** and **<TEXTAREA>** elements.

Syntax

```
objectName.type
mimeTypes[index].type
```

objectName is the value of attribute NAME of an element in a form or an element in an elements array.

Example

```
<FORM NAME="form">
<INPUT TYPE="text" MAXLENGTH=40 SIZE=40 VALUE="text field">
<INPUT TYPE="button" VALUE="Pressing does nothing">
</FORM>

<SCRIPT LANGUAGE="JavaScript">
for (i=0; i < document.form.elements.length; i++)
     {
     document.write("Form element "+i+" is type
"+document.form.elements[i].type+"<BR>")
     }
</SCRIPT>
```

undefined – Property

Represents the value *undefined*.
A variable that has not been assigned a value is an *undefined* type. A method or instructions returns *undefined* if no value has been assigned to the variable evaluated.

You can use the *undefined* property to determine if a variable has been assigned a value.

Syntax

```
undefined
```

URL – Property of object: document

See also

This property specifies the complete URL of a document. It is read-only..

Syntax

```
document.URL
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.write("The current URL is document.URL = " + document.URL);
</SCRIPT>
```

userAgent – Property of object: navigator

See also

This property is a text string representing the value of the user-agent sent in the HTTP protocol from client to server. This value is used by the server to identify the user. Used as read-only.

Syntax

```
navigator.userAgent
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.write("Identification of client program navigator.userAgent =
"+navigator.userAgent);
</SCRIPT>
```

value – Property of objects: Button, Checkbox, FileUpload, Hidden, Option, Password, Radio, Reset, Submit, Text and Textarea

See also

This property is a text string representing the **VALUE** attribute of the corresponding object.

Syntax

```
objectName.value
radioName[index].value
selectName.options.[index].value
fileUploadName.value
optionName.value
```

objectName is the value of the NAME attribute of the following objects: Hidden, Password, Text, Textarea, Button, Reset, Submit or Checkbox, or an element in an elements array.
radioName is the value of attribute NAME of a Radio object.
selectName is the value of attribute NAME of the Select object or an element in an elements array.
index is an integer representing a radio button of a Radio object or an option of the Select object.
fileUploadName is the value of the NAME attribute of the FileUpload object or an element in an elements array.
optionName is the name of a Select option created with the Option() constructor.

Example

```
<FORM NAME="form">
<INPUT TYPE="text" NAME=enter LENGTH=40 SIZE=40 VALUE="Enter status">
<INPUT TYPE="button" VALUE="change status" onClick="changestatus()">
</FORM>

<SCRIPT LANGUAGE="JavaScript">
function changestatus()
{
window.status=document.form.input.value;
}
</SCRIPT>
```

visibility – Property of object: Layer

See also

Specifies whether the *layer* is visible or not.
The **`show`** value indicates that the *layer* should be visible, the **`hide`** value indicates that the *layer* should be invisible and the **`inherit`** value indicates that the current *layer* inherits the *layer* parent value.

vlinkColor – Property of object: document

See also

This property is a text string that specifies the color of visited links.

Syntax

```
document.vlinkColor
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.vlinkColor="#FF0000";
document.write("Visited links will be shown in RED <BR>");
</SCRIPT>

<A HREF="http://www.imaginet.fr/ime/">Ungi!</A>
```

vspace – Property of object: Image

See also

This property is a text string indicating the vertical space value that must be inserted between an image and the surrounding text. The value is expressed in pixels. It is the same as the **HSPACE** attribute of the **<IMG>** element. This property is read-only.

Syntax

```
imageName.vspace
```

imageName is the name of an Image object or an element in an images array.

Example

```
<IMG SRC="ungi.gif" NAME="ungi" WIDTH=120 HEIGHT=80 BORDER=4 VSPACE=35>
<BR>
Logo UNGI !
<BR>
<SCRIPT LANGUAGE="JavaScript">
document.writeln("Vertical spacing around image: "+document.ungi.vspace);
</SCRIPT>
<BR>
```

which – Property of object: event

See also

Value specifying either the mouse button that was pressed or the ASCII value of a pressed key.
For a mouse, 1 is the left button, 2 is the middle button, and 3 is the right button.

Note
Setting this property requires the *UniversalBrowserWrite* privilege.

whiteSpace – Property of object: Style

See also

This property is used to indicate how white-space inside the element is handled.

Syntax

```
styleObject.whiteSpace = {normal | pre}
```

Parameters

styleObject represents a Style object.
normal This value tells the browser software to merge space sequences (several consecutive spaces) into one space.
pre This value tells the browser software to keep the space sequences intact (several consecutive spaces).

width – Property of objects: document, event, Image, screen and Style

See also

This property is a text string specifying the width in pixels of an image border or as a percentage. It represents the **WIDTH** attribute of the **<IMG>** tag. The **width** property must be used as read-only.

Syntax

```
imageName.width
```

imageName is the name of an Image object or an element in an images array.

Example

```
<IMG SRC="ungi.gif" NAME="ungi" WIDTH=120 HEIGHT=80 BORDER=4>
<BR>
Logo UNGI !
<BR>
<SCRIPT LANGUAGE="JavaScript">
document.writeln("Image width: "+document.ungi.width);
</SCRIPT>
<BR>
```

window – Property of objects: Layer and window

See also

This property represents the window object or the Frame object containing the current *layer*.

Syntax

```
window.propertyName
window.methodName
```

propertyName is the defaultStatus, status, length or name property when the call references a window object.
propertyName is the length or name property when the call references a Frame object.
methodName is any method associated with a window object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
window.defaultStatus = "Status bar";
<SCRIPT LANGUAGE="JavaScript">
```

x – Property of objects: Anchor, event, Layer and Link

See also

For the Layer object, this property represents the horizontal position of the left edge of the *layer* relative to its parent *layer* parent. The value is expressed in pixels.

For the Anchor object, this property represents the horizontal position of the anchor's left edge, in pixels, relative to the left edge of the document. The value is expressed in pixels.

For the event object, this property represents a number specifying the width of the object when used with the onResize event or the pointer's horizontal position, expressed in pixels, relative to the *layer* from which the event arose.

y – Property of objects: Anchor, event, Layer and Link

See also

For the Layer object, this property represents the vertical position of the left edge of the *layer* relative to its parent *layer* parent. The value is expressed in pixels.

For the Anchor object, this property represents the vertical position of the anchor's left edge, in pixels, relative to the left edge of the document. The value is expressed in pixels.

For the event object, this property represents a number specifying the height of the object when used with the onResize event or the pointer's vertical position, expressed in pixels, relative to the *layer* from which the event arose.

zIndex – Property of object: Layer

See also

The relative order of the current *layer* in the stack (stack index). The value is a number. The *layer* having the highest **`zIndex`** is on top, while the layer having the smallest z-index lies below the whole stack.

AtEndOfLine - Property of object: TextStream

See also

This property is used to determine if the file pointer is located immediately before the end of line returns in a *TextStream* file. If it is, the it returns a **true** value, if it is not, then it returns a **false** value. Read-only.

Syntax

```
object.AtEndOfLine
```

The **object** argument corresponds to the name of the **TextStream** object.

Example

```
function GetALine(filespec)
{
  var fso, a, s, ForReading;
  ForReading = 1, s = "";
  fso = new ActiveXObject("Scripting.FileSystemObject");
  a = fso.OpenTextFile(filespec, ForReading, false);
  while (!a.AtEndOfLine)
  {
    s += a.Read(1);
  }
  a.Close( );
  return(s);
}
```

AtEndOfStream - Property of object: TextStream

See also

This property is used to determine if the file pointer is found at the end of a *TextStream* file. If it is, then it returns a **true** value, if it is not, then it returns a **false** value. Read-only.

Syntax

```
object.AtEndOfStream
```

The **object** argument corresponds to the name of the **TextStream** object.

Example

```
function GetALine(filespec)
{
  var fso, f, s, ForReading;
  ForReading = 1, s = "";
  fso = new ActiveXObject("Scripting.FileSystemObject");
  f = fso.OpenTextFile(filespec, ForReading, false);
  while (!f.AtEndOfStream)
    s += f.ReadLine( );
  f.Close( );
  return(s);
}
```

Attributes - Property of objects: File and Folder

See also

This property returns or defines the attributes of files or folders. Read/write or write only, depending on the attribute.

Syntax

```
object.Attributes [= newattributes]
```

object refers here to the name of a **File** object or a **Folder** object.

newattributes optional. If used, **newattributes** indicates the new attribute value of the specified **object** element.

Parameters

The **newattributes** argument takes one or several of the following values:

Constant	Value	Description
Normal	0	Normal file. No attribute defined.
ReadOnly	1	Read-only file.
Hidden	2	Hidden file.
System	4	System file.
Volume	8	Name of volume drive. The attribute is read/write.
Directory	16	Folder (*directory*). The attribute is read-only.
Archive	32	The file was modified after it was last saved. The attribute is read/write.
Alias	64	Link or shortcut. The attribute is read-only.
Compressed	128	Compressed file. The attribute is read-only.

AvailableSpace - Property of object: Drive

See also

This property returns a value indicating the amount of available space for the user on the specified drive (on the server or on the network drive).

Syntax

```
object.AvailableSpace
```

object corresponds here to a **Drive** **object.**

Note:

The value returned by the **AvailableSpace** property is usually the same as the **FreeSpace** property. However, differences could arise depending on systems that manage quotas.

Example:

```
function ShowAvailableSpace(drvPath)
{
  var fso, d, s;
  fso = new ActiveXObject("Scripting.FileSystemObject");
  d = fso.GetDrive(fso.GetDriveName(drvPath));
  s = "Drive" + drvPath.toUpperCase() + " - ";
  s += d.VolumeName + "<br>";
  s += "Space available on disk: " + d.AvailableSpace/1024 + "
Kilobytes";
  return(s);
}
```

Column - Property of object: TextStream

See also

Column is a read-only property and returns a column number corresponding to the position of the current character in a **TextStream** file.

Syntax

```
object.Column
```

object refers here to the name of a **TextStream** object.

Note:

After entering a new line character but before entering another character, **Column** is equal to 1. Used to create objects by means of numeric constants.

CompareMode - Property of object: Dictionary

The **CompareMode** property defines and returns a comparison mode of string keys in a **Dictionary** object.

Syntax

```
object.CompareMode[ = compare]
```

The **CompareMode** property is made up of the following elements:

object	refers to the name of a **Dictionary** object.
compare	optional. When specified, **compare** is a value representing the compare mode. Possible values: **0** (Binary), **1** (Text), **2** (Database). Values ***higher than 2*** can be used to make reference to comparisons using specific local ID (LCID).

Note:

An error occurs when the comparison mode of a **Dictionary** object already containing data is modified.

Count

- Property of object: of Dictionary and of Drives, Files and Folders collections.

See also

The **Count** property returns the number of elements in a collection or in a **Dictionary** object. Read-only.

Syntax

```
object.Count
```

object corresponds to the name of one of the elements in the **Application** list.

Example

```
function CountDemo()
{
  var a, d, i, s;                  // Creation of variables.
  d = new ActiveXObject("Scripting.Dictionary");
  d.Add ("a", "Argentina");          // Add keys and elements
  d.Add ("b", "Belgium");
  d.Add ("c", "Canada");
  a = (new VBArray(d.Keys()));     // Obtaining keys.
  s = "";
  for (i = 0; i < d.Count; i++)    //Dictionary run.
  {
    s += a.getItem(i) + " - " + d(a.getItem(i)) + "<br>";
  }
  return(s);                       // Send results.
}
```

DateCreated - Property of objects: File and Folder

See also

The **DateCreated** property returns a value representing the date and hour the file or folder was created. Read-only.

Syntax

```
object.DateCreated
```

object must correspond to a **File** or **Folder** object.

Example:

```
function ShowFileInfo(filespec)
{
  var fso, f, s;
  fso = new ActiveXObject("Scripting.FileSystemObject");
  f = fso.GetFile(filespec);
  s = "File or folder created on: " + f.DateCreated;
  return(s);
}
```

DateLastAccessed - Property of objects: File and Folder

See also

The **DateLastAccessed** property returns a value representing the date and time the file or folder was last accessed.

Syntax

```
object.DateLastAccessed
```

object must correspond to a **File** or **Folder** object.

Example:

```
function ShowFileAccessInfo(filespec)
{
  var fso, f, s;
  fso = new ActiveXObject("Scripting.FileSystemObject");
  f = fso.GetFile(filespec);
  s = filespec.toUpperCase() + "<br>";
  s += "File or folder created on: " + f.DateCreated + "<br>";
  s += "Last accessed on: " + f.DateLastAccessed +
"<br>";
  s += "Last modified on: " + f.DateLastModified;
  return(s);
}
```

DateLastModified - Property of objects: File and Folder

See also

The **DateLastModified** property returns a value representing the date and time the file or folder was last modified. Read-only.

Syntax

```
object.DateLastModified
```

object must correspond to a **File** or **Folder** object.

Example

```
function ShowFileAccessInfo(filespec)
{
  var fso, f, s;
  fso = new ActiveXObject("Scripting.FileSystemObject");
  f = fso.GetFile(filespec);
  s = filespec.toUpperCase() + "<br>";
  s += "File created on: " + f.DateCreated + "<br>";
  s += "Last accessed on: " + f.DateLastAccessed + "<br>";
  s += "Last modified on: " + f.DateLastModified;
  return(s);
}
```

Drive  - Property of objects: File and Folder

See also

The **Drive** property returns a value representing the drive letter on which the file or folder is located. Read-only.

Syntax

```
object.Drive
```

object must correspond to a **File** or **Folder** object.

DriveLetter - Property of object: Drive

See also

This property returns the letter identifying the physical or network drive. Read-only.

Syntax

```
object.DriveLetter
```

object corresponds to a **Drive** object.

Notes:

The **DriveLetter** properties returns a string of length zero ("") when the specified drive is not associated to a letter. This is particularly the case of a network drive that is not mapped to a drive letter.

Example:

```
function ShowDriveLetter(drvPath)
{
  var fso, d, s;
  fso = new ActiveXObject("Scripting.FileSystemObject");
  d = fso.GetDrive(fso.GetDriveName(drvPath));
  s = "Drive " + d.DriveLetter.toUpperCase( ) + ": - ";
  s += d.VolumeName + "<br>";
  s += "Disk available space: " + d.AvailableSpace/1024 + " kilobytes";
  return(s);
}
```

Drives - Property of object: FileSystemObject

See also

The Drives property returns a collection of **Drives** made up of all the **Drive** objects available on the local machine.

Syntax

```
object.Drives
```

object corresponds to a **FileSystemObject** object.

Note:

Removable drives need not be present to form part of the **Drives** collection.
The members of a **Drives** collection are accessed with the help of the **Enumerator** object and the **for** instruction.

DriveType - Property of object: Drive

See also

The **DriveType** property returns a value indicating the type of specified drive.

Syntax

```
object.DriveType
```

object corresponds to a **Drive** object.

Example:

```
function ShowDriveType(drvpath)
{
  var fso, d, s, t;
  fso = new ActiveXObject("Scripting.FileSystemObject");
  d = fso.GetDrive(drvpath);
  switch (d.DriveType)
  {
    case 0: t = "Unknown"; break;
    case 1: t = "Removable"; break;
    case 2: t = "Fixed"; break;
    case 3: t = "Network"; break;
    case 4: t = "CD-ROM"; break;
    case 5: t = "Virtual disk"; break;
  }
  s = "Drive " + d.DriveLetter + ": - " + t;
  return(s);
}
```

Files - Property of object: Folder

See also

The **Files** returns a **Files** collection made up of all the **File** objects found in the specified folder, including hidden and system files.

Syntax

```
object.Files
```

object corresponds to a **Folder** object.

Note:

The members of a **Drives** collection are accessed with the help of the **Enumerator** object and the **for** instruction.

FileSystem - Property of object: Drive

See also

The **FileSystem** property returns the type of file system used on the specified drive.

Syntax

```
object.FileSystem
```

object corresponds to a **Drive** object.

Note:

The possible types of system files are: FAT, NTFS and CDFS.

Example:

```
function ShowFileSystemType(drvPath)
{
  var fso,d, s;
  fso = new ActiveXObject("Scripting.FileSystemObject");
  d = fso.GetDrive(drvPath);
  s = d.FileSystem;
  return(s);
}
```

FreeSpace - Property of object: Drive

See also

This property returns a value indicating the amount of available space for the user on the specified drive (on the server or on the network drive). Read-only.

Syntax

```
object.FreeSpace
```

object always corresponds to a **Drive** object.

Note:

The value returned by the **FreeSpace** property is usually the same as the **AvailableSpace** property. However, differences could arise depending on systems that manage quotas.

Example:

```
function ShowFreeSpace(drvPath)
{
  var fso, d, s;
  fso = new ActiveXObject("Scripting.FileSystemObject");
  d = fso.GetDrive(fso.GetDriveName(drvPath));
  s = "Drive " + drvPath.toUpperCase( ) + " - ";
  s += d.VolumeName + "<br>";
  s += "Available space: " + d.FreeSpace/1024 + " kilobytes";
  return(s);
}
```

IsReady - Property of object: Drive

See also

The **IsReady** property returns a value to indicate whether the specified drive is ready: **True** if the drive is ready and **False** if it's not.

Syntax

```
object.IsReady
```

object corresponds to a **Drive** object.

Note:

For removable disks and CD-ROMs, the **IsReady** property returns **True** only if the necessary media is in place and accessible.

Example:

```
function ShowDriveInfo(drvpath)
{
  var fso, d, s, t;
  fso = new ActiveXObject("Scripting.FileSystemObject")
  d = fso.GetDrive(drvpath)
  switch (d.DriveType)
  {
    case 0: t = "Unknown"; break;
    case 1: t = "Removable"; break;
    case 2: t = "Fixed"; break;
    case 3: t = "Network"; break;
    case 4: t = "CD-ROM"; break;
    case 5: t = "Virtual disk"; break;
  }
  s = "Drive " + d.DriveLetter + ": - " + t;
  if (d.IsReady)
    s += "<br>" + "Drive ready.";
  else
    s += "<br>" + "Drive not ready.";
  return(s);
}
```

IsRootFolder - Property of object: Folder

See also

The **IsRootFolder** property returns a value to indicate if the specified folder is the root folder: **True** if it is, and **False** if it's not.

Syntax

```
object.IsRootFolder
```

object corresponds to a **Folder** object.

Item  - Property of objects: Dictionary and of Drives, Files and Folders collections

See also

The **Item** property defines or returns an item for the specified ***key*** of a **Dictionary** object. For collections, returns an item according to the specified ***key***. Read/write.

Syntax

```
object.Item(key)[ = newitem]
```

object	corresponds to a name of a collection or a **Dictionary** object.
key	key associated to an item (extracted or added).
newitem	optional. Only for **Dictionary** objects, doesn't apply to collections. If used, **newitem** corresponds to a new value associated with a specified key.

Note:

When a **key** is not found during the modification of an item, a new ***key*** is created with the specified **newitem**. When **key** is not found when trying to extract an existing element, a new (***key***) is created and its corresponding element is left empty.

Key  - Property of object: Dictionary

See also

The **Key** property defines a *key* in a **Dictionary** object.

Syntax

```
object.Key(key) = newkey
```

object	refers to the name of a **Dictionary** object.
key	The modified *key* value.
newkey	New value replacing the specified (***key***.

Note:

When no *key* is found while loading a *key*, a new (*key*) is created and its related *item* element remains empty.

Line - Property of object: TextStream

See also

The **Line** property returns the number of the current line in a **TextStream** file. Read-only property.

Syntax

```
object.Line
```

object refers to the name of a **TextStream** object.

Note:

During the first access to a file and before writing to it, **Line** is equal to 1.

Name  - Property of objects: File and Folder

See also

The **Name** property defines or returns the name of a file or a folder. Read/write property.

Syntax

```
object.Name [= newname]
```

object refers to the name of a **File** or **Folder** object.
newname optional. When used, **newname** is the new name of the specified **object** element.

Example:

```
function ShowFileAccessInfo(filespec)
{
    var fso, f, s;
    fso = new ActiveXObject("Scripting.FileSystemObject");
    f = fso.GetFile(filespec);
    s = f.Name + " on drive " + f.Drive + "<br>";
    s += "File created on: " + f.DateCreated + "<br>";
    s += "Accessed on: " + f.DateLastAccessed + "<br>";
    s += "Modified on: " + f.DateLastModified;
    return(s);
}
```

ParentFolder - Property of objects: File and Folder

See also

The **ParentFolder** property returns parent folder object for the specified file or folder. Read-only property.

Syntax

```
object.ParentFolder
```

object refers to a **File** or **Folder** object.

Path  - Property of objects: Drive, File and Folder

See also

The **Path** property returns the access path of a specified file, folder or drive.

Syntax

```
object.Path
```

object corresponds to a **File**, **Folder** or **Drive** object.

RootFolder - Property of object: Drive

See also

The **RootFolder** property returns a **Folder** object. This represents the root folder of the specified drive. Read-only property.

Syntax

```
object.RootFolder
```

object corresponds to a **Drive** object.

Note:

The files and folders contained in the drive are accessible by means of the returned **Folder** object.

SerialNumber - Property of object: Drive

See also

This property returns the serial number (decimal) of the disk volume.

Syntax

```
object.SerialNumber
```

object corresponds to a **Drive** object.

Note:

The **SerialNumber** property is useful, among other things, to ensure that the right disk is placed on a removable disk.

ShareName - Property of object: Drive

See also

ShareName returns the name of the specified shared (network) drive.

Syntax

```
object.ShareName
```

object corresponds to a **Drive** object.

Note:

When **object** is not a network drive, **ShareName** returns a string of length zero ("").

Example:

```
function ShowDriveInfo(drvpath)
{
  var fso, d, s;
  fso = new ActiveXObject("Scripting.FileSystemObject");
  d = fso.GetDrive(fso.GetDriveName(fso.GetAbsolutePathName(drvpath)));
  s = "Drive" + d.DriveLetter + ": - " + d.ShareName;
  return(s);
}
```

ShortName - Property of objects: File and Folder

See also

ShortName returns the abbreviated name of a file or a folder (using the 8.3 name format used by older programs).

Syntax

```
object.ShortName
```

object refers to a **File** or **Folder** object.

Example:

```
function ShowShortName(filespec)
{
    var fso, f, s;
    fso = new ActiveXObject("Scripting.FileSystemObject");
    f = fso.GetFile(filespec);
    s = "The short (DOS) name of file" + "" + f.Name;
    s += "" + "<br>";
    s += "is: " + "" + f.ShortName + "";
    return(s);
}
```

ShortPath - Property of objects: File and Folder

See also

ShortPath returns the abbreviated access path of a file or folder (using the 8.3 name format used by older programs).

Syntax

```
object.ShortPath
```

object refers to a **File** or **Folder** object.

Example:

```
function ShowShortPath(filespec)
{
    var fso, f, s;
    fso = new ActiveXObject("Scripting.FileSystemObject");
    f = fso.GetFile(filespec);
    s = "The short (DOS) access path of file " + "" + f.Name;
    s += "" + "<br>";
    s += "is: " + "" + f.ShortPath + "";
    return(s);
}
```

Size  - Property of objects: File and Folder

See also

Size returns a value representing the size (in kilobytes) of a specified file or folder. If it's a folder, the size includes all the files and subfolders contained in the main folder.

Syntax

```
object.Size
```

object refers to a **File** or **Folder** object.

SubFolders - Property of object: Folder

See also

The **SubFolders** property, returns a **Folders** collection which includes all the subfolders of a specified folder, including those folders that have enabled the *hidden* or *system* attribute.

Syntax

```
object.SubFolders
```

object corresponds to a **Folder** object.

Example:

```
function ShowFolderList(folderspec)
{
  var fso, f, fc, s;
  fso = new ActiveXObject("Scripting.FileSystemObject");
  f = fso.GetFolder(folderspec);
  fc = new Enumerator(f.SubFolders);
  s = "";
  for (;!fc.atEnd(); fc.moveNext())
    {
      s += fc.item();
      s += "<br>";
    }
    return(s);
}
```

TotalSize - Property of object: Drive

See also

The **TotalSize** property returns a value representing the total space (in kilobytes) of a drive or a shared network.

Syntax

```
object.TotalSize
```

object corresponds to a **Drive** object.

Type - Property of objects: File and Folder

See also

This property returns a value representing the type of file or folder specified.
For example, if the file specified end with the extension *.txt* the value returned is "Text document".

Syntax

```
object.Type
```

object refers to a **File** or **Folder** object.

VolumeName - Property of object: Drive

See also

The **VolumeName** property defines or returns the *volume name* of the specified drive.

Syntax

```
object.VolumeName [= newname
```

object corresponds to the name of a **Drive** object.
newname optional. If used, **newname** is the name of the specified **object** element.

Properties of object: Anchor

name
text (Anchor)
x
y

Events of object: Area

onDblClick
onMouseOut
onMouseOver

Properties of object: Array

constructor
index
input
length (Array)
prototype

Methods of object: Array

concat (Array)
join
pop
push
reverse
shift
slice (Array)
splice
sort
toSource
toString
unshift
valueOf

Properties of object: Boolean

constructor
prototype

Methods of object: Boolean

toSource
toString
valueOf

Properties of object: Button

form
name
type
value

Methods of object: Button

blur
click
focus
handleEvent

Events of object: Button

onBlur
onClick
onFocus
onMouseDown
onMouseUp

Properties of object: Checkbox

checked
defaultChecked
form
name
type
value

Methods of object: Checkbox

blur
click
focus
handleEvent

Events of object: Checkbox

onBlur
onClick
onFocus

Properties of object: Date

constructor
prototype

Methods of object: Date

getDate
getDay
getFullYear
getHours
getMilliseconds
getMinutes
getMonth
getSeconds
getTime
getTimezoneOffset
getUTCDate
getUTCDay
getUTCFullYear
getUTCHours
getUTCMilliseconds
getUTCMinutes
getUTCMonth
getUTCSeconds
getVarDate
getYear
parse
setDate
setFullYear
setHours
setMilliseconds
setMinutes
setMonth
setSeconds
setTime
setUTCDate
setUTCFullYear
setUTCHours
setUTCMilliseconds
setUTCMinutes
setUTCMonth
setUTCSeconds
setYear
toGMTString
toLocaleString
toSource
toString
toUTCString
UTC
valueOf

Properties of object: document

alinkColor
anchors
applets
bgColor
classes
cookie
domain
embeds
fgColor
formName
forms
height
ids
images
lastModified
layers
linkColor
links
plugins
referrer
tags
title
URL
vlinkColor
width

Methods of object: document

captureEvents
close
contextual
getSelection
handleEvent
open
releaseEvents
routeEvent
write
writeln

Events of object: document

onClick
onDblClick
onKeyDown
onKeyPress
onKeyUp
onMouseDown
onMouseUp

Methods of object: Enumerator

atEnd
item
moveFirst
moveNext

Properties of object: Error

description
number

Properties of object: event

data
height
layerX
layerY
modifiers
pageX
pageY
screenX
screenY
target
type
which
width
x
y

Properties of object: FileUpload

form
name
type
value

Methods of object: FileUpload

blur
focus
handleEvent
select

Events of object: FileUpload

onBlur
onChange
onFocus

Properties of object: Form

action
elements
encoding
length
method
name
target

Methods of object: Form

handleEvent
reset
submit

Events of object: Form

onReset
onSubmit

Properties of object: Function

arguments
arguments.callee
arguments.caller
arguments.length
arity
caller
constructor
length (Function)
prototype

Methods of object: Function

apply
call
toSource
toString
valueOf

Properties of object: Global

NaN (Global)
Infinity

Methods of object: Global

escape
eval
isFinite
isNaN
parseFloat
parseInt
unescape

Properties of object: Hidden

form
name
type
value

Properties of object: History

current
length
next
previous

Methods of object: History

back
forward
go

Events of object: Image

onAbort
onError
onKeyDown
onKeyPress
onKeyUp
onLoad

Properties of object: Image

border
complete
height
hspace
lowsrc
name
src
vspace
width

Methods of object: Image

handleEvent

Properties of object: JavaArray

length

Methods of object: JavaArray

toString

Events of object: Layer

onBlur
onFocus
onLoad
onMouseOut
onMouseOver

Properties of object: Layer

above
background
below
bgColor
clip.bottom
clip.height
clip.left
clip.right
clip.top
clip.width
document
left
name
pageX
pageY
parentLayer
siblingAbove
siblingBelow
src
top
visibility
window
x
y
zIndex

Methods of object: Layer

captureEvents
handleEvent
load
moveAbove
moveBelow
moveBy
moveTo
moveToAbsolute
releaseEvents
resizeBy
resizeTo
routeEvent

Events of object: Link

onClick
onDblClick
onKeyDown
onKeyPress
onKeyUp
onMouseDown
onMouseOut
onMouseOver
onMouseUp

Properties of object: Link

hash
host
hostname
href
pathname
port
protocol
search
target
text
x
y

Methods of object: Link

handleEvent

Properties of object: Location

hash
host
hostname
href
pathname
port
protocol
search

Methods of object: Location

reload
replace

Properties of object: Math

E
LN2
LN10
LOG2E
LOG10E
PI
SQRT1_2
SQRT2

Methods of object: Math

abs
acos
asin
atan
atan2
ceil
cos
exp
floor
log
max
min
pow
random
round
sin
sqrt
tan

Properties of object: MimeType

description
enabledPlugin
suffixes
type

Properties of object: navigator

appCodeName
appName
appVersion
language
mimeTypes
platform
plugins
userAgent

Methods of object: navigator

javaEnabled
plugins.refresh
preference
savePreferences
taintEnabled

Properties of object: Number

constructor
MAX_VALUE
MIN_VALUE
NaN (Number)
NEGATIVE_INFINITY
POSITIVE_INFINITY
prototype

Methods of object: Number

toSource
toString
valueOf

Properties of object: Object

constructor
prototype

Methods of object: Object

eval
toSource
toString
unwatch
valueOf
watch

Properties of object: Option

defaultSelected
index
length
selected
text
value

Properties of object: Packages

className
java
netscape
sun

Events of object: Password

onBlur
onFocus

Properties of object: Password

defaultValue
form
name
type
value

Methods of object: Password

blur
focus
handleEvent
select

Properties of object: Plugin

description
filename
length
name

Events of object: Radio

onBlur
onClick
onFocus

Properties of object: Radio

checked
defaultChecked
form
name
type
value

Methods of object: Radio

Properties of object: RegExp

$1...$9
$_
$*
$&
$+
$`
$'
constructor
global
ignoreCase
index
input
lastIndex
lastMatch
lastParen
leftContext
multiline
prototype
rightContext
source

Methods of object: RegExp

compile
exec
test
toSource
toString
valueOf

Properties of object: Regular Expression

lastIndex
source

Methods of object: Regular Expression

compile
exec
test

Events of object: Reset

onBlur
onClick
onFocus

Properties of object: Reset

form
name
type
value

Methods of object: Reset

blur
click
focus
handleEvent

Properties of object: screen

availHeight
availLeft
availTop
availWidth
colorDepth
height
pixelDepth
width

Events of object: Select

onBlur
onChange
onFocus

Properties of object: Select

form
length
name
options
selectedIndex
type

Methods of object: Select

blur
focus
handleEvent

Properties of object: String

constructor
length (String)
prototype

Methods of object: String

anchor
big
blink
bold
charAt
charCodeAt
concat (String)
fixed
fontcolor
fontsize
fromCharCode
indexOf
italics
lastIndexOf
link
match
replace
search
slice (String)
small
split
strike
sub
substr
substring
sup
toLowerCase
toSource
toString
toUpperCase
valueOf

Properties of object: Style

align
backgroundColor
backgroundImage
borderBottomWidth
borderColor
borderLeftWidth
borderRightWidth
borderStyle
borderTopWidth
clear
color
display
fontFamily
fontSize
fontStyle
fontWeight
lineHeight
listStyleType
marginBottom
marginLeft
marginRight
marginTop
paddingBottom
paddingLeft
paddingRight
paddingTop
textAlign
textDecoration
textIndent
textTransform
whiteSpace
width

Methods of object: Style

borderWidths
margins
paddings

Events of object: Submit

onBlur
onClick
onFocus

Properties of object: Submit

form
name
type
value

Methods of object: Submit

blur
click
focus
handleEvent

Events of object: Text

onBlur
onChange
onFocus
onSelect

Properties of object: Text

defaultValue
form
name
type
value

Methods of object: Text

blur
focus
handleEvent
select

Events of object: Textarea

onBlur
onChange
onFocus
onKeyDown
onKeyPress
onKeyUp
onSelect

Properties of object: Textarea

defaultValue
form
name
type
value

Methods of object: Textarea

blur
focus
handleEvent
select

Methods of object: VBArray

dimensions
getItem
lbound
toArray
ubound

Events of object: window

onBlur
onDragDrop
onError
onFocus
onLoad
onMove
onResize
onUnload

Properties of object: window

closed
crypto
defaultStatus
document
frames
history
innerHeight
innerWidth
length
location
locationbar
menubar
name
offscreenBuffering
opener
outerHeight
outerWidth
pageXOffset
pageYOffset
parent
personalbar
screenX
screenY
scrollbars
self
status
statusbar
toolbar
top
window

Methods of object: window

alert
atob
back
blur
btoa
captureEvents
clearInterval
clearTimeout
close
confirm
crypto.random
crypto.signText
disableExternalCapture
enableExternalCapture
find
focus
forward
handleEvent
home
moveBy
moveTo
open
print
prompt
releaseEvents
resizeBy
resizeTo
routeEvent
scroll
scrollBy
scrollTo
setHotKeys
setInterval
setResizable
setTimeout
setZOptions
stop

Properties of object: Dictionary

Count
Item
Key

Methods of object: Dictionary

Add
Exists
Items
Keys
Remove
RemoveAll

Properties of object: Drive

AvailableSpace
DriveLetter
DriveType
FileSystem
FreeSpace
IsReady
Path
RootFolder
SerialNumber
ShareName
TotalSize
VolumeName

Properties of object: File

Attributes
DateCreated
DateLastAccessed
DateLastModified
Drive
Name
ParentFolder
Path
ShortName
ShortPath
Size
Type

Methods of object: File

Copy
Delete
Move
OpenAsTextStream

Properties of object: Drives

Count
Item

Properties of object: Files

Count
Item

Properties of object: FileSystemObject

Drives

Méthodes pour l'objet FileSystemObject

BuildPath
CopyFile
CopyFolder
CreateFolder
CreateTextFile
DeleteFile
DeleteFolder
DriveExists
FileExists
FolderExists
GetAbsolutePathName
GetBaseName
GetDrive
GetDriveName
GetExtensionName
GetFile
GetFileName
GetFolder
GetParentFolderName
GetSpecialFolder
GetTempName
MoveFile
MoveFolder
OpenTextFile

Properties of object: Folder

Attributes
DateCreated
DateLastAccessed
DateLastModified
Drive
Files
IsRootFolder
Name
ParentFolder
Path
ShortName
ShortPath
Size
SubFolders
Type

Methods of object: Folder

Copy
Delete
Move
OpenAsTextStream

Properties of object: **Folders**

Count
Item

Methods of object: Folders

Add

Properties of object: TextStream

AtEndOfLine
AtEndOfStream
Column
Line

Methods of object: TextStream

Close
Read
ReadAll
ReadLine
Skip
SkipLine
Write
WriteBlankLines
WriteLine

abs – Method of object: Math

See also

Returns the absolute value of a number.

Syntax

```
Math.abs(Number)
```

Number is a numeric expression.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Math.abs(n)
value = -45.7
document.write("The absolute value of variable 'value = "+value+"' is
<BR>Math.abs
(value) = "+Math.abs(value)+"<BR><BR>");
</SCRIPT>
```

acos – Method of object: Math

See also

Returns the arc cosine (in radians) of a number. The value returned is between **+0** and **+π**.

Syntax

```
Math.acos(Number)
```

Number is a numeric expression between **-1** and **1**. If **Number** is outside this range, the value returned is **NaN**.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Math.acos(n)
value=-0.268
document.write("The ac cosine of variable 'value = "+value+"' is
<BR>Math.acos(value) = "+Math.acos(value)+" radians.<BR><BR>");
</SCRIPT>
```

alert – Method of object: window

See also

Displays an Alert dialog box with a message and an OK button. This method is used to display a message that does not require a user decision.

Even though the **alert** method employs the window object, you do not need to specify a window reference to call it. For example, `windowReference.alert()` isn't necessary.

Syntax

```
alert("aMessage")
```

`aMessage` is a character string or a property of an existing object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
alert ('pay attention to messages in the alert box');
</SCRIPT>
```

anchor – Method of object: String

See also

This method creates an HTML anchor used as a hypertext target. You need to use the **anchor** methods with the write or writeln methods to create and display an HTML element in your document. You can also create an element with the **anchor** method and then display it in your document using the write or writeln methods.

Elements created with this method are written to the elements array. Refer to the Anchor object for more information on the elements array.

Syntax

```
textName.anchor(attributeName)
```

textName is either a text string or a property of an existing object. **textName** represents a text string that acts as a hypertext target.

attributeName is either a text string or a property of an existing object. **attributeName** represents the **NAME** attribute in the **<A>** command.

Example

```
<SCRIPT LANGUAGE="JavaScript">
var description="UNGI !";
document.writeln(description.anchor("ungi"));
</SCRIPT>
```

apply – Method of object: Function

See also

Applies on an object another object's method.

Syntax

```
apply(cetArg[, tableauArguments])
```

Parameters

`thisArg`	Parameter for the object it calls.
`argumentsArray`	An arguments array for the object.

asin – Method of object: Math

See also

Returns the arc sine (in radians) of a number. The value returned is found between **-π/2** and **+π/2**.

Syntax

```
Math.asin(Number)
```

Number is a numeric expression between **-1** and **1**. If **Number** is outside this range, the value returned is **NaN**.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Math.asin(n)
value=-0.268
document.write("The arc sine of variable 'value = "+value+"' is
<BR>Math.asin(value) = "+Math.asin(value)+" radians.<BR><BR>");
</SCRIPT>
```

atan – Method of object: Math

See also

Returns the arc tangent (in radians) of a number. It returns a numeric value between **-π/2** and **+π/2**.

Syntax

```
Math.atan(Number)
```

Number is a numeric expression.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Math.atan(n)
value=-0.268
document.write("The arc tangent of variable 'value = "+value+"' is
<BR>Math.atan(value) = "+Math.atan(value)+" radians.<BR><BR>");
</SCRIPT>
```

atan2 – Method of object: Math

See also

Returns a (theta) angle of a (r,theta) coordinate that correspond to the specified Cartesian data (x,y).

Syntax

```
Math.atan2(x,y)
```

x is a numeric expression representing the Cartesian x-coordinate.
y is a numeric expression representing the Cartesian y-coordinate.
It returns a numeric value between -π and π. It represents the angle of a (y,x) point.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Math.atan2(x,y)
x=27
y=12
document.write("The arctange of the Cartesian coordinates 'x = "+x+" ' and
y = ' "+y+" ' is  <BR>Math.atan2(x,y) = "+Math.atan2(x,y)+"
radians.<BR><BR>");
</SCRIPT>
```

atEnd – Method of object: Enumerator

See also

Returns a Boolean value if an Enumerator object is found at the end of a collection.

Syntax

```
myEnum.atEnd( )
```

The **myEnum** argument represent any Enumerator object.

Return value

true if the current element is the last one in the collection or is not defined, or if the collection is empty.
false in all other situations.

atob – Method of object: window

See also

Decodes a data string that was coded using *base-64*.

Syntax

```
atob(encodedData)
```

Parameter

`encodedData`: A data string, coded using *base-64*.

back – Method of objects: History and window

See also

This method loads the previous URL in the history list. This method performs the same action as a user choosing the **Back** button in the browser.
The back method is the same as the `history.go(-1)` method.

Syntax:

```
history.back()
```

Example

```
<FORM NAME="form">
<INPUT TYPE="button" VALUE="return" onClick="return()">
</FORM>

<SCRIPT LANGUAGE="JavaScript">
function return()
{
history.back();
}
</SCRIPT>
```

big – Method of object: String

See also

This method causes a text string to be displayed in a big font. It is similar to using the **<BIG>** HTML tag. You must use the write or writeln methods to format and display the text string in the document.

Syntax

```
textName.big()
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
string = new String("This is a big display!");
document.writeln("normal display:"+string+"<BR>");
document.writeln("BIG display:"+string.big());
</SCRIPT>
```

blink – Method of object: String

See also

This method is used to make a text string blink, exactly like the **<BLINK>** tag. You must use the write or writeln methods to format and display the text string in the document.

Syntax

```
textName.blink()
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
string = new String("Please note that only Netscape displays blinking
text!");
document.writeln("normal display: "+string+"<BR>");
document.writeln("BLINKing display : "+string.blink()+"<BR>");
</SCRIPT>
```

blur – Method of objects: Button, Checkbox, FileUpload, Password, Radio, Reset, Select, Submit, Text, Textarea and window

See also

This methods removes focus" from the specified object. This object must be an element in the form.
The **blur** method can also be applied to Frame and window objects. When "focus" is removed form a window, it is sent to the background.

Syntax

```
passwordName.blur()
listName.blur()
textName.blur()
textareaName.blur()
frameReference.blur()
windowReference.blur()
```

passwordName is the value of attribute NAME of the Password object or an element in an elements array.
listName is the value of attribute NAME of the Select object or an element in an elements array.
textName is the value of attribute NAME of the Text object or an element in an elements array.
textareaName is the value of attribute NAME of the Textarea object or an element in an elements array.
frameReference is a valid way to reference a frame, as it is described in the Frame object.
windowReference is a valid way to reference a window, as it is described in the window object.

Example

```
To remove focus from a window:
<FORM NAME="form">
<INPUT TYPE="button" VALUE="blur" onClick="window.blur();">
</FORM>
```

bold – Method of object: String

See also

This method is used to display a text string in bold. It is similar to using the **<B>** HTML tag. You must use the write or writeln methods to format and display the text string in the document.

Syntax

```
textName.bold()
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
string = new String("This is a bold display!");
document.writeln("normal display:"+string+"<BR>");
document.writeln("BOLD display:"+string.big());
</SCRIPT>
```

borderWidths – Method of object: Style

See also

Used to specify the width of an HTML border element. The value can be expressed in pixels (px), points (pt), etc.

Syntax

```
borderWidths(top, right, bottom, left)
```

Parameters

top	is a string that specifies the **Style.borderTopWidth** property value.
right	is a string that specifies the **Style.borderRightWidth** property value.
bottom	is a string that specifies the **Style.borderBottomWidth** property value.
left	is a string that specifies the **Style.borderLeftWidth** property value.

btoa – Method of object: window

See also

Used to create an encoded *base-64* type ASCII string for a binary data string.

Syntax

```
btoa(chaîneÀEncoder)
```

Parameters

`chaîneÀEncoder`	binary string to be coded.

call – Method of object: Function

See also

Used to execute another object's method.

Syntax

```
functionName.call(arguments)
```

functionName represents the object called.
arguments represents the arguments of the **functionName** method.

captureEvents – Method of objects: document, Layer and window

See also

Indicates that a type of event must be addressed to the object.

Syntax

```
captureEvents(eventType)
```

Parameters

`eventType`	type of event that has to address the object.

Note

With regards to the window object, the use this property requires *UniversalBrowserWrite* privileges.

ceil – Method of object: Math

See also

Returns the least integer greater than or equal to a number.

Syntax

```
Math.ceil(Number)
```

Number is a numeric expression.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Math.ceil(n)
value=-4.268
document.write("The ceil value "+value+" of variable is
<BR>Math.ceil(value) = "+Math.ceil(value)+" .<BR><BR>");
</SCRIPT>
```

charAt – Method of object: String

See also

Returns the specified character from the string.

Syntax

```
textName.charAt(index)
```

index An integer between 0 and the length of the characters in **textName** -1 (**nomTexte.length - 1**), or a property of an existing object.
The characters in a text string are indexed from left to right. The index of the first character is 0 and the index of the last character is **textName.length - 1**. If the index supplied is out of range, JavaScript returns an empty string.

Example

```
<FORM NAME="form">
<INPUT TYPE="text" NAME=output MAXLENGTH=40 SIZE=40 VALUE="enter some
text">
<INPUT TYPE="button" VALUE="change status" onClick="derCarStatus()">
</FORM>

<SCRIPT LANGUAGE="JavaScript">

function derCarStatus()
{
string = new String();
string = document.form.output.value;
string = "the last character is: "+string.charAt(string.length - 1);
window.status= string;
}
</SCRIPT>
```

charCodeAt – Method of object: String

See also

Returns the Unicode value of the specified character.

Syntax

```
textName.charCodeAt(index)
```

textName refers to a text string.
index corresponds to the index value of the string character.

clearInterval – Method of object: window

See also

Cancels the authorised delay specified with the setInterval method.

Syntax:

```
clearInterval(intervalID)
```

Parameters

intervalID	Returned delay by a previous call to the setInterval method.

clearTimeout – Method of object: window

See also

Cancels a timeout that was set with the setTimeout method.

Syntax:

```
clearTimeout(timeoutID)
```

Parameters

timeoutID s a timeout setting that was returned by a previous call to the setTimeout method.

Example

```
<FORM>
Clicking triggers a counter; it waits 5000 milliseconds and then sends an
alert:
<INPUT TYPE="checkbox" VALUE="help" NAME="help" onClick="Timer()">
</FORM>
<SCRIPT LANGUAGE="JavaScript">
function Timer()
{
timeoutID = window.setTimeout("window.alert('5 seconds have
elapsed.')",5000);
}
</SCRIPT>

<FORM>
Oups ! Timeout cancelled!
<INPUT TYPE="checkbox" VALUE="help" NAME="help"
onClick="window.clearTimeout
(timeoutID)">
</FORM>
```

click – Method of objects: Button, Checkbox, Radio, Reset and Submit

See also

Simulates a mouse-click (pressing and releasing the left button) on an element in a form.

Syntax

```
buttonName.click()
buttonNameRadio[index].click()
buttonNameCheckbox.click()
```

buttonName is either the value of the NAME attribute of a Reset or Submit object, or an element in an elements array.
buttonNameRadio is the value of attribute NAME of the Radio object or an element in an elements array.
index is an integer representing a radio button of a Radio object.
checkboxButtonName is the value of attribute NAME of the Checkbox object or an element in an elements array.

Example

```
<FORM>
<INPUT TYPE="button" VALUE="click here"
onClick="document.form.target.click();return true;">
</FORM>
<FORM NAME="form">
<INPUT TYPE="button" NAME="target" VALUE="target button">
</FORM>
```

close – Method of object: document

See also

This method closes a document that was opened using the **`document.open()`** method. Refer to the open) method.

Syntax

```
document.close()
```

Example

```
Input the text that will be placed in the second window:
<FORM NAME=form>
<TEXTAREA NAME="Enter" ROWS=5 COLS=40></TEXTAREA>
<INPUT TYPE="button" VALUE="click here " onClick="open()">
</FORM>
<SCRIPT LANGUAGE="JavaScript">
function open()
{
Window2=open("","","scrollbars=no,width=400,height=200");
Window2.document.open();
Window2.document.write("<HEAD><TITLE>second window!</TITLE></HEAD>");
Window2.document.write("<CENTER><BIG><B>"+document.form.enter.value+"</
B></BIG>");
Window2.document.write("<FORM><INPUT TYPE='button' VALUE='close'
onClick='window.close()'></FORM>");
Window2.document.write("</CENTER>");
Window2.document.close();
}
</SCRIPT>
```

close – Method of object: window

See also

Used to close a window that was opened using the window.open() method. For the other windows, a confirmation message will appear on the screen. If no window is specified, JavaScript will close the current window.

Syntax

```
windowReference.close()
```

Example

```
<FORM>
<INPUT TYPE="button" VALUE="Click here" onClick="Window2.close();return
true;">
</FORM>
<SCRIPT LANGUAGE="JavaScript">
Window2=open("","subscribe", "scrollbars=no,width=150,height=200");
Window2.document.write("<HEAD><TITLE>second window!</TITLE></HEAD>")
Window2.document.write("<CENTER><BIG><B>This is the second window
/B></BIG></CENTER>")
</SCRIPT>
```

compile – Method of objects: RegExp and Regular Expression

See also

Compiles a regular expression object during execution of a script.

Syntax

```
regexp.compile(pattern[, flags])
```

Parameters

regexp represents the name of the regular expression.

pattern A string containing the text of the regular expression.

flags, when specified, the values are:

- **g**: global search
- **i**: case insensitive
- **gi**: global search and case insensitive.

concat (Array) – Method of object: Array

See also

Joins two arrays and returns a new array.

Syntax

```
arrayName1.concat(arrayName2, arrayName3, ..., arrayNameN)
```

arrayName1 to **arrayNameN** represents the **Array** objects that will be concatenated.

concat (String) – Method of object: String

See also

Combines the text of two or more strings and returns a new string.

Syntax

```
textName1.concat(textName2, textName3)
```

textName1, **textName2** and **textName3** Strings to concatenate to the desired string.

confirm – Method of object: window

See also

Displays a Confirm dialog box with the specified message and OK and Cancel buttons. Use the confirm method to ask the user to make a decision.

The **confirm** method returns TRUE if the user presses the OK button and FALSE if he presses the CANCEL button.

Even though the **confirm** method employs the window object, you do not need to specify a window reference to call it. For example, `windowReference.confirm()` is not necessary.

Syntax

```
confirm("aMessage")
```

`aMessage` is a character string or a property of an existing object.

Example

```
<FORM>
<INPUT TYPE="button" VALUE="click here to quit." onClick="confirmation()">
</FORM>
<SCRIPT LANGUAGE="JavaScript">
function confirmation()
{
results = confirm('are you sure you want to quit?');
if(result=="1")
     this.close();
}
</SCRIPT>
```

contextual – Method of object: document

See also

Uses contextual selection criteria to specify a Style object that can set the style of individual HTML tags.

Syntax

```
contextual(context1, ...[contextN,] affectedStyle)
```

Parameters

`context1, ...[contextN]`	The Style objects described by the document.classes property or by document.tags, establish the context for the Style affected object.
`affectedStyle`	The Style object whose style properties you want to change.

COS – Method of object: Math

See also

Returns the cosine of a number. The cos method returns a numeric value between **-1** and **1**, which represents the value of the angle.

Syntax

```
Math.cos(Number)
```

Number is a numeric expression which represents the size of an angle in degrees.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Math.cos(n)
value=-1.56
document.write("The cosine of angle 'value = "+value+"' radians is
<BR>Math.cos
(value) = "+Math.cos(value)+" .<BR><BR>");
</SCRIPT>
```

crypto.random – Method of object: window

See also

Returns a data pseudo-random string whose length is defined by the number of bytes.

Syntax

```
crypto.random(byteNumber)
```

crypto.signText – Method of object: window

See also

Returns a string of encoded data which represents a signed object.

Syntax

```
crypto.signText (text, selectionStyle [, authority1 [, ... authorityN]])
```

Parameters

text	A string evaluating to the text the user must sign.
selectionStyle	A string equivalent to: **ask** indicates that the user will be presented with a dialog box containing a list of possible certificates. **auto** indicates that the browser will automatically select of one the certificates of **authority1**... **authorityN** parameters.
authority1... authorityN	Optional string indicating the certificate authorizations the server accepts.

dimensions – Method of object: VBArray

See also

Returns the dimensions of a VBArray object.

Syntax

```
array.dimension( )
```

disableExternalCapture – Method of object: window

See also

Disables external event capturing defined by the enableExternalCapture method.

Syntax

```
disableExternalCapture()
```

enableExternalCapture – Method of object: window

See also

Allows a window made up of different frames to capture events found in pages found in different sites (different servers).

Syntax

```
enableExternalCapture()
```

escape – Method

See also

Returns the ASCII (hexadecimal) encoding of an argument in the ISO-Latin-1 character set. This top-level function is not associated with any particular object and is part of the JavaScript language.

The value returned by the escape function is a string of the form "%xx," where xx is the ASCII encoding of a character in the argument. Except for characters other than numbers, letters and characters + - * / _ @ and . (dot), the value returned by the escape function is a string preceded by the **%** symbol. If you pass the **escape** function an alphanumeric character, the escape function returns the same character.

Syntax

```
escape("text")
```

text is an ISO-Latin-1character set.

Example

```
<FORM NAME="form">
<INPUT TYPE="text" NAME=input MAXLENGTH=40 SIZE=40 VALUE="Enter some text">
<INPUT TYPE="button" VALUE="ASCII conversion." onClick="conversion()"><BR>
<INPUT TYPE="text" NAME=convert MAXLENGTH=40 SIZE=40>
</FORM>
<SCRIPT LANGUAGE="JavaScript">
function conversion()
{
document.form.convert.value=escape(document.form.input.value);
}
</SCRIPT>
```

eval – Method

See also

The eval method evaluates a string of JavaScript code without reference to a particular object and returns a value. The **eval** function is a top-level function not associated to any particular object. The function is part of the JavaScript language.

Syntax

```
eval(text)
```

`text` could be any text string representing a JavaScript expression. This expression can contain variables and a property of an existing object.

Example

```
Enter a mathematical expression:<BR>
<FORM NAME="form">
<INPUT TYPE="text" NAME=input MAXLENGTH=40 SIZE=40>
<INPUT TYPE="button" VALUE="evaluation." onClick="conversion()"><BR>
<INPUT TYPE="text" NAME=convert MAXLENGTH=40 SIZE=40>
</FORM>
<SCRIPT LANGUAGE="JavaScript">
function conversion()
{
document.form.convert.value=eval(document.form.output.value);
}
</SCRIPT>
```

exec – Method of objects: RegExp and Regular Expression

See also

Executes the search for a match in a specified string. Returns a result array.

Syntax

```
regexp.exec([str])regexp([str])
```

Parameters

regexp represents the name of a regular expression.

str represents the string to be compared with the regular expression. If omitted, the RegExp.input value is used.

exp – Method of object: Math

See also

Returns **e number**, where **number** is an argument and **e** is Euler's constant, the basis of natural logarithms.

Syntax

```
Math.exp(Number)
```

Number is a numeric expression.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Math.exp(n)
value=2.53
document.write("The power of<FONT SIZE +1>e</FONT><SUP>value</SUP>  or e =
Euler's constant and 'value = "+value+"' is <BR>Math.exp(value) =
"+Math.exp(value)+" .<BR><BR>");
</SCRIPT>
```

find – Method of object: window

See also

Find the text string indicated in the contents of the specified window.

Syntax:

```
find([string[, caseSensitive, backward]])
```

Parameters

string represents the text string to find.
caseSensitive is a Boolean value. If TRUE, it indicates that the search is case sensitive. If you use this parameter, you must also used the **backward** parameter.
backward is a Boolean value. If TRUE, it specifies a search in the opposite direction. If you use this parameter, you must also used the **caseSensitive** parameter.

Returns

true if the string was found, or **false**.

fixed – Method of object: String

See also

Causes a string to be displayed in fixed-pitch font similar to that of typewriters. It is similar to using the **<TT>** HTML tag. You must use the write or writeln methods to format and display the text string in the document.

Syntax

```
textName.fixed()
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
string = new String("This is a display!");
document.writeln("normal display :"+string+"<BR>");
document.writeln("fixed-type display:"+string.fixed());
</SCRIPT>
```

floor – Method of object: Math

See also

Returns the greatest integer less than or equal to a number.

Syntax

```
Math.floor(Number)
```

Number is a numeric expression.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Math.floor(n)
value=8.268
document.write("The ceil value "+value+" of variable is
<BR>Math.ceil(value) = "+Math.floor(value)+" .<BR><BR>");
</SCRIPT>
```

focus – Method of objects: Button, Checkbox, FileUpload, Password, Radio, Reset, Select, Submit, Text, Textarea and window

See also

This method gives "focus" to the specified object. This object is usually an element in a form.
It can also be applied to Frame and window objects. When an object gets "focus" in a window, it is displayed in the foreground.

Syntax

```
passwordName.focus()
listName. focus()
textName. focus()
textareaName. focus()
frameReference.focus()
windowReference.focus()
```

passwordName is the value of attribute NAME of the Password object or an element in an elements array.
listName is the value of attribute NAME of the Select object or an element in an elements array.
textName is the value of attribute NAME of the Text object or an element in an elements array.
textareaName is the value of attribute NAME of the Textarea object or an element in an elements array.
frameReference is a valid way to reference a frame, as it is described in the Frame object.
windowReference is a valid way to reference a window, as it is described in the window object.

Example

```
<FORM>
<INPUT TYPE="button" VALUE="Click here "
onClick="document.form.target.focus();return true;">
</FORM>
<FORM NAME="form">
<INPUT TYPE="button" NAME="Target" VALUE="Target button">
</FORM>
```

fontcolor – Method of object: String

See also

Causes a string to be displayed in the specified color as the font element and the **<FONT COLOR=aColor>** attribute. You must use the write or writeln methods to format and display the text string in the document.

Syntax

```
textName.fontcolor(color)
```

`color` is a text string or a property of an existing object, representing a color with a hexadecimal number. You can also used the names of colors (red, blue, yellow, green, etc.)

Example

```
<SCRIPT LANGUAGE="JavaScript">
string = new String("green string!");
document.write("this string is a "+string.fontcolor("00FF00"));
</SCRIPT>
```

fontsize – Method of object: String

See also

Causes a string to be displayed in the specified dimensions as the font element and the **<FONT SIZE=Dimensions>** attribute. You must use the write or writeln methods to format and display the text string in the document.

Syntax

```
textName.fontsize(Dimensions)
```

Dimension is an integer between 1 and 7 or a text representing an integer or a property of an existing object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
string = new String("Variable size line.");
size = 0;
for(i=0; i<string.length; i++)
{
size++;
if(size==8)
     size=0;
//caract =
document.write(string.charAt(i).fontsize(size));
}
</SCRIPT>
```

forward – Method of objects: History and window

See also

This method loads the next URL in the history list. This method performs the same action as a user choosing the **Forward** button in the browser.
This method is the same as the history.go(1) method.

Syntax:

```
history.forward()
```

Example

```
<FORM NAME="form">
<INPUT TYPE="button" VALUE="forward" onClick="forward()">
</FORM>

<SCRIPT LANGUAGE="JavaScript">
function forward()
{
history.forward();
}
</SCRIPT>
```

fromCharCode – Method of object: String

See also

Returns a string created by using the specified sequence of Unicode values.

Syntax

```
String.fromCharCode(Unicode1, Unicode2, …, Unicoden)
```

Unicode1, Unicode2, …, Unicoden represents a sequence of numbers that are Unicode values.

getDate – Method of object: Date

See also

Returns the day of the month for the specified date. The value returned is an integer between 1 and 31.

Syntax

```
dateObjectName.getDate()
```

dateObjectName is the name of a date object or a property of an existing object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// returns the day of the month:
// today.getDate()
today= new Date();
daymonth = today.getDate()
document.write("Today is day "+daymonth+" of this month<BR>");
</SCRIPT>
```

getDay – Method of object: Date

See also

Returns the day of the week for the specified date. The value returned in an integer corresponding to the day of the week, where zero is Sunday and 6 is Saturday.

Syntax

```
dateObjectName.getDay()
```

dateObjectName is the name of a date object or a property of an existing object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// returns the day of the week for a specific date (0= Sunday):
// today.getDay()
today = new Date();
weekday= today.getDay()
document.write("Today is day "+weekday+" of the week (0 = Sunday, 6 =
Saturday)<BR>");
</SCRIPT>
```

getFullYear – Method of object: Date

See also

Returns the specified year using four digits.

Syntax:

```
dateObjectName.getFullYear()
```

`dateObjectName` is the name of a date object or a property of an existing object.

getHours – Method of object: Date

See also

Returns the hour for the specified hour. The value returned is an integer between 0 and 23.

Syntax

```
dateObjectName.getHours()
```

dateObjectName is the name of a date object or a property of an existing object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Get hour:
// today.getHours()
today = new Date();
hour = today.getHours()
document.write("The time is   "+hour+" hours <BR>");
</SCRIPT>
```

getItem – Method of object: VBArray

See also

Returns an element at the indicated location.

Syntax

```
safeArray.getItem(dimension1[, dimension2, ...], dimensionN)
```

Parameters

safeArray represent a VBArray object.
dimension1, ..., dimensionN represents the exact location of the required VBArray element. **N** represent the dimensions number of the VBArray object.

getMilliseconds – Method of object: Date

See also

Returns the milliseconds in the current time.

Syntax

```
dateObjectName.getMilliseconds()
```

dateObjectName is the name of a date object or a property of an existing object.

getMinutes – Method of object: Date

See also

Returns the minutes in the current time. The value returned is an integer between 0 and 59.

Syntax

```
dateObjectName.getMinutes()
```

dateObjectName is the name of a date object or a property of an existing object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Return minutes:
// today.getMinutes()
today = new Date();
minute = today.getMinutes()
document.write("This is minutes "+minute+" of this hour<BR>");
</SCRIPT>
```

getMonth – Method of object: Date

See also

Returns the month in the specified date. The value returned is an integer between 0 and 11, where 0 is January and 11 is December.

Syntax

```
dateObjectName.getMonth()
```

dateObjectName is the name of a date object or a property of an existing object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Return month:
// today.getMonth()
today = new Date();
month = today.getMonth()
document.write("This is month "+month+" of this year (Note: counted from 0
to 11)<BR>");
</SCRIPT>
```

getSeconds – Method of object: Date

See also

Returns the seconds in the current time. The value returned is an integer between 0 and 59.

Syntax

```
dateObjectName.getSeconds()
```

dateObjectName is the name of a date object or a property of an existing object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Return seconds:
// today.getSeconds()
today = new Date();
second = today.getSeconds()
document.write("These are the seconds "+second+" of the current hour
<BR>");
</SCRIPT>
```

getSelection – Method of object: document

See also

Returns a string containing text of the current selection.

Syntax

```
getSelection()
```

Note

This method only works in the current document.

getTime – Method of object: Date

See also

Returns the numeric value corresponding to the time for the specified date. The value returned is a number in milliseconds. You can use this method to assign a date and time to another Date object.

Syntax

```
dateObjectName.getTime()
```

dateObjectName is the name of a Date object or the property of an existing object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Return the  number in milliseconds elapsed during the day :
// today.getTime()
today = new Date();
millisecond = today.getTime()
document.write("How many "+milliseconde+" milliseconds have elapsed since
the 1st of January 1970? (this value can be assigned to another
dateObject)<BR>");
</SCRIPT>
```

getTimezoneOffset – Method of object: Date

See also

Returns the difference (in minutes) between the local time and GMT (Greenwich Meridian Time) time.

Syntax

```
dateObjectName.getTimezoneOffset()
```

dateObjectName is the name of a Date object or the property of an existing object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Return the difference (in minutes) between the local time and GMT time:
// today.getTimezoneOffset()
today = new Date();
GMToffset = today.getTimezoneOffset()
document.write("There is a difference of "+GMToffset+" minutes between our
local time and the GMT<BR>");
</SCRIPT>
```

getUTCDate – Method of object: Date

See also

Returns the day of the month specified according to the UTC standard. The value returned is an integer between 1 and 31.

Syntax

```
dateObjectName.getUTCDate()
```

dateObjectName is the name of a Date object or the property of an existing object.

getUTCDay – Method of object: Date

See also

Returns the number of the day in the specified week according to the UTC standard). The value returned is an integer between 0 and 6, 0 for Sunday and 6 for Saturday.

Syntax

```
dateObjectName.getUTCDay()
```

dateObjectName is the name of a Date object or the property of an existing object.

getUTCFullYear – Method of object: Date

See also

The value returned is the year using four digits, as per the UTC standard.

Syntax

```
dateObjectName.getUTCFullYear()
```

dateObjectName is the name of a Date object or the property of an existing object.

getUTCHours – Method of object: Date

See also

Returns the hour for a specified hour, as per the UTC standard.

Syntax

```
dateObjectName.getUTCHours()
```

dateObjectName is the name of a Date object or the property of an existing object.

getUTCMilliseconds – Method of object: Date

See also

Returns the milliseconds for a specified hour, as per the UTC standard.

Syntax

```
dateObjectName.getUTCMilliseconds()
```

dateObjectName is the name of a Date object or the property of an existing object.

getUTCMinutes – Method of object: Date

See also

Returns the minutes for a specified hour, as per the UTC standard.

Syntax

```
dateObjectName.getUTCMinutes()
```

dateObjectName is the name of a Date object or the property of an existing object.

getUTCMonth – Method of object: Date

See also

Returns the month for a specified date, as per the UTC standard.

Syntax

```
dateObjectName.getUTCMonth()
```

dateObjectName is the name of a Date object or the property of an existing object.

getUTCSeconds – Method of object: Date

See also

Returns the seconds for a specified hour, as per the UTC standard.

Syntax

```
dateObjectName.getUTCSeconds()
```

dateObjectName is the name of a Date object or the property of an existing object.

getVarDate – Method of object: Date

See also

Returns the **VT_DATE** value of a Date object.

Syntax

```
dateobj.getVarDate( )
```

The **dateobj** arguments references a Date object.

Note

ActiveX and other objects that accept and return date values using the **VT_DATE** format are used with the getVarDate method.

getYear – Method of object: Date

See also

Returns the year in the specified date. The value returned is the year less 1900. For example, if the year is 1987, the value returned is 87.

Syntax

```
dateObjectName.getYear()
```

dateObjectName is the name of a Date object or the property of an existing object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// This method returns the year in a specified date.
// The value returned is the year less 1900. For example, if the year is
1987, the value returned is 87.
// today.getYear()
today = new Date();
year = today.getYear()
document.write("A number of  "+year+" years have elapsed since 1900<BR>");
</SCRIPT>
```

go – Method of object: History

See also

This method loads an existing URL from the history list. This method navigates to the location in the history list determined by the argument that you specify.

Syntax

```
history.go(delta | "location")
```

Parameters

delta is an integer or a property of an existing object, representing the relative position in the history list. The **delta** argument can be a positive or negative integer. If **delta** is greater than zero, the go method loads the URL that is that number of entries forward in the history list. If **delta** is lower than zero, the go method loads the URL that is that number of entries backward in the history list. If **delta** is equal to zero, the go method reloads the current page.
location is a text string of a property of an existing object, representing all or part of a URL found in a history list.

Example

```
<FORM NAME="form">
<INPUT TYPE="button" VALUE="return" onClick="return()">
</FORM>

<SCRIPT LANGUAGE="JavaScript">
function return()
{
history.go(-1);
}
</SCRIPT>
```

handleEvent – Method of objects: Button, Checkbox, document, FileUpload, Form, Image, Layer, Link, Password, Radio, Reset, Select, Submit, Text, Textarea and window

See also

Used to assign a handler for all types of events for a specified object. .

Syntax

```
handleEvent(event)
```

Parameters

`event` represents the name of an event for which the object specified has an event handler.

home – Method of object: window

See also

Loads the URL of the start page specified in the options (or preferences) section of the user's browser.
This method has the same effect as using the **Home** button in Netscape Navigator or **Start** in Internet Explorer (**Home** button).

Syntax

```
home()
```

indexOf – Method of object: String

See also

The characters in a text string are indexed from left to right. The index of the first character is zero, and the index of the last character is **textName.length-1**, that is, the length of the text string name called minus 1.

The **indexOf** method returns the index in the text string of the first occurrence found.

If no value is specified for the **fromIndex** argument, the default value is set to 0. If the **SearchValue** is not found, JavaScript returns -1.

Syntax

```
textName.indexOf(SearchValue, [fromIndex] )
```

SearchValue is a character or a character string or a property of an existing object, representing the value to search for.
fromIndex is the location within the calling string to start the search from.
fromIndex can be any integer from zero to **textName.length-1** or a property of an existing object.

Example

```
<FORM NAME="form">
text:<BR>
<INPUT TYPE="text" NAME=enter MAXLENGTH=40 SIZE=40 VALUE="Enter some text">
<BR>
String to find:<BR>
<INPUT TYPE="text" NAME=string MAXLENGTH=40 SIZE=40 VALUE="one">
<BR>
<INPUT TYPE="button" VALUE="Search" onClick="search()">
<BR>
String position:<INPUT TYPE="text" NAME=position MAXLENGTH=4 SIZE=4>
</FORM>
<SCRIPT LANGUAGE="JavaScript">
function search()
{

document.form.position.value=escape(document.form.enter.value.indexOf(docum
ent.form.string.value));

}
</SCRIPT>
```

isFinite – Method

See also

IsFinite evaluates an argument and determines if the value is a finite number.

Syntax:

```
isFinite(number)
```

Parameter

The **number** mandatory argument is a numeric value.

Results

false if the value of **number** is **NaN**, a positive infinite or a negative infinite.
true in any other situation.

isNaN – Method

See also

isNaN evaluates an argument to determine if it is not a number (**NaN**, *Not a Number*). It returns **TRUE** or **FALSE**.

Syntax

```
isNaN(testValue)
```

Parameter

`testValue` is the value to be evaluated.

Results

true if the value is **NaN**.
false if it is not NaN.

Example

```
<FORM NAME="form">
Enter a number of an alphanumeric string:<BR>
<INPUT TYPE="text" NAME=enter MAXLENGTH=40 SIZE=40>
<BR>
<INPUT TYPE="button" VALUE="Testing!"
onClick="testnum(form.enter.value);")">
<BR>
</FORM>

<SCRIPT LANGUAGE="JavaScript">
function testnum(num)
{
inum=parseInt(num);
if(isNaN(inum))
     alert('This is a string!');
else
     alert('This is a numeric value!');
}
</SCRIPT>
```

italics – Method of object: String

See also

This method causes a string to be displayed in italic, as if it were in the **<I>** HTML tag. You must use the write or writeln methods to format and display the text string in the document.

Syntax

```
textName.italics()
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
string = new String("This is a display!");
document.writeln("normal display :"+string+"<BR>");
document.writeln("display in italic:"+string.italics());
</SCRIPT>
```

item – Method of object: Enumerator

See also

Returns the current element in a collection.

Syntax

```
myEnum.item( )
```

Parameter

Regardless of the object, Enumerator can be used as argument of the **item** method.

Results

The **item** method returns the current element.
When the collection is empty or the current element is not defined, it returns the value **undefined**.

javaEnabled – Method of object: navigator

See also

This method returns **true** if Java is enabled for the browser used and **false** if it isn't.

Syntax

```
navigator.javaEnabled()
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
if(navigator.javaEnabled)
     document.write("This browser is Java enabled.")
else
     document.write("This browser is not Java enabled.");
</SCRIPT>
```

join – Method of object: Array

See also

This method joins all elements of an array into a text string.

Syntax

```
arrayName.join(separator)
```

Parameters

arrayName is the name of a Array object or the property of an existing object.
separator specifies a character to separate each element of the array. If omitted, the array elements are separated with a comma (,).

Example

```
<SCRIPT LANGUAGE="JavaScript">
array = new Array()
array[0] = "welcome"
array[1] = "to"
array[2] = "the"
array[3] = "JavaScript"
array[4] = "tutorial"

document.write(array.join(" "));
```

lastIndexOf – Method of object: String

See also

The characters in a text string are indexed from left to right. The index of the first character is **0** and the index of the last character is **textName.length-1**, that is, the length of the text string name called minus **1**.

The **lastIndexOf** returns the index within the calling string of the last occurrence of the specified value.

If a value is not specified for the **fromIndex** argument, the default value is **0**. If the **SearchValue** is not found, JavaScript returns **-1**.

Syntax

```
textName.lastindexOf(SearchValue, [fromIndex] )
```

Parameters

textName is either a text string or a property of an existing object.
SearchValue is a character or a character string or a property of an existing object, representing the value to search for.
fromIndex is the location within the calling string to start the search from.
fromIndex can be any integer from 0 to **textName.length-1** or a property of an existing object.

Example

```
<FORM NAME="form">
text:<BR>
<INPUT TYPE="text" NAME=enter MAXLENGTH=40 SIZE=40 VALUE="Enter some text">
<BR>
String to search for:<BR>
<INPUT TYPE="text" NAME=string MAXLENGTH=40 SIZE=40 VALUE="e">
<BR>
<INPUT TYPE="button" VALUE="search" onClick="search()">
<BR>
Position of string:<INPUT TYPE="text" NAME=position MAXLENGTH=4 SIZE=4>
</FORM>
<SCRIPT LANGUAGE="JavaScript">
function search()
{
document.form.position.value=escape(document.form.enter.value.lastIndexOf(d
ocument.form.string.value));

}
</SCRIPT>
```

lbound – Method of object: VBArray

See also

This method returns the lowest index of a dimension of a VBArray object.

Syntax

```
safeArray.lbound(dimension)
```

Parameters

safeArray represents a VBArray object.
dimension is an optional parameter. It indicates the dimension of a VBArray object where you want to determine the inferior limit. If omitted, **lbound** acts as if value 1 has been passed.

link – Method of object: String

See also

Creates a hypertext link that requests another URL. This method must be used with the write or writeln methods to create and display a hyperlink in the document.

Links created using this method become elements in the links array. Refer to the Link object for more information on links arrays.

Syntax

```
string.link(hrefAttribute)
```

Parameters

string is either a text string or a property of an existing object.

href represents the value of the **HREF** attribute of the **<A>** element.

Example

```
<SCRIPT LANGUAGE="JavaScript">
ungi="A new Internet Guide!";
URL="http://www.imaginet.fr/ime/";
document.write("Learn with " + ungi.link(URL));
</SCRIPT>
```

load – Method of object: Layer

See also

Changes the source of a layer to the contents of the specified (URL) file and changes the width of the *layer* according to the second parameter.

Syntax

```
load(sourcestring, width)
```

Parameters

sourcestring represents a string initiating the URL of an external file that is the target for the contents to be inserted.
width represents the width of the *layer* expressed in pixels.

log – Method of object: Math

See also

Returns the natural logarithm (base e) of a number.

Syntax

```
Math.log(Number)
```

Parameter

Number is a positive numeric expression. If the **Number** value is out of this range, the value returned is always -1.797693134862316e+308.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Math.log(n)
value=2.32
document.write("The natural logarithm of variable 'value = "+value+"' is
<BR>Math.log
(value) = "+Math.log(value)+" .<BR><BR>");
</SCRIPT>
```

margins – Method of object: Style

See also

This method specifies the distance between the sides of an HTML element and the adjacent elements.

Syntax:

```
margins(top, right, bottom, left)
```

Parameters

top is a text string specifying the value of the **Style.marginTop** property.
right is a text string specifying the value of the **Style.marginRight** property.
bottom is a text string specifying the value of the **Style.marginBottom** property.
left is a text string specifying the value of the **Style.marginLeft** property.

match – Method of object: String

See also

Searches and returns a text string in a table.

Syntax:

```
textName.match(regExp)
```

Parameters

textName represents a text string in which the search is carried out.

rgExp represents the expression searched.

max – Method of object: Math

See also

Returns the greater of two numbers.

Syntax

```
Math.max(Number1, Number2)
```

Parameters

Number1 and **Number2** are numeric expressions.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Math.max(n1,n2)
value1=-4.268
value2=2.78
document.write("Between the variables 'value1 = "+value1+"' and 'value2 =
"+value2+"' the higher number is  <BR>Math.max(value1,value2) =
"+Math.max(value1,value2)+" .<BR><BR>");
</SCRIPT>
```

min – Method of object: Math

See also

Returns the lesser of two numbers.

Syntax

```
Math.min(Number1, Number2)
```

Parameters

Number1 and **Number2** are numeric expressions.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Math.min(n1,n2)
value=-0.268
document.write("Between the two variables 'value1 = "+value1+"' and 'value2
= "+value2+"' the lowest number is <BR>Math.min(value1,value2) =
"+Math.min(value1,value2)+" .<BR><BR>");
</SCRIPT>
```

moveAbove – Method of object: Layer

See also

Places the current *layer* on top of the specified *layer* in the stack order without modifying the horizontal and vertical position of the current *layer* in the page. After the operation, the two *layers* share the same *layer* parent.

Syntax:

```
moveAbove(aLayer)
```

Parameters

aLayer represents the *layer* on top of which the current *layer* is located in the stack.

moveBelow – Method of object: Layer

See also

Places the current *layer* below the specified *layer* in the stack order without modifying the horizontal and vertical position of the current *layer* in the page. After the operation, the two *layers* share the same *layer* parent.

Syntax:

```
moveBelow(aLayer)
```

Parameters

aLayer represents the *layer* under which the current *layer* is located in the stack.

moveBy – Method of objects: Layer and window

See also

This method is used to modify the position of a *layer*.

Syntax:

```
moveBy(horizontal, vertical)
```

Parameters

horizontal is the horizontal movement value expressed in pixels.
vertical is the vertical movement value expressed in pixels.

moveFirst – Method of object: Enumerator

See also

Move the current element to the first position in a collection.

Syntax:

```
myEnum.moveFirst( )
```

Parameter

myEnum represents any Enumerator object.

moveNext – Method of object: Enumerator

See also

Place the current element after another element in the collection.

Syntax:

```
myEnum.moveNext( )
```

Parameter

myEnum represents any Enumerator object.

moveTo – Method of objects: Layer and window

See also

Move the left upper corner of a window to specified screen coordinates.

Syntax

```
moveTo(x-coordinate, y-coordinate)
```

Parameters

x-coordinate is a number representing the window's top border, expressed in screen coordinates (pixels).
y-coordinate is a number representing the window's left border, expressed in screen coordinates (pixels).

moveToAbsolute – Method of object: Layer

See also

Change the position of the *layer* using an absolute positioning in the page, without taking into account the *layer* parent. The movement values are expressed in pixels.

Syntax

```
moveToAbsolute(x, y)
```

Parameters

x is a number representing the window's top border, expressed in pixels.
y is a number representing the window's left border, expressed in pixels.

Note

This method generates the same results that are obtained adjusting the pageX and pageY properties of the Layer object.

Number – Method

See also

Converts a specified object to a number. The **Number** function is a top-level function not associated to any particular object.

Syntax:

```
Number(obj)
```

Parameter

`obj` represents the object to be converted.

Example

If the object is Date, the value returned is presented in milliseconds, calculated as of January 01, 1970 UTC. The result is a positive value after that date and a negative value before that date.

If the object is a text string that doesn't contain a well defined numeric value, the value returned is **NaN**.

open – Method of object: document

See also

This method opens a document to insert in it the output of write and writeln methods.

Syntax:

```
open([mimeType, [replace]])
```

Parameters

mimeType is used to specify the following document types : text/html, text/plain, image/gif, image/jpg, image/x-bitmap, plugIn

replace. If this parameter is used, mimeTypes must have the **text/html** value. In this case, the new documents reuses the entry in the history list used in the previous document.

Example

```
Input the text that will be placed in the second window:
<FORM NAME=form>
<TEXTAREA NAME="input" ROWS=5 COLS=40></TEXTAREA>
<INPUT TYPE="button" VALUE="Click here" onClick="open()">
</FORM>
<SCRIPT LANGUAGE="JavaScript">
function open()
{
Window2=open("","","scrollbars=no,width=400,height=200");
Window2.document.open();
Window2.document.write("<HEAD><TITLE>second window!</TITLE></HEAD>");
Window2.document.write("<CENTER><BIG><B>"+document.form.output.value+"</
B></BIG>");
Window2.document.write("<FORM><INPUT TYPE='button' VALUE='close'
onClick='window.close()'></FORM>");
Window2.document.write("</CENTER>");
Window2.document.close();
}
</SCRIPT>
```

open – Method of object: window

See also

This method is used to open a new browser window.

Syntax

```
[windowVar] [window].open(URL, windowName [, window Options])
```

Parameters

windowVar	is the name of the new window. You must use this variable to reference the window's properties, methods and contents.
URL	indicate the URL to load in the new window. Refer to the Location object to obtain more details on the components of an URL.
windowName	is the window name that is specified as a value of the **TARGET** attribute for the **<FORM>** or **<A>** elements.
windowOptions	is a list of options for a window. If several options are defined, these must be separated by commas without spaces between the comma and the next option:
alwaysLowered	If the value specified is **yes**, this option creates a new floating window that will be displayed behind all the other browser windows, whether they are activated or not. This is a special option that must be used with a target script.. (JavaScript 1.2)
alwaysLowered	If the value specified is **yes**, this option creates a new floating window that will be displayed in front of all the other browser windows, whether they are activated or not. This is a special option that must be used with a target script.. (JavaScript 1.2)
dependent	If the value specified is **yes**, this option creates a child window of the current window. A child window closes when the parent window is closed. (JavaScript 1.2)
directories	If the value specified is **yes**, this option adds the standard browser buttons such as *What's New* and *What's Cool*.
height	indicates the window height in pixels. (JavaScript 1.0 and 1.1)
hotkeys	If the value specified is **no** (or 0), this option deactivates most of the shortcut keys of those windows that do not have menu bars. The application's security and close keys remain available all the time. (JavaScript 1.2)
innerHeight	Specifies the window's content area height in pixels. To create a window smaller than 100 x 100 pixels, use a signed script. Replaces the **Height** option, the latter must always comply with the norm for compatibility purposes (JavaScript 1.2).
innerWidth	Specifies the window's content area width in pixels. To create a window smaller than 100 x 100 pixels, use a signed script. Replaces the **Height** option, the latter must always comply with the norm for compatibility purposes (JavaScript 1.2).
location	If the value is set to **yes**, this option inserts an input field that is used to indicate the URL or the local access path of a Web page. This is the address bar.
menubar	If the value is set to **yes**, this option inserts a menu bar at the top of the window.
outerHeight	Specifies the window's vertical external size in pixels. To create a window smaller than 100 x 100 pixels, use a signed script (JavaScript 1.2).
personalbar	If the value is set to **yes**, this option inserts a personal toolbar which displays the user's personal bookmarks file. (JavaScript 1.2).
resizable	If the value is set to **yes**, this option allows the user to resize the window.
screenX	Indicate the window's position with relation to the left side of the screen. To place a window outside the screen limits, used a signed script (JavaScript 1.2).
screenX	Indicate the window's position with relation to the top of the screen. To place a window outside the screen limits, used a signed script (JavaScript 1.2).
scrollbars	If the value is set to **yes**, this option adds vertical and horizontal scrollbars when the content is larger than window's size.

status	If the value is set to **yes**, this option adds a status bar at the bottom of the window.
titlebar	If the value is set to **yes**, this option creates a window with a title bar. To define a window without a title bar (**no**), used a signed script (JavaScript 1.2).
toolbar	If the value is set to **yes**, this option adds a standard tools bar to the browser window.
width	Specify the window's width in pixels. (JavaScript 1.0 and 1.1)
z-lock	(JavaScript 1.2) If the value is set to **yes**, this option creates a new window that can be placed before the other browser windows while it is activated. To use this option the script must be signed (JavaScript 1.2).

Example

```
<SCRIPT LANGUAGE="JavaScript">
Window2=open("","Subscribe","scrollbars=yes,width=150,height=200,status=yes
,menubar=yes,location=yes,resizable=yes");
Window2.document.write("<HEAD><TITLE>second window!</TITLE></HEAD>")
Window2.document.write("<CENTER><BIG><B>This is the second
window!</B></BIG></CENTER>")
</SCRIPT>
```

paddings – Method of object: Style

See also

indicates the spacing value that must be inserted between the sides of an element and its content (text, image).

Syntax:

```
paddings(top, right, bottom, left)
```

Parameters

top is a string indicating the value of the Style.paddingTop property.
right is a string indicating the value of the Style.paddingRight property.
bottom is a string indicating the value of the Style.paddingBottom property.
left is a string indicating the value of the Style.paddingLeft property.

Note

Values can be expressed using the following units of measurement:
px, pt, in, %, pc, cm, mm

parse – Method of object: Date

See also

Returns the number in milliseconds elapsed between January 1st, 1970 00:00:00, local time, and a date specified in a text string.

Syntax

```
Date.parse(dateString)
```

Parameter

`dateString` is a text string representing a given date or a property of an existing object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Return the number of milliseconds elapsed since January 1st, 1970
// and a text string representing a given date or a property of an existing
object:
mytenthann = Date.parse("25 sep, 79 08:25:55")
document.write(mytenthann+" milliseconds have elapsed since January 1st,
1970 and my tenth anniversary (25 sep 1979 08:25:55)<BR>");
</SCRIPT>
```

parseFloat – Method

See also

Parses a string argument and returns a floating point number. The **parseFloat** function is not a method associated with an object, but part of the JavaScript language itself.

Syntax

```
parseFloat(textString)
```

Parameter

textString is a string of characters representing a value that you want to parse.

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.write(parseFloat("0.0314E+2")+"<BR>");
document.write(parseFloat("0314E+2")+"<BR>");
document.write(parseFloat("good morning")+"<BR>");
</SCRIPT>
```

parseInt – Method

See also

Parses a string argument and returns an integer of the specified radix or base. The **parseInt** function is not a method associated with an object, but part of the JavaScript language itself.

Syntax

```
parseInt(string [,radix])
```

Parameter

string is a string of characters representing a value that you want to parse.
radix is an integer that represents the radix of the returned value.

Example

```
<SCRIPT LANGUAGE="JavaScript">
document.write(parseInt("3.1416")+"<BR>");
document.write(parseInt("0x11")+"<BR>");
document.write(parseInt("FXX123", 16)+"<BR>");
document.write(parseInt("17", 8)+"<BR>");
document.write(parseInt("O55")+"<BR>");
document.write(parseInt("01101101", 2)+"<BR>");
document.write(parseInt("ungi", 2)+"<BR>");
</SCRIPT>
```

plugins.refresh – Method of object: navigator

See also

This method makes available embedded modules *plugins*) recently installed, and updates the corresponding arrays (ex.: **Plugin array**) and optionally refreshes the open documents containing the embedded modules.

This method can be called using one of the following instructions:

```
navigator.plugins.refresh(true)
navigator.plugins.refresh(false)
```

If **true** is used, **plugins.refresh** makes available embedded modules (*plugins*) recently installed, updates the corresponding arrays, and refreshes the documents that are opened that contain the embedded modules.

If **false** is used, the corresponding arrays are updated but open documents containing the embedded modules are not refreshed.

pop - Method of object: Array

See also

Removes the last element of an Array object and returns it as a result. The length of the object is decreased by 1.

Syntax

```
arrayName.pop()
```

arrayName represents the Array object name created.

pow – Method of object: Math

See also

Returns base to the exponent power.

Syntax

```
Math.pow(base, exponent
```

Parameters

base is numeric expression.
exponent is a numeric expression. If the resulting number is not feasible, the value returned is zero.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Math.pow(base,exponent)
base=5.268
exponent = 2.38
document.write("The power of base <FONT SIZE
+1>"+base+"</FONT><SUP>"+exponent+"</SUP> is Mth.pow(base,exponent) <BR> is
Math.pow("+base+","+exponent+")= "+Math.pow(base,exponent)+" .<BR><BR>");
</SCRIPT>
```

preference – Method of object: navigator

See also

Allows a signed script to obtain and adjust certain Web browser preferences (software options).

Syntax

```
preference(prefName[, setValue])
```

Parameters

`prefName` is a string representing the name of an option (preference) to choose and change Options that can be modified are shown in the table below.
`setValue` is the value assigned to the option. This value could be a string, a number or a Boolean value.

Note

This method returns the value of an option. If you use this method to modify an option, it returns a new value.
Read a value using this method requires the *UniversalPreferencesRead* privilege.
Adjust a value using this method requires the *UniversalPreferencesRead* privilege.

To modify…	use…	Value...
Automatically load images	`general.always_load_images`	**true** or **false**
Enable Java	`security.enable_java`	**true** or **false**
Enable JavaScript	`javascript.enabled`	**true** or **false**
Enable style sheets	`browser.enable_style_sheets`	**true** or **false**
Enable SmartUpdate	`autoupdate.enabled`	**true** or **false**
Accept all cookies	`network.cookie.cookieBehavior`	**0**
Accept only cookies that get sent back to the originating server	`network.cookie.cookieBehavior`	**1**
Disable cookies	`network.cookie.cookieBehavior`	**2**
Warn before accepting a cookie	`network.cookie.warnAboutCookies`	**true** or **false**

print – Method of object: window

See also

This method is used to print the contents of a window.

Syntax

```
print()
```

prompt – Method of object: window

See also

Displays a dialog box with a message and an input field.
Even though the **prompt** method employs the window object, you do not need to specify a window reference to call it. For example, `windowReference.promt()` isn't necessary.

Syntax

```
prompt(Message, [defaultValue])
```

Parameters

`Message` is a character string or a property of an existing object. The text string represents the message.
`defaultValue` is a text string, an integer or a property of an existing object that represents the default value of the entry field. If you do not indicate an initial value for the `defaultValue`, the dialog box will display the value: <undefined>.

Example

```
<FORM>
<INPUT TYPE="button" VALUE="open" onClick="askquestion()">
</FORM>
<SCRIPT LANGUAGE="JavaScript">
function askquestion()
{
answer=prompt('Would you like to open a second window?\n Enter the text to
be displayed in that window:','your text');

if(answer)
     {
     Window2=open("","second","scrollbars=no,width=150,height=200");
     Window2.document.write("<HEAD><TITLE>/second window!</TITLE></HEAD>")

Window2.document.write("<CENTER><BIG><B>"+answer+"</B></BIG></CENTER>")
}
}
</SCRIPT>
```

push - Method of object: Array

See also

Add the object or objects found in arguments (elements) to the Array object.

Syntax

```
arrayName.push(Objet1, Objet2, …, Objetn)
```

Parameters

arrayName represents the Array object created.
Objet1 to **Objetn** represent the objects added to the **arrayName** object.

random – Method of object: Math

See also

Returns a pseudo-random number between 0 and 1.

Syntax

```
Math.random()
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Math.random()
document.write("This method returns a random number between 0 and 1 (very
useful for games!<BR> example : Math.random() = "+Math.random()
+" .<BR><BR>");
</SCRIPT>
```

releaseEvents – Method of objects: document, Layer and window

See also

Indicate to a targeted window or document to release the capture of a event type and to send the event to objects that normally are in charge.

Syntax

```
releaseEvents(eventType1 [|eventTypeN...])
```

Parameters

eventType1...eventTypeN represents the type of event to be captured.

reload – Method of object: Location

See also

Forces a reload of the window's current document specified by the URL of the location.href property. The **reload** acts exactly like the browser's Reload button.

Syntax:

```
location.reload([true])
```

Parameter

true forces an unconditional HTTP GET of the document from the server. This should not be used unless you have reason to believe that disk and memory caches are off or broken, or the server has a new version of the document.

Example

```
reload this page!
<FORM>
<INPUT TYPE="button" VALUE="Reload!"
onClick="window.location.reload(true)">
</FORM>
```

replace – Method of objects: Location and String

See also

This method loads (and erases) the specified URL over the current history entry. After calling the **replace**, method, the user cannot navigate to the previous URL by using the browser's BACK button.

Syntax

```
location.replace("URL")
```

URL specifies the URL to load.

Example

```
Replace the current URL with UNGI's!
<FORM>
<INPUT TYPE="button" VALUE="Replace URL" onClick="window.location.replace
('http://www.imaginet.fr/ime/')">
</FORM>
```

reset – Method of object: Form

See also

This method returns all the fields in a form to their initial values. Simulates a mouse click on a reset (*reset*) button for the form.

Syntax

```
formName.reset()
```

Parameter

formName is the name of a form or an element in a tables array.

Example

```
<FORM NAME=form>
<BR>
<TEXTAREA NAME="box" ROWS=5 COLS=40></TEXTAREA>
<BR>
<INPUT TYPE="button" VALUE="Erase" onClick="erase()");
<BR>
</FORM>
<SCRIPT LANGUAGE="JavaScript">
document.form.box.value = "This is text that will be erased ";
function erase()
{
reponse=confirm('Do you really want to erase the form ? ');
if(response=true)
     {
     document.form.reset();
     }
}
</SCRIPT>
```

resizeBy – Method of objects: Layer and window

See also

Used with the Layer object.
Resizes the *layer* by the specified height and width values for the *layer*.

Syntax:
```
resizeBy(width, height)
```

Parameters
width is the value to be assigned to the *layer* , expressed in pixels.
height is the value to be assigned to the *layer* , expressed in pixels.

Note
This method is the same as assigning **width** and **height** to clip.width and clip.height.

Used with the window object:
Resizes the window by moving its right lower corner.

Syntax:
```
resizeBy(horizontal, vertical)
```

Parameters
horizontal represents the number of pixels by which to resize the window horizontally.
vertical The number of pixels by which to resize the window vertically.

resizeTo – Method of objects: Layer and window

See also

Used with the Layer object:
Resize the *layer* to have the specified height and width values.

Syntax:
```
resizeTo(width, height)
```

Parameters
width represents the width of the *layer* expressed in pixels.
height represents the height of the *layer* expressed in pixels.

Note
This method is the same as using clip.width and clip.height.

Used with the window object:
Resize the window to have the specified height and width values.

Syntax:
```
resizeTo(outerWidth, outerHeight)
```

Parameters
outerWidth is the window's horizontal size expressed in pixels.
outerHeight is the window's vertical size expressed in pixels.

reverse – Method of object: Array

See also

Transposes the elements of an array, that is, the first array element becomes the last and the last becomes the first.

Syntax

```
arrayName.reverse()
```

Parameter

arrayName is the name of a Array object or the property of an existing object.

Example

```
<FORM NAME=form>
<BR>
<INPUT TYPE="text" NAME=table MAXLENGTH=60 SIZE=60>
<BR>
<INPUT TYPE="button" VALUE="Reverse" onClick="reverse()");
<BR>
</FORM>
<SCRIPT LANGUAGE="JavaScript">
array = new Array()
array[0] = "Welcome"
array[1] = "to"
array[2] = "Ungi's"
array[3] = "Javascript"
array[4] = "tutorial"


document.form.array.value = array.join();
function reverse()
{
array.reverse();
document.form.array.value = array.join();

}
</SCRIPT>
```

round – Method of object: Math

See also

Returns the value of a number rounded to the nearest integer.

Syntax

```
Math.round(Number)
```

Parameter

`Number` is a numeric expression.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Math.round(n)
value=15.5
document.write("Round the variable 'value'= "+value+"' to the nearest
integer <BR>Math.round(value) = "+Math.round(value)+" .<BR><BR>");
</SCRIPT>
```

routeEvent – Method of objects: document, Layer and window

See also

Passes a captured event along the normal event hierarchy, i.e. to a higher specialized function.

Syntax

```
routeEvent(event)
```

Parameters

`event` is the name of the event to be routed.

savePreferences – Method of object: navigator

See also

Saves the Navigator preferences (user's preferences) to the local file *preferences.js*.

Syntax

```
SavePreferences()
```

Note

Saving user preferences requires the *UniversalPreferencesWrite* privilege.

scroll – Method of object: window

See also

This method scrolls a window to specified coordinates. The coordinates of the left upper corner of a window are: (0,0).

Syntax

```
windowReference.scroll(x-coordinate,y-coordinate)
```

Parameters

windowReference is a valid way to reference a window, as it is described in the window object.
x-coordinate is an integer representing the x-coordinate in pixels (width).
y-coordinate is an integer representing the y-coordinate in pixels (height).

Note

The scroll method is no longer in used after JavaScript 1.2 and has been replaced by scrollBy and scrollTo methods.

Example

```
<CENTER>
Choose direction:<BR>
<FORM NAME="form">
<INPUT TYPE="button" VALUE="top" onClick="top()"><BR>
<INPUT TYPE="button" VALUE="left" onClick="left()">
<INPUT TYPE="button" VALUE=right" onClick="right()"><BR>
<INPUT TYPE="button" VALUE="bottom" onClick="bottom()"><BR>
<BR>
</FORM>
</CENTER>
<SCRIPT LANGUAGE="JavaScript">
Window2=open("birds.jpg","","scrollbars=yes,width=200,height=200");
maxx = Window2.document.images[0].width;
maxy = Window2.document.images[0].height;
x = maxx/2;
y = maxy/2;
Window2.scroll(x,y);

function top()
{
y-=10;
if(y<0) y=0;
Window2.scroll(x,y);
}

function left()
{
x-=10;
if(x<0) x=0;
Window2.scroll(x,y);
}

function right()
{
x+=10;
```

```
if(x>maxx) x=maxx;
Window2.scroll(x,y);
}

function bottom()
{
y+=10;
if(y>maxy) y=maxy;
Window2.scroll(x,y);
}

</SCRIPT>
```

scrollBy – Method of object: window

See also

Scrolls the viewing area of a window by the specified amount, by adding or subtracting a number of pixels to the current scroll location.

Syntax

```
scrollBy(horizontal, vertical)
```

Parameters

horizontal The number of pixels (added or subtracted) by which to scroll the viewing area horizontally.
vertical The number of pixels (added or subtracted) by which to scroll the viewing area vertically.

Note

The window.scrollbars property must be set to **true** for the **scrollBy** method to have an effect.

scrollTo – Method of object: window

See also

Scrolls the viewing area of the window so that specified coordinates become the top-left corner.

Syntax

```
scrollTo(x-coordinate, y-coordinate)
```

Parameters

x-coordinate is a number expressed in pixels representing the x-coordinate of the display area.
y-coordinate is a number expressed in pixels representing the y-coordinate of the display area.

Note

The window.scrollbars property must be set to **true** for the **scrollTo** method to have an effect.

search – Method of object: String

See also

Executes the search for a match between a regular expression inside a text string.

Syntax

```
textName.search(rgExp)
```

Parameters

textName represents a text string in which the search is carried out.
rgExp represents the expression searched.

select – Method of objects: FileUpload, Password, Text and Textarea

See also

This method is used to select the input field of the specified object, either Password, Text or Textarea. This object is an element in a form.

Syntax

```
passwordName.select()
textName.select()
textareaName.select()
```

Parameters

passwordName is the value of attribute NAME of the Password object or an element in an elements array.
textName is the value of attribute NAME of the Text object or an element in an elements array.
textareaName is the value of attribute NAME of the Textarea object or an element in an elements array.

Example

```
<FORM NAME=form>
<BR>
<TEXTAREA NAME="box" ROWS=5 COLS=40>This is the text to select</TEXTAREA>
<BR>
<INPUT TYPE="button" VALUE="Select" onClick="select()");
<BR>
</FORM>
<SCRIPT LANGUAGE="JavaScript">
function select()
{
document.form.box.focus();
document.form.box.select();
}
</SCRIPT>
```

setDate – Method of object: Date

See also

Sets the day of the month for a specified date (local time).

Syntax

```
dateObjectName.setDate(dayValue)
```

Parameters

dateObjectName is the name of a Date object or the property of an existing object.
dayValue is an integer between 1 and 3, or a property of an existing object representing the day of the month.

Example

```
<SCRIPT LANGUAGE="JavaScript">
today = new Date();
// Set the day of the month to the date (1 à 31):
// datehour.setDate(dateValue)
today.setDate(25)
document.write(today.toLocaleString()+" today is the 25th  =>
today.setDate(25)<BR>");
</SCRIPT>
```

setFullYear – Method of object: Date

See also

Sets the full year for a specified date (local time).

Syntax

```
dateObjectName.setFullYear(value)
```

Parameters

dateObjectName is the name of a Date object or the property of an existing object.
value is a four- digit number (**1999**), or three values separated by commas (**1999, 02, 21**).

setHotKeys – Method of object: window

See also

Enables or disables hot keys in a window which does not have menus.

Syntax:

```
setHotKeys(trueOrFalse)
```

Parameters

`trueOrFalse` is a Boolean value specifying whether hot keys are enabled:
true enables shortcut keys
false disables shortcut keys

Note

To enable or disable shortcut keys, the *UniversalBrowserWrite* privilege is required.

setHours – Method of object: Date

See also

Sets the hours for a specified date (local time).

Syntax

```
dateObjectName.setHours(hoursValue)
```

Parameters

dateObjectName is the name of a Date object or the property of an existing object.
hoursValue An integer between 0 and 23, or a property of an existing object, representing the hour.

Example

```
<SCRIPT LANGUAGE="JavaScript">
today = new Date();
// Set the hour according to the time (0 to 23):
// today.setHours(hoursValue)
today.setHours(8)
document.write(today.toLocaleString()+"It is 8 hours  =>
today.setHours(8)<BR>");
</SCRIPT>
```

setInterval – Method of object: window

See also

Evaluates an expression or calls a function every time a specified number of milliseconds elapses, or until cancelled by a call to the clearInterval method.

Syntax

```
setInterval(expression, msec)setInterval(function, msec[, arg1[, ..., argN]])
```

Parameters

`function`	Any function.
`expression`	A string containing a Javascript expression. The string must be placed between quotation marks.
`msec`	A numeric value or a numeric string, expressed in milliseconds.
`arg1, ..., argn`	The arguments, if any, passed to function.

setMilliseconds – Method of object: Date

See also

Sets the milliseconds for a specified date.

Syntax

```
dateObjectName.setMilliseconds(millisecondsValue)
```

Parameters

dateObjectName is the name of a Date object or the property of an existing object.
millisecondsValue is an integer between 0 and 999. If you specify a number higher than 999, the seconds value will be increased and the milliseconds value will be updated accordingly.

setMinutes – Method of object: Date

See also

Sets the minutes for a specified date (local time).

Syntax

```
dateObjectName.setMinutes(minutesValue)
```

Parameters

dateObjectName is the name of a Date object or the property of an existing object.
minutesValue is an integer between 0 and 59 or a property of an existing object representing minute units.

Example

```
<SCRIPT LANGUAGE="JavaScript">
today = new Date();
// Set minutes in time  (0 to 59):
// today.setMinutes(minutesValue)
today.setMinutes(37)
document.write(today.toLocaleString()+" 37 minutes  =>
today.setMinutes(37)<BR>");
</SCRIPT>
```

setMonth – Method of object: Date

See also

Sets the month for a specified date (local time).

Syntax

```
dateObjectName.setMonth(monthValue)
```

Parameters

dateObjectName is the name of a Date object or the property of an existing object.
monthValue is an integer between 0 and 11 or a property of an existing object, representing a month in the year.

Example

```
<SCRIPT LANGUAGE="JavaScript">
today = new Date();
// Set the month according to the date (0 to 11):
// today.setMonth(monthValue)
today.setMonth(8)
document.write(today.toLocaleString()+" It is September =>
today.setMonth(8)<BR>");
</SCRIPT>
```

setResizable – Method of object: window

See also

Specifies whether a user is permitted to resize a window.

Syntax:

```
setResizable(trueOrFalse)
```

Parameters

`trueOrFalse` is a Boolean value:
true allows the user to modify the window size.
false prevents the user from modifying the window size.

setSeconds – Method of object: Date

See also

Sets the seconds for a specified date (local time).

Syntax

```
dateObjectName.setSeconds(secondsValue)
```

Parameters

dateObjectName is the name of a Date object or the property of an existing object.
secondsValue is an integer between 0 and 59 or a property of an existing object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
today = new Date();
// Set the seconds for the time (0 to 59):
// today.setSeconds(secondsValue)
today.setSeconds(36)
document.write(today.toLocaleString()+" 36 seconds  =>
today.setSeconds(36)<BR>");
</SCRIPT>
```

setTime – Method of object: Date

See also

Sets the value of the Date object (local time).

Syntax

```
dateObjectName.setTime(timeValue)
```

Parameters

dateObjectName is the name of a Date object or the property of an existing object.
timeValue is an integer or a property of an existing object, representing the number of milliseconds elapsed since January 1st., 1970 00:00:00.

Example

```
<SCRIPT LANGUAGE="JavaScript">
today = new Date();
// Set a complete date according to the number of seconds elapsed
// since Jan. 1st,  1970 à 00:00:00
// today.setTime(timeValue)
mesdixans = Date.parse("25 sep, 79 08:25:55")
today.setTime(mesdixans)
document.write("return the date according to the milliseconds elapsed since
01/01/1970 :<BR> today.settime(mesdixans) = "+today.toLocaleString()
+"<BR>");
</SCRIPT>
```

setTimeout – Method of object: window

See also

This method is used to evaluate an expression after a specified number of milliseconds elapses.

Syntax

```
TimeoutID=setTimeout(expression, msec)
```

Parameters

TimeoutID is an identifier only used to cancel an evaluation with the clearTimeout method.
expression is a text string or a property of an existing object.
msec is a numeric value, a numeric string or a property of an existing object in milliseconds units.

Example

```
<FORM>
Here clicking triggers a counter; after 5000 milliseconds an alert is
issued:
<INPUT TYPE="checkbox" VALUE="help" NAME="help" onClick="Timer()">
</FORM>
<SCRIPT LANGUAGE="JavaScript">
function Timer()
{
timeoutID = window.setTimeout("window.alert('5 seconds have
elapsed.')",5000);
}
</SCRIPT>

<FORM>
Oups ! Timeout cancelled!
<INPUT TYPE="checkbox" VALUE="help" NAME="help"
onClick="window.clearTimeout(timeoutID)">
</FORM>
```

setUTCDate – Method of object: Date

See also

Sets the day of the month for a specified date according to the UTC standard (universal time).

Syntax

```
dateObjectName.setUTCDate(dayValue)
```

Parameters

dateObjectName is the name of a Date object or the property of an existing object.
dayValue is an integer between 1 and 3, or a property of an existing object representing the day of the month.

setUTCFullYear – Method of object: Date

See also

Sets the full year for a specified date according to the UTC standard (universal time).

Syntax

```
dateObjectName.setUTCFullYear(value)
```

Parameters

`dateObjectName` is the name of a Date object or the property of an existing object.
`value` is a four-digit number (`1999`), or three values separated by commas (`1999, 02, 21`).

setUTCHours – Method of object: Date

See also

Sets the hour for a specified date according to the UTC standard (universal time).

Syntax

```
dateObjectName.setUTCHours(hoursValue)
```

Parameters

dateObjectName is the name of a Date object or the property of an existing object.
hoursValue An integer between 0 and 23, or a property of an existing object, representing the hour.

setUTCMilliseconds – Method of object: Date

See also

Sets the milliseconds for a specified date according to the UTC standard (universal time).

Syntax

```
dateObjectName.setUTCMilliseconds(millisecondsValue)
```

Parameters

dateObjectName is the name of a Date object or the property of an existing object.
millisecondsValue is an integer between 0 and 999. If you specify a number higher than 999, the seconds value will be increased and the milliseconds value will be updated accordingly.

setUTCMinutes – Method of object: Date

See also

Sets the minutes for a specified date according to the UTC standard (universal time).

Syntax

```
dateObjectName.setUTCMinutes(minutesValue)
```

Parameters

dateObjectName is the name of a Date object or the property of an existing object.
minutesValue is an integer between 0 and 59 or a property of an existing object representing minute units.

setUTCMonth – Method of object: Date

See also

Sets the month for a specified date according to the UTC standard (universal time).

Syntax

```
dateObjectName.setUTCMonth(monthValue)
```

Parameters

dateObjectName is the name of a Date object or the property of an existing object.
monthValue is an integer between 0 and 11 or a property of an existing object, representing a month in the year.

setUTCSeconds – Method of object: Date

See also

Sets the seconds for a specified date according to the UTC standard (universal time).

Syntax

```
dateObjectName.setUTCSeconds(secondsValue)
```

Parameters

dateObjectName is the name of a Date object or the property of an existing object.
secondsValue is an integer between 0 and 59 or a property of an existing object.

setYear – Method of object: Date

See also

Sets the full year for a specified date (local time).

Syntax

```
dateObjectName.setYear(yearValue)
```

Parameters

dateObjectName is the name of a Date object or the property of an existing object.
yearValue is an integer higher than 1900 or a property of an existing object.

Note

In view of the 2000 year bug, it is best to use the setFullYear method.

Example

```
Exemple <SCRIPT LANGUAGE="JavaScript">
today = new Date();
// /set the full year date (number of years after 1900) :
// today.setYear(yearValue)
today.setYear(1987)
document.write(today.toLocaleString()+" This is 1987  =>
today.setYear(87)<BR>");
</SCRIPT>
```

setZOptions – Method of object: window

See also

Specifies the z-order stacking behavior of a window.

Syntax

```
setZOptions(windowPosition)
```

Parameters

windowPosition is a string evaluating to any of the following values:
alwaysRaised creates a new window that floats on top of other windows, whether it is active or not.
alwaysLowered creates a new window that floats below other windows, whether it is active or not.
z-lock creates a new window that does not rise above other windows when activated.

shift – Method of object: Array

See also

Removes the first element from an Array object and returns it as a result. The length of the object is decreased by 1.

Syntax

```
arrayName.shift()
```

arrayName represents the Array object name created.

sin – Method of object: Math

See also

Returns the sine of a number. The sin method returns a numeric value between -1 and 1, which represents the value of the angle.

Syntax

```
Math.sin(Number)
```

Number is a numeric expression which represents the size of an angle in degrees.

slice (Array) – Method of object: Array

See also

This method extracts a number of elements from an Array object and creates a new Array object with them. This method does not modify the (length) of the Array object.

Syntax

```
arrayName.slice(begin, end)
```

Parameters

arrayName represents the Array object.
begin represents the starting element from the group of elements from which to extract the Array object. This value is counted as of element 0.
end represents the ending element from the group of elements from which to extract the Array object. This value is counted as of element 0 or –0. If **end** is omitted, the elements following the **begin** element will be extracted.

slice (String) – Method of object: String

See also

Extracts a section of a string and returns a new string.

Syntax

```
textName.slice(beginSlice, endSlice)
```

Parameters

textName represents a text string in which is found the sub-string to be extracted.
beginSlice represents the position of the first character of the sub-string, calculated starting at **0**.
endSlice represents the position of the last character in the sub-string, calculated staring at **0** or –**0**.

small – Method of object: String

See also

Causes a string to be displayed in a small font, as if it were in a **<SMALL>** tag. You must use the write or writeln methods to format and display the text string in the document.

Syntax

```
textName.small()
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
string = new String("This is a display!");
document.writeln("normal display:"+string+"<BR>");
document.writeln("display with a small font:"+string.small());
</SCRIPT>
```

sort – Method of object: Array

See also

This method sorts the elements in an array according to a class order.

Syntax

```
arrayName.sort(compareFunction)
```

Parameters

arrayName is the name of a Array object or the property of an existing object.
compareFunction Specifies a function that defines the sort order. If omitted, the array will be sorted lexicographically.

Example

```
<FORM NAME=form>
<BR>
<INPUT TYPE="text" NAME=table MAXLENGTH=60 SIZE=60>
<BR>
<INPUT TYPE="button" VALUE="sort" SIZE=10 onClick="sorted()");
<BR>
</FORM>
<SCRIPT LANGUAGE="JavaScript">
array = new Array()
array[0] = "Welcome"
array[1] = "to"
array[2] = "Ungi's"
array[3] = "Javascript"
array[4] = "tutorial"


document.form.array (in some cases).value = array (in some cases).join();
function sort()
{
table.sort();
document.form.array (in some cases).value = array (in some cases).join();

}
</SCRIPT>
```

splice - Method of object: Array

See also

Adds and/or removes elements to an Array object.

Syntax

```
arrayName.splice(begin, number, element1, element2, …, elementn)
```

Parameters

arrayName represents the Array object.
begin represents the index number of the element from which the elements are removed.
number represents the number of elements removed.
element1 to **elementn** represents the elements added to the Array object.

split – Method of object: String

See also

This methods splits a String object into sub-strings in an array.

Syntax

```
stringName split([separator][, limit])
```

Parameters

stringName is any text string or a property of an existing object.

separator Specifies the character to use for separating the string. If the separator is omitted, the array returned contains an element consisting of the entire string.

limit is a number specifying a limit on the number of splits to be found.

Example

```
<FORM NAME=form>
<BR>
<INPUT TYPE="text" NAME=input value="Enter a string to be split"
MAXLENGTH=60 SIZE=60>
<BR>
<INPUT TYPE="button" VALUE="convert to an array" SIZE=10
onClick="separate()");
<BR>
<BR>
<TEXTAREA NAME="results" ROWS=5 COLS=40></TEXTAREA>
</FORM>
<SCRIPT LANGUAGE="JavaScript">
array (in some cases) = new Array()
function separate()
{
string = new String();
array (in some cases) = document.form.input.value.split(" ");
for(i=0;i<array (in some cases).length;i++)
{
     string = string + "element["+i+"] = "+ array (in some cases)[i] +"\n";
}
document.form.results.value = string;
}
</SCRIPT>
```

sqrt – Method of object: Math

See also

Returns the square root of a number.

Syntax

```
Math.(Number)
```

Number must be a positive numeric expression. If **Number** is out of range, the value returned will be zero.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Math.sqrt(n)
value=-9
document.write("The square root of variable 'value = "+value+"' is
<BR>Math.
sqrt(value) = "+Math.sqrt(value)+" .<BR><BR>");
</SCRIPT>
```

stop – Method of object: window

See also

Stops the current download.

Syntax

```
stop()
```

Note

This method has the same effect as using the Stop button in the browser.

strike – Method of object: String

See also

This method causes a string to be displayed as struck-out text, as the **<STRIKE>** tag used in HTML. You must use the write or writeln methods to format and display the text string in the document.

Syntax

```
textName.strike()
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
string = new String("This is a display!");
document.writeln("normal display :"+string+"<BR>");
document.writeln("strikethrough text:"+string.strike());
</SCRIPT>
```

String – Method

See also

Converts the specified object in a string.

Syntax

```
String(Object)
```

Parameters

obj represents the object to be converted.

Note:

The string function is a top-level JavaScript function not associated to any particular object.

Example:

When the object is a Date object, **String** returns a more readable string representation of the date.

sub – Method of object: String

See also

Causes a string to be displayed as a *subscript*, as if it were the **<SUB>** tag used in HTML. You must use the write or writeln methods to format and display the text string in the document.

Syntax

```
textName.sub()
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
string = new String("This is a display!");
document.writeln("normal display :"+string+"<BR>");
document.writeln("displayed as subscript:"+string.sub());
</SCRIPT>
```

submit – Method of object: Form

See also

This method is used to submit a form. It works exactly as the (SUBMIT) button.

Syntax

```
formName.submit()
```

formName is the name of a form or an element in a tables array.

substr – Method of object: String

See also

Returns a sub-group of a text string.

Syntax

```
textName.substr(start, length)
```

Parameters

textName represents a text string.
start represents the position of the first character of the sub-string, calculated starting at position **0**.
length is the number of characters to extract, calculated from the **start** value.

Notes

If **start** is higher than **0** and **length** is higher than the number of remaining characters, the method returns no characters.
If **length** is equal to **0**, the method returns no characters.
If **length** has no value assigned, the characters following **start** are returned.

substring – Method of object: String

See also

The characters in a text string are indexed from left to right. The index of the first character is zero, and the index of the last character is **textName.length-1**, that is, the length of the text string name called minus 1.

The **substring** method returns a subset of a string object.

Syntax:

```
textName.substring(indexA, indexB)
```

Parameters

indexA can be any integer between **0** and **textName.length-1** or a property of an existing object.
indexB can be any integer between **0** and **textName.length-1** or a property of an existing object.

If **indexA** is smaller than **indexB**, the **substring** method will return the section beginning with **indexA** and will end with the **indexB** preceding character. If **indexA** is greater than **indexB**, the **substring** method will return the section beginning with **indexB** and will end with the **indexA** preceding character. If **indexA** is equal to **indexB**, the **substring** method returns an empty string.

Example

```
<FORM NAME="form">
<INPUT TYPE="text" NAME="original" MAXLENGTH=40 SIZE=40 VALUE="text to be
separated">
<INPUT TYPE="button" VALUE="Divide" onClick="divide()">
<BR>
results:<BR>
<INPUT TYPE="text" NAME="Results" MAXLENGTH=40 SIZE=40 VALUE="">
</FORM>

<SCRIPT LANGUAGE="JavaScript">
function divide()
{
half = (document.form.original.value.length)/2;
document.form.results.value =
document.form.original.value.substring(0,half);
}
```

sup – Method of object: String

See also

Causes a string to be displayed as a *superscript*, as if it were the **<SUP>** tag used in HTML. You must use the write or writeln methods to format and display the text string in the document.

Syntax

```
textName.sup()
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
string = new String("This is a display!");
document.writeln("normal display :"+string+"<BR>");
document.writeln("displayed as superscript:"+string.sup());
</SCRIPT>
```

taint – Method

See also

This function adds "tainting " to a property. For more information on this function, refer to "data tainting"

Note

However, it is highly recommended that this function not be used as it not very reliable. This method was discontinued after JavaScript 1.2.

Syntax

```
taint(propertyName)
```

propertyName is the property that receives the **taint** function.

taintEnabled – Method of object: navigator

See also

Indicates if data tainting is enabled. The tainting function prevents other scripts from passing information that should be secure and private: directory structure, user navigation session history, etc.

However, it is highly recommended that this function not be used as it not very reliable. This method was discontinued after JavaScript 1.2.

Syntax

```
navigator.taintEnabled()
```

tan – Method of object: Math

See also

Returns the tangent of a number (an angle).

Syntax

```
Math.tan(Number)
```

Number is a numeric expression which represents the size of an angle in degrees.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Math.tan(n)
value=1.56
document.write("The tangent of angle 'value = "+value+"' is <BR>Math.tan
(value) = "+Math.tan(value)+".<BR><BR>");
</SCRIPT>
```

test – Method of objects: RegExp and Regular Expression

See also

Executes the search for a match between a regular expression and a specified string. It returns **TRUE** or **FALSE**.

Syntax

```
regexp.test([str])
```

Parameters

regexp represents the name of the regular expression.

str represents the string to be compared with the regular expression. If omitted, the value of the RegExp.input is used.

toArray – Method of object: VBArray

See also

This method is used to convert a VBArray multidimensional object into a Javascript single dimension array. The method returns a standard JavaScript array.

Syntax:

```
safeArray.toArray( )
```

safeArray is a VBArray object.

toGMTString – Method of object: Date

See also

Converts a date to a text string. This method uses the Internet GMT convention. The value format returned depends on the platform used.

Syntax

```
dateObjectName.toGMTString()
```

dateObjectName is the name of a Date object or the property of an existing object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Convert a date object into a text string using the GMT convention
// Date.toGMTString()
today = new Date();
stringGMT = today.toGMTString()
document.writeln("As per GMT, today is :"+string);
</SCRIPT>
```

toLocaleString – Method of object: Date

See also

Converts a date to a text string. This method uses the local convention.

Syntax

```
dateObjectName.toLocaleString()
```

dateObjectName is the name of a Date object or the property of an existing object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Convert a date object into a text string using the local convention
// Date.toGMTString()
today = new Date();
datelocale = today.toLocaleString()
document.writeln("According to local time it is: "+datelocale);
</SCRIPT>
```

toLowerCase – Method of object: String

See also

This method returns a text string (**textName**) into lowercase. It does not affect the **textName** value.

Syntax

```
textName.toLowerCase()
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
string = new String("ThIS is A STRIng IN LowERcaSE AND UPPerCaSe!");
document.writeln("normal display :"+string+"<BR>");
document.writeln("displayed in lowercase:"+string.toLowerCase());
</SCRIPT>
```

toSource - Method of objects: Array, Boolean, Date, Function, Number, Object, RegExp and String

See also

Returns a string representing the source code of the object.

Syntax

```
Object.toSource()
```

toString - Method of objects: Array, Boolean, Date, Function, JavaArray, Number, Object, RegExp, and String

See also

Converts an object into a text string.

Syntax

```
Object.toString([radix])
```

`Object` is the object to be converted to a text string.
`[radix]` indicates the base used to represent numeric values.

Example

```
Binary converter:<BR>
<FORM NAME=form>
<BR>
Enter a number:
<INPUT TYPE="text" NAME=input value="255" MAXLENGTH=10 SIZE=10>
<BR>
<INPUT TYPE="button" VALUE="Convert" SIZE=10 onClick="binary()");
<BR>
<BR>
<INPUT TYPE="text" NAME=results MAXLENGTH=60 SIZE=60>
</FORM>
<SCRIPT LANGUAGE="JavaScript">
function binary()
{
number=parseInt(document.form.input.value);
if(isNaN(number))
     {
     alert('This is not a number!');
     return;
}
document.form.results.value = number.toString(2);
}
</SCRIPT>
```

toUpperCase – Method of object: String

See also

This method returns a text string (**textName**) into uppercase. It does not affect the **textName** value.

Syntax

```
textName.toUpperCase()
```

Example

```
<SCRIPT LANGUAGE="JavaScript">
string = new String("ThIS is A STRIng IN LowERcaSE AND UPPerCaSe!");
document.writeln("normal display :"+string+"<BR>");
document.writeln("displayed in uppercase:"+string.toUpperCase());
</SCRIPT>
```

toUTCString – Method of object: Date

See also

Converts the value of the Date object into a string, using the UTC standard (universal time).

Syntax

```
dateObjectName.toUTCString()
```

dateObjectName is the name of a Date object or the property of an existing object.

ubound – Method of object: VBArray

See also

Returns the highest index of a given dimension of a VBArray object.

Syntax

```
safeArray.ubound(dimension)
```

safeArray represent a VBArray object.
dimension represents the dimensions of the VBArray object to be evaluated. Optional.

unescape – Method

See also

This functions returns an ASCII string. The **unescape** function is not a method associated to an object but part for the JavaScript language.
The string returned by this function is a series of characters in the ISO-Latin-1 character set.

Syntax

```
unescape("text")
```

text is a text string or a property of an existing object using one of the following formats:
%integer, where ***integer*** is a number between 0 and 255,
hex, where ***hex*** is a hexadecimal number between 0x0 and 0xFF.

Example

```
<FORM NAME="form">
<INPUT TYPE="text" NAME=enter MAXLENGTH=40 SIZE=40 VALUE="Enter some text
">
<INPUT TYPE="button" VALUE="conversion ASCII." onClick="conversion()"><BR>
Conversion in 'escape' mode:<BR>
<INPUT TYPE="text" NAME=convert MAXLENGTH=100 SIZE=100>
re-conversion in normal mode:<BR>
<INPUT TYPE="text" NAME=convert2 MAXLENGTH=40 SIZE=40>
</FORM>
<SCRIPT LANGUAGE="JavaScript">
function conversion()
{
document.form.convert.value=escape(document.form.input.value);
document.form.convert2.value=unescape(document.form.convert.value);
}
</SCRIPT>
```

unshift - Method of object: Array

See also

Adds argument elements to the beginning of an Array object.

Syntax

```
arrayName.unshift(element1, element2, …, elementn)
```

Parameters

arrayName represents the Array object.
element1 to **elementn** represents the elements to add to the beginning of the **arrayName** object.

untaint – Method

See also

The function removes "tainting" from a property. For more information on this function, refer to "data tainting"

Note

However, it is highly recommended that this function not be used as it not very reliable. This method was discontinued after JavaScript 1.2.

Syntax

```
untaint(propertyName)
```

propertyName is the property from which "*tainting*" is removed.

unwatch - Method of object: Object

See also

Removes a *watchpoint* set with the watch method.

Syntax:

```
unwatch(prop)
```

Parameters

`prop` represents the name of the object property.

UTC – Method of object: Date

See also

Returns the number of milliseconds elapsed since January 1, 1970 00:00:00 (GMT) and a Date object.

Syntax:

```
Date.UTC(year, month, day [, hrs] [, min] [, sec])
```

Parameters

year represents a year after 1900.
month represents a month between 0 and 11.
day represents a day between 1 and 31.
hrs represents hours between 0 and 23.
min represents minutes between 0 and 59.
sec represents seconds between 0 and 59.

Example

```
<SCRIPT LANGUAGE="JavaScript">
// Return the number of milliseconds elapsed since Jan 1 1970, GMT
// Date.UTC(year,mos,day[,hr][,min][,sec])
document.write("Date.UTC(25,9,69) = "+Date.UTC(25,9,69)+"<BR>");
Date.UTC(25,9,69)
</SCRIPT>
```

valueOf – Method of objects: Array, Boolean, Date, Function, Number, Object, RegExp and String

See also

Returns the primitive value of an object.

Syntax

```
Objet.valueOf()
```

watch – Method of object: Object

See also

Watches the evolution of an object's property and runs a function when the property's value changes.

Syntax:

```
watch(prop, handler)
```

Parameters

prop represents the name of the object property.
handler is the function to call.

write – Method of object: document

See also

This method writes one or more HTML expressions to a document in the specified window. This method is exactly the same as the writeln method, except it doesn't insert a carriage return at the end of the line.
You can use the **write** method with any type of **<SCRIPT LANGUAGE="JavaScript">** element or with an event.

Syntax

```
document.write(expression1 [, expression2], ..[, expressionN])
```

You can specify any JavaScript expression or a property of an existing object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
greeting= "Good morning";
one = 1;
two = 2;
document.write(greeting,
           ", here are some examples of how to use <I><B>write</I></B>
                 :<BR>",
           (parseInt("458")<parseInt("78")),
           "<BR>",
           eval("78 + 8"),
           "<BR>",
           "78+8",
           "<BR>",
           eval(one+two),
           "<BR>",
           (parseInt(two)+parseInt(one))
           );
</SCRIPT>
```

writeln – Method of object: document

See also

This method writes one or more HTML expressions to a document in the specified window. This method adds a carriage return at the end of the line.
You can use the **writeln** method with any type of **<SCRIPT LANGUAGE="JavaScript">** element or with an event.

Syntax

```
document.write(expression1 [, expression2], ..[, expressionN])
```

You can specify any JavaScript expression or a property of an existing object.

Example

```
<SCRIPT LANGUAGE="JavaScript">
greeting = "good morning";
one = 1;
two = 2;
document.writeln(greeting,
           ", here are some examples of how to use <I><B>writeln</I></B>
           :<BR>",
           ", writeln adds a carriage return:<BR>",
           (parseInt("458")<parseInt("78")),
           "<BR>",
           eval("78 + 8"),
           "<BR>",
           "78+8",
           "<BR>",
           eval(one+two),
           "<BR>",
           (parseInt(two)+parseInt(one))
           );
</SCRIPT>
```

DATA TAINTING

Browsers automatically prevent a server's scripts to have access to the properties of a document found on another server. This restriction is a precaution against gaining access by means of a script, to private information, such as directory structures or the last Web sites the user has visited. However, this restriction also prevents other legitimate and necessary applications.

The "*data tainting* function keeps the restriction with regards to security but by means of a special mechanism, it allows the execution of legitimate applications. When "*data tainting*" is applied, the script found in a window can read the properties of another window, regardless of the server where the document in the window is located. If the script running in the first window tries to send data from the properties of another window form another server, a confirmation box appears so that the user can accept or cancel the operation. By default, all the properties contained in the second window have applied the "*taint*" function so that their properties cannot be sent to a server except to the server containing the document in the window.

To add "*data tainting*" to a property, you must use the "*taint*" function. To remove "*data tainting*" from a property, you must use the "*untaint*" function.

Add - Method of object: Dictionary

See also

This method adds a pair of "key elements" to a **Dictionary object.**

Syntax

```
object.Add (key, item)
```

object	must be the name of a **Dictionary object.**
key	is associated to the item added.
item	is associated to the key added.

Note:

An error is returned if the ***key*** value already exists.

Example

```
var d;
d = new ActiveXObject("Scripting.Dictionary");
d.Add("a", "Argentine");
d.Add("b", "Belgium");
d.Add("c", "Canada");
```

Add - Method of the Folders Collection

See also

The **Add** method adds a new **Folder** to a collection of **Folders**.

Syntax

```
object.Add (folderName)
```

object	represents the name of a collection of **Folders**.
folderName	New **Folder** name to add.

Example:

```
function AddNewFolder(path,folderName)
{
  var fso, f, fc, nf;
  fso = new ActiveXObject("Scripting.FileSystemObject");
  f = fso.GetFolder(path);
  fc = f.SubFolders;
  if (folderName != "" )
    nf = fc.Add(folderName);
  else
    nf = fc.Add("Name of new folder");
}
```

BuildPath - Method of object: FileSystemObject

See also

This method adds a name to an existing access path.

Syntax

```
object.BuildPath(path, name)
```

object	refers to a **FileSystemObject** object.
path	existing access path to which the **name** element is added. The path can be either an absolute or relative path. It isn't necessary to indicate an existing folder.
name	name added to an existing **path** element.

Note:

If necessary, the **BuildPath** method inserts an additional separator between the existing path and name.

Close - Method of object: TextStream

See also

This method is used to close an open **TextStream** file.

Syntax

```
object.Close( );
```

object refers to the name of a **TextStream** object.

Copy - Method of objects: File and Folder

See also

This method is used to copy a file or folder from one place to another.

Syntax

```
object.Copy( destination[, overwrite] );
```

object	refers to the name of a **File** or **Folder** object.
destination	place where the file or folder should be copied to. Do not use wildcards.
overwrite	optional. Boolean value which returns **True** if the existing files or folders should be replaced (default value) ; **False** if they are not to be replaced.

Note:

The results of the **Copy** method (for objects **File** and **Folder**) are the same as those obtained using **FileSystemObject.CopyFile** or **FileSystemObject.CopyFolder**. Moreover, these alternate methods allow copying several files or folders.

CopyFile - Method of object: FileSystemObject

See also

This method is used to copy one or several files from one location to another.

Syntax

```
object.CopyFile ( source, destination[, overwrite] )
```

object	**object** refers to the name of a **FileSystemObject** object.
source	text string that can handle wildcards to specify one or several files that are to be copied.
destination	text string indicating the location where the source file or files are to be copied. Do not use wildcards.
overwrite	optional. Boolean value which returns **True** if the existing files or folders should be replaced (default value) ; **False** if they are not to be replaced. Note: **CopyFile** is cancelled if the destination is read-only.

Note:

You can use wildcards if these are part of the last part of the **source** access argument, i.e. *c:**mydocuments*\ *letters**.doc, but not c:\\mydocuments*\\mary.doc

CopyFolder - Method of object: FileSystemObject

See also

This method is used to copy a file or folder from one place to another.

Syntax

```
object.CopyFolder ( source, destination[, overwrite] )
```

object	**object** refers to the name of a **FileSystemObject** object.
source	text string that can handle wildcards to specify one or several folders that are to be copied.
destination	text string indicating the location where the source folders and sub-folders are to be copied. Do not use wildcards.
overwrite	optional. Boolean value which returns **True** if the existing folder or folders should be replaced (default value) ; **False** if they are not to be replaced.

Note:

You can use wildcards if these are part of the last part of the **source** access argument, i.e. *c:**mydocuments*\ *letters**.doc, but not c:\\mydocuments**

CreateFolder - Method of object: FileSystemObject

See also

This method is used to create a new folder.

Syntax

```
object.CreateFolder(foldername)
```

object	refers to the name of a **FileSystemObject** object.
foldername	text string that identifies the name of the folder to be created.

Note:

Returns an error if the folder already exists.

CreateTextFile - Method of objects: FileSystemObject and Folder

See also

This method is used to create a name for a specified file and to return a **TextStream** object that can be used to read and write to the file.

Syntax

```
object.CreateTextFile(filename[, overwrite[, unicode]])
```

object	refers to the name of a **FileSystemObject** or **Folder** object.
filename	text string that identifies the name of the file to be created.
overwrite	optional. Boolean value that indicates if an existing file can be erased. The value is **true** if the file can be erased, and **false** if not. If omitted, the existing files are not erased.
unicode	optional. Boolean value indicating if the file created uses the Unicode (**true**) or ASCII (**false**) format. If omitted, an ASCII file is created.

Delete - Method of objects: File and Folder

See also

This method is used to delete a file or folder.

Syntax

```
object.Delete( force );
```

object	refers to a **File** or **Folder** object.
force	optional. Boolean value that returns **True** when the files or folders with the *read-only* attributed are to be deleted and **False** (default) if they are not be deleted.

Notes:

An error is returned if the specified file or folder does not exist.

The results obtained with the Delete method on a **File** or **Folder** object are the same as those obtained with the **FileSystemObject.DeleteFile** or **FileSystemObject.DeleteFolder** methods.

The Delete method deletes a folder without verifying whether its contents are empty or not.

DeleteFile - Method of object: FileSystemObject

See also

This method is used to delete a specified file.

Syntax

```
object.DeleteFile ( filespec[, force] );
```

object	refers to the name of a **FileSystemObject** object.
filespec	name of file to be deleted. **filespec** can contain wildcards in the last part of the access path.
force	optional. Boolean value: **true** to delete the read-only files and **false** (default value) if they are not to be deleted.

Note:

DeleteFile returns an error if any of the files specified is not found. This method is cancelled when it encounters the first error. The deletions that occurred before the error was found are not cancelled.

DeleteFolder - Method of object: FileSystemObject

See also

This method is used to delete a folder and its contents.

Syntax

```
object.DeleteFolder ( folderspec[, force] );
```

object	refers to the name of a **FileSystemObject** object.
folderspec	name of folder to be deleted. **folderspec** can contain wildcards in the last part of the access path.
force	optional. Boolean value: **true** to delete the read-only folders and **false** (default value) if they are not to be deleted.

Note:

The **DeleteFolder** method deletes the folder whether it is empty or not.

DeleteFolder returns an error if any of the folders specified is not found. This method is cancelled when it encounters the first error. The deletions that occurred before the error was found are not cancelled.

DriveExists - Method of object: FileSystemObject

See also

This method returns the **True** value if the specified drive exists and **False** if it doesn't exist.

Syntax

```
object.DriveExists(drivespec)
```

object refers to the name of a **FileSystemObject** object.
drivespec is the drive letter or the full access path.

Note:
In case of removable drives, the **DriveExists** returns **true**, even if the drive is not connected. The **IsReady** property of the **Drive** object determines if the drive is ready.

Exists - Method of object: Dictionary

See also

This method returns the **true** value if the specified element in the **Dictionary** object exists, and **false** if it doesn't.

Syntax

```
object.Exists(key)
```

object	refers to the name of a **Dictionary** object.
key	refers to the ***key*** value that is being searched in the **Dictionary** object.

FileExists - Method of object: FileSystemObject

See also

This method returns the **True** value if the specified file exists and **False** if it doesn't exist.

Syntax

```
object.FileExists(filespec)
```

object	always refers to the name of a **FileSystemObject** object.
filespec	file name to be verified (does it exist?). An absolute or relative access path must be specified if the file does not exist in the current folder.

FolderExists - Method of object: FileSystemObject

See also

This method returns the **True** value if the specified folder exists and **False** if it doesn't exist.

Syntax

```
object.FolderExists(folderspec)
```

object	always refers to the name of a **FileSystemObject** object.
filespec	folder name to be verified (does it exist?). An absolute or relative access path must be specified if the folder does not exist in the current folder.

GetAbsolutePathName - Method of object: FileSystemObject

See also

This method returns an unambiguous and complete access path according to the path supplied in the argument.

Syntax

```
object.GetAbsolutePathName(pathspec)
```

object	refers to the name of a **FileSystemObject** object.
pathspec	to specify an access path that is to be converted into an unambiguous and complete access path.

Note:

An access path is complete and unambiguous when it shows a complete reference in the root of the specified drive.

GetBaseName - Method of object: FileSystemObject

See also

This method returns a string containing the base name of the last part of the access path (without the file's extension).

Syntax

```
object.GetBaseName(path)
```

object	refers to the name of a **FileSystemObject** object.
path	specifies the access path component for which the base name has to be returned.

Note:

The **GetBaseName** method returns a string of zero length ("") if no section corresponds to the **path** argument.

GetDrive - Method of object: FileSystemObject

See also

This method returns the **Drive** object corresponding to the drive specified in the access path.

Syntax

```
object.GetDrive ( drivespec );
```

object	refers to the name of a **FileSystemObject** object.
drivespec	The **drivespec** arguments can be used in the following manner: (c, c:, c:\ ), and any specification related to a network (\\machine2\shared1).

Note:

In case of a "shared networks", the system is checked to make sure the shared network exists.

The method returns an error if the **drivespec** element does not conform to one of the accepted forms or does not exist.

GetDriveName - Method of object: FileSystemObject

See also

This method returns a string containing the name of the specified drive in the access path.

Syntax

```
object.GetDriveName(path)
```

object	refers to the name of a **FileSystemObject** object.
path	information on the access path including the name of the drive.

Note:
GetDriveName returns a string of length zero ("") when the drive is not detected.

GetExtensionName - Method of object: FileSystemObject

See also

GetExtensionName returns a string containing the extension name of the last part of an access path.

Syntax

```
object.GetExtensionName(path)
```

object	refers to the name of a **FileSystemObject** object.
path	specifies the path including the extension that must be returned.

Note:

The root directory (\) is considered a component of the network drives.

GetExtensionName returns a string of length zero ("") if none of the components corresponds to the **path** argument.

GetFile - Method of object: FileSystemObject

See also

The **GetFile** method returns a **File** object corresponding to a file name present in the access path.

Syntax

```
object.GetFile(filespec)
```

object	refers to the name of a **FileSystemObject** object.
filespec	**filespec** is the absolute or relative access path of a specified file.

GetFileName - Method of object: FileSystemObject

See also

The **GetFileName** returns the last component of the path indicated that is not part of the drive specifications.

Syntax

```
object.GetFileName(pathspec)
```

object	refers to the name of a **FileSystemObject** object.
pathspec	Absolute or relative access path of a specified file.

Note:

A string of length zero ("") is returned if **pathspec** does not end with the specified component.

GetFileVersion - Method of object: FileSystemObject

See also

This method returns the version number of the specified file.

Syntax

```
object.GetFileVersion(pathspec)
```

object	refers to the name of a **FileSystemObject** object.
pathspec	specifies the absolute or relative access path of a specified file.

Note:

The **GetFileVersion** method returns a string of length zero ("") when **pathspec** does not end with the file name or when the file doe snot contain information on its version.

GetFolder - Method of Folder and FileSystemObject

See also

This method returns a **Folder** object corresponding to a folder specified in the access path.

Syntax

```
object.GetFolder(folderspec)
```

object	refers to the name of a **FileSystemObject** object.
pathspec	Absolute or relative access path of a specified folder.

GetParentFolderName - Method of object: FileSystemObject

See also

Returns a string containing the name of the parent folder of the last component in a specified access path.

Syntax

```
object.GetParentFolderName(path)
```

object	refers to the name of a **FileSystemObject** object.
path	specifies the path of the component for which the name of the parent folder must be returned..

Note:

GetParentFolderName returns a string of length zero ("") if no parent folder exists for the component specified in the **path** argument.

GetSpecialFolder - Method of object: FileSystemObject

See also

This method returns the specified *special folder* object.

Syntax

```
object.GetSpecialFolder(folderspec)
```

object refers to the name of a **FileSystemObject** object.

folderspec name of special folder that has to be returned. This element can be any of the following constants:

WindowsFolder	0	The **Windows folder** (files installed by Windows OS).
SystemFolder	1	The **System folder** (libraries, fonts, and drivers for peripherals).
TemporaryFolder	2	The **Temp folder** (temporary files). Their access path is found in the environment variable TMP.

GetTempName - Method of object: FileSystemObject

See also

GetTempName returns the name of a temporary folder or file (random name). Useful for operations requiring the type of folders or files.

Syntax

```
object.GetTempName ( );
```

object is an optional element that always refers to a **FileSystemObject** object.

Note:

To create a temporary folder or file, you must use the **CreateTextFile** method. The **GetTempName** method is used only to supply a name for said folder or file.

Items - Method of object: Dictionary

See also

This method creates a table containing all the elements of a **Dictionary** object.

Syntax

```
object.Items( )
```

object refers to the name of a **Dictionary** object.

Keys - Method of object: Dictionary

See also

The **Keys** method returns a table containing all the existing keys in a **Dictionary** object.

Syntax

```
object.Keys( )
```

object refers to the name of a **Dictionary** object.

Move - Method of objects: File and Folder

See also

This method is used to move a file or folder from one place to another.

Syntax

```
object.Move( destination );
```

object	refers to the name of a **File** or **Folder** object.
destination	Location to which the file or folder is to be moved. Do not use wildcards.

Note:

The results of the **Move** method (on objects **File** and **Folder**) are the same as those obtained with the **FileSystemObject.MoveFile** or **FileSystemObject.MoveFolder** methods. Moreover, these alternate methods allow moving several files or folders.

MoveFile - Method of object: FileSystemObject

See also

This method is used to move one or several files from one location to another.

Syntax

```
object.MoveFile ( source, destination );
```

object	**object** refers to the name of a **FileSystemObject** object.
source	text string that can contain wildcards and specify the access path for the files to be moved.
destination	text string indicating the location where the source file or files are to be copied. Do not use wildcards.

Note:

This method allows moving files between volumes only if the operating system allows it.

MoveFolder - Method of object: FileSystemObject

See also

This method is used to move one or several folders from one location to another.

Syntax

```
object.MoveFolder ( source, destination );
```

object	**object** refers to the name of a **FileSystemObject** object.
source	specifies the access path of the folders that are to be moved. The string for this argument can include wildcards only for the last component of the access path.
destination	text string indicating the location where the source folders and sub-folders are to be moved. Do not use wildcards.

Note:

This method allows moving folders between volumes only if the operating system allows it.

OpenAsTextStream - Method of object: File

See also

This method is used to open a specified file and return a **TextStream** object that can be used to read, write and add data to the file.

Syntax

```
object.OpenAsTextStream([iomode, [format]])
```

object refers to the name of a **File** object.

iomode optional. Indicates input/output mode: *ForReading*, *ForWriting* or *ForAppending*.

ForReading	1	Open a file as read-only.
ForWriting	2	Open a file in write mode. If a file with the same name exists, its contents are erased.
ForAppending	8	Open a file and append data at the end.

format optional. One of three **Tristate** values indicate the opened file format. If not indicated, the file is open in the default ASCII format.

TristateUseDefault	-2	Opens the file and uses the system's default parameters.
TristateTrue	-1	Opens the file in Unicode format.
TristateFalse	0	Opens the file in ASCII format.

Note:

The **OpenAsTextStream** method is the same as using the **OpenTextFile** **method** of the **FileSystemObject** object. The **OpenAsTextStream** method can also be used to write to a file.

OpenTextFile - Method of object: FileSystemObject

See also

This method opens the specified file and returns a **TextStream** object. This object can be used to read, write and append data to a file.

Syntax

```
object.OpenTextFile(filename[, iomode[, create[, format]]])
```

object	**Object** always refers to the name of a **FileSystemObject** object.
filename	string specifying the file to be opened.
iomode	optional. Could be one of the three following constants: *ForReading*, *ForWriting* or *ForAppending*.

ForReading	1	Open a file as read-only.
ForWriting	2	Open a file in write mode. If a file with the same name exists, its contents are erased.
ForAppending	8	Open a file and append data at the end.

create	optional. Boolean value: Specifies if a new file must be crated when the specified **filename** element does not exist. **True**, a new file is created, or **False** no files are created. If not specified, no new files are created.
format	optional. It can take one of three **Tristate** values to indicate the type of format for the opened file. If it is not indicated, the file will be opened in ASCII format.

TristateUseDefault	-2	Opens the file and uses the system's default parameters.
TristateTrue	-1	Opens the file in Unicode format.
TristateFalse	0	Opens the file in ASCII format.

Read - Method of object: TextStream

See also

The **Read** method is used to indicate a number of characters to read in a **TextStream** file and return the text string obtained.

Syntax

```
object.Read(characters)
```

object	**Object** is always the name of a **TextStream** object.
characters	number of characters to be read in a file.

ReadAll - Method of object: TextStream

See also

The **ReadAll** method reads an entire **TextStream** file and returns the text string obtained.

Syntax

```
object.ReadAll( );
```

object refers to the name of a **TextStream** object.

Note:

When dealing with large files, **ReadAll** uses the memory resources in an unreliable manner. Therefore it is best to use a different technique, such as *one line at a time* to read a large file.

ReadLine - Method of object: TextStream

See also

The method read a whole line (up to the character of the following line but without including it) from a **TextStream** file and returns the string obtained.

Syntax

```
object.ReadLine( )
```

object refers to the name of a **TextStream** object.

Remove - Method of object: Dictionary

See also

The **Remove** method removes a pair of "key elements" of a **Dictionary** object.

Syntax

```
object.Remove(key)
```

object	**Object** refers to the name of a **Dictionary** object.
key	indicates the (*key*), associated to the pair of "key elements" that must be removed from the **Dictionary** object.

Note:

An error is returned if the pair of "key elements" does not exist.

RemoveAll - Method of object: Dictionary

See also

The **RemoveAll** method removes all the "key element" pairs from a **Dictionary** object.

Syntax

```
object.RemoveAll( )
```

object refers to the name of a **Dictionary** object.

Skip - Method of object: TextStream

See also

Ignore a specified number of characters while reading a TextStream file.

Syntax

```
object.Skip(characters)
```

object	**Object** refers to the name of a **TextStream** object.
characters	specifies the number of characters that have to be ignored while reading a file.

Note:
Ignored characters are abandoned.

SkipLine - Method of object: TextStream

See also

This method allows skipping the following line while reading a **TextStream** file.

Syntax

```
object.SkipLine( )
```

object refers to the name of a **TextStream** object.

Write - Method of object: TextStream

See also

This method is used to write a specified string of characters to a **TextStream** file.

Syntax

```
object.Write(string)
```

object always refers to the name of a **TextStream** object.
string indicates the text to be written to the file.

Note:

The specified text string are written without spaces or separation characters between each string. To insert a "new line" character or a string ending with a "new line" character, you must use the **WriteLine** method.

WriteBlankLines - Method of object: TextStream

See also

Write a specified number of new line characters in a TextStream file.

Syntax

```
object.WriteBlankLines(lines)
```

object	**Object** always refers to a name of a **TextStream** object.
lines	specifies the number of new line characters to be inserted in the file.

WriteLine - Method of object: TextStream

See also

This method is used to write a string text and a new line character in a **TextStream** file.

Syntax

```
object.WriteLine([string])
```

object	always refers to the name of a **TextStream** object.
string	optional. Indicates the text to be written to the file. If no text is indicated, a new line character will be inserted in the file.

onAbort – Event of the Image Object

See also

This event is activated when the user cancels loading an image. This event could take place when the user clicks on a hyperlink or presses the browser's **Stop** button while an image is being loaded.

Syntax

`onKeyDown="JavaScript code` or `call a Javascript function"`

Example

```
<IMG SRC="unURL" onAbort="functionName">
```

onafterprint – Event

The **onafterprint** acts on the source object after the documents associated with it have been printed.

Syntax

```
onKeyDown="JavaScript code or call a Javascript function"
```

onafterupdate – Event

The **onafterupdate** event takes place immediately after the object containing the data has carried out an update of the data on the source object.

Syntax

```
onafterupdate ="JavaScript code or call a Javascript function"
```

onbeforecopy – Event

The **onbeforecopy** event acts on the source object before the selection is copied to the clipboard.

Syntax

```
onbeforecopy ="JavaScript code or call a Javascript function"
```

onbeforecut – Event

The **onbeforecut** event acts on the source object before the selection is cut from the document.

Syntax

```
onbeforecut ="JavaScript code or call a Javascript function"
```

onbeforeeditfocus – Event

The **onbeforeeditfocus** is activated before a control makes a change at the user interface level (UI, User Interface).

Syntax

```
onbeforeeditfocus ="JavaScript code or  call on a Javascript function"
```

onbeforepaste – Event

The **onbeforepaste** event acts on the target object before the selection is pasted from the clipboard to the document.

Syntax

```
onbeforepaste ="JavaScript code or call a Javascript function"
```

onbeforeprint – Event

The **onbeforeprint** event acts on the object before the document associated with it are printed.

Syntax

```
onbeforeprint ="JavaScript code or call a Javascript function"
```

onbeforeunload – Event

The **onbeforeunload** event is activated before a page is unloaded from a frame.

Syntax

```
onbeforeunload ="JavaScript code or call  a Javascript function"
```

onbeforeupdate – Event

The **onbeforeupdate** is activated before an object containing the data updates the data of said source object.

Syntax

```
onbeforeupdate ="JavaScript code or call  a Javascript function"
```

onBlur – Event of objects: Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, Textarea and window

See also

This event takes place when the user exits a field of the HTML form, a frame or a (window) by pressing the **TAB** key on the keyboard or using the mouse. The function called may be used to validate the information entered by the user.

Syntax

```
onBlur="JavaScript code or call a JavaScript function"
```

Example

```
<INPUT TYPE="text" VALUE="aValue" NAME="aName" onBlur="functionName">
```

onbounce – Event

The **onbounce** event is activated when the behavior attribute of the <MARQUEE> element is "`alternate`" and the element's content (text) reaches one of the edges of the scrolling area.

Syntax

```
onbounce ="JavaScript code or call a Javascript function"
```

oncellchange – Event

The **oncellchange** event is activated when the data is modified.

Syntax

```
oncellchange ="JavaScript code or call a Javascript function"
```

onChange – Event of objects: FileUpload, Select, Text and Textarea

See also

This event takes place when the user modifies de value of a list box, a drop down box, a text box or a textarea box. The function called may be used to validate the information entered in the fields of a form.

Syntax

```
onChange="JavaScript code or  call on a JavaScript function"
```

Example

```
<INPUT TYPE="text" VALUE="aValue" NAME="aName" onChange="functionName">
```

onClick – Event of objects: Button, Checkbox, document, Link, Radio, Reset and Submit

See also

This event takes place when a user presses and then releases the mouse button over a button, an option in a form, a hyperlink or a document.

Syntax

```
onClick="JavaScript code or call a Javascript function"
```

Example

```
<INPUT TYPE="button" VALUE="aValue" onClick="functionName">
```

oncontextmenu – Event

The **oncontextmenu** event is activated when the user presses the mouse right button over an element, which also activates a context menu.

Syntax

```
oncontextmenu ="JavaScript code or call on a Javascript function"
```

oncopy – Event

The **oncopy** event is activated when the user copies an object or a selection is copied to the clipboard.

Syntax

```
oncopy ="JavaScript code or call a Javascript function"
```

oncut – Event

The **oncut** event is activated when an object or a selection is removed from the document and placed on the operating system's clipboard.

Syntax

```
oncut ="JavaScript code or call a Javascript function"
```

ondataavailable – Event

The **ondataavailable** event is activated every time that the information sent by the source object is displayed by the system.

Syntax

```
ondataavailable ="JavaScript code or call a Javascript function"
```

ondatasetchange – Event

The **ondatasetchange** event is activated when the information exposed to a source object is modified.

Syntax

```
ondatasetchange ="JavaScript code or call a Javascript function"
```

ondatasetcomplete – Event

The **ondatasetcomplete** event is activated to indicate that all the information of the source object is available.

Syntax

```
ondatasetcomplete ="JavaScript code or call a Javascript function"
```

onDblClick – Event of objects: Area, document and Link

See also

This event takes place when the mouse button is pressed twice rapidly and then released over an element on a form or a hyperlink.

Syntax

```
onDblClick="JavaScript code or call a Javascript function"
```

ondrag – Event

The **ondrag** event is activated when a "drag and drop" is carried out using the mouse over an element.

Syntax

```
ondrag ="JavaScript code or call a Javascript function"
```

onDragDrop – Event of the object window

See also

This event takes place when the user drops and object (for example, a file) in the browser's window using the mouse.

Syntax

```
onDragDrop="JavaScript code or  call on a Javascript function"
```

ondragend – Event

The **ondragend** event is activated at the end of a "drag and drop" operation.

Syntax

```
ondragend ="JavaScript code or call a Javascript function"
```

ondragenter – Event

The **ondragenter** is activated when the user drags the element over a valid target ("drag and drop" operation carried out with the mouse).

Syntax

```
ondragenter ="JavaScript code or call a Javascript function"
```

ondragleave – Event

The **ondragleave** event is activated when the user drags an element outside a valid target (a "drag and drop" operation carried out with the mouse).

Syntax

```
ondragleave ="JavaScript code or call a Javascript function"
```

ondragover – Event

The **ondragover** is activated when the user drags an element and places it over a valid target (a "drag and drop" operation carried out with the mouse).

Syntax

```
ondragover ="JavaScript code or call a Javascript function"
```

ondragstart – Event

The **ondragstart** event is activated when the user starts a "drag and drop" operation with the element (a "drag and drop" operation carried out with the mouse).

Syntax

```
ondragstart ="JavaScript code or call a Javascript function"
```

ondrop – Event

The **ondrop** acts on the target object when the user releases the mouse button after a "drag and drop" operation.

Syntax

```
ondrop ="JavaScript code or call a Javascript function"
```

onError – Event of objects: Image and window

See also

This event takes place when an error occurs while loading an image in a document.

Syntax

```
onError="JavaScript code or call a Javascript function"
```

Example

```
<IMG SRC="unURL" onError="functionName">
```

onerrorupdate – Event

The **onerrorupdate** event is activated if an error occurs while updating information related to the source object.

Syntax

```
onerrorupdate ="JavaScript code or call a Javascript function"
```

onfilterchange – Event

The **onfilterchange** event is activated while a visual filter changes state or completes a transition.

Syntax

```
onfilterchange ="JavaScript code or call a Javascript function"
```

onfinish – Event

The **onfinish** is activated when a MARQUEE animation element is completed.

Syntax

```
onfinish ="JavaScript code or call a Javascript function"
```

onFocus – Event of objects: Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, Textarea and window

See also

This event takes place when an element (a form field, frame, layer, etc.) or an object (window) gets the focus. Focus can be achieved using the mouse, the **TAB** key on the keyboard or by a focus method.

Syntax

```
onFocus="JavaScript code or call a Javascript function"
```

Example

```
<INPUT TYPE="text" VALUE=«aValue» NAME=«aName» onFocus="functionName">
```

onhelp – Event

The **onhelp** event is activated when the user presses the F1 key on the keyboard and the browser's window is active.

Syntax

```
onhelp ="JavaScript code or call a Javascript function"
```

onKeyDown – Event of objects: document, Image, Link and Textarea

See also

This event takes place when the user presses a key on the keyboard.

Syntax

```
onKeyDown="JavaScript code or  call on a JavaScript function"
```

onKeyPress – Event of objects: document, Image, Link and Textarea

See also

This event takes place when the user releases the key that he was previously pressing.

Syntax

```
onKeyPress="JavaScript code or call a Javascript function"
```

onKeyUp – Event of objects: document, Image, Link and Textarea

See also

As a rule, the **onKeyUp** event takes place when the user releases a key on the keyboard. However, several precautions have to be taken into consideration when using this event, giving the possible confusion that could arise with the onKeyPress event.

Syntax

```
onKeyUp="JavaScript code or call a Javascript function"
```

onLoad – Event of objects: Image, Layer and window

See also

This event takes place when the browser completes loading an HTML page, all the frames of a **<FRAMESET>** element or an image.

Syntax

```
onLoad="JavaScript code or call a Javascript function"
```

Example

```
<BODY onLoad="functionName">
```

onlosecapture – Event

The **onlosecapture** event is activated when an element loses mouse focus.

Syntax

```
onlosecapture ="JavaScript code or call a Javascript function"
```

onMouseDown – Event of objects: Button, document and Link

See also

This event takes place when the user presses one of the two mouse buttons.

Syntax

```
onMouseDown="JavaScript code or call a Javascript function"
```

onMouseMove – Event

See also

This event takes place when the mouse pointer is moved and placed over an object.

Syntax

```
onMouseMove="JavaScript code or call a Javascript function"
```

onMouseOut – Event of objects: Area, Layer and Link

See also

This event takes place each time the user moves the mouse pointer out of an area containing a hyperlink (image area, hyperlink text).

Syntax

```
onMouseOut="JavaScript code or call a Javascript function"
```

Example

```
<AREA SHAPE=".." COORDS=".." HREF=".." onMouseOut="functionName()">
```

onMouseOver – Event of objects: Area, Layer and Link

See also

This event takes place each time that the mouse pointer is moved over an area that contains a hyperlink (hyperlink text or image area).

Syntax

```
onMouseOver="JavaScript code or call a Javascript function"
```

Example

```
A HREF="aURL" onMouseOver=functionName()">   </A>
     AREA SHAPE=".." COORDS=".." HREF=".." onMouseOver="functionName()">
```

onMouseUp – Event of objects: Button, document and Link

See also

This event takes place when the user releases the mouse button.

Syntax

```
onMouseUp="JavaScript code or call a Javascript function"
```

onMove – Event of the object window

See also

This event takes place when the user or a script moves a window or frame.

Syntax

```
onMove="JavaScript code or call a Javascript function"
```

onpaste – Event

The **onpaste** event is activated when the user pastes the contents of the clipboard on the document.

Syntax

```
onpaste ="JavaScript code or call a Javascript function"
```

onpropertychange – Event

The **onpropertychange** event is activated when the element's property changes.

Syntax

```
onpropertychange ="JavaScript code or call a Javascript function"
```

onreadystatechange – Event

The **onreadystatechange** event is activated when the element status changes.

Syntax

```
onreadystatechange ="JavaScript code or call a Javascript function"
```

onReset – Event of the object Form

See also

This event takes place when the user activates a form's **reset** button.

Syntax

```
onReset="JavaScript code or call a Javascript function"
```

Example

```
<FORM ACTION="URL" METHOD=POSTorGET onReset="functionName()">
```

onResize – Event of the object window

See also

This event takes place when the user modifies the browser's window size.

Syntax

```
onResize="JavaScript code or call a Javascript function"
```

onrowenter – Event

The **onrowenter** event is activated when the current row has changed from the data source and new data is available.

Syntax

```
onrowenter ="JavaScript code or call a Javascript function"
```

onrowexit – Event

The **onrowexit** event is activated just before the control of the data source modifies the object's current row.

Syntax

```
onrowexit ="JavaScript code or call a Javascript function"
```

onrowsdelete – Event

The **onrowsdelete** event is activated just before deleting rows from a set of registries.

Syntax

```
onrowsdelete ="JavaScript code or call a Javascript function"
```

onrowsinserted – Event

The **onrowsinserted** event is activated after new rows are inserted in a set of registries.

Syntax

```
onrowsinserted ="JavaScript code or call a Javascript function"
```

onscroll – Event

The **onscroll** event is activated when the user moves the cursor over an object's scroll bar.

Syntax

```
onscroll ="JavaScript code or call a Javascript function"
```

onSelect – Event of objects: Text and Textarea

See also

This event takes place when the user select text in a text box or in a multiline text box.

Syntax

```
onSelect="JavaScript code or call a Javascript function"
```

Example

```
<INPUT TYPE="text" VALUE=«aValue» NAME=«aName» onSelect="functionName">
```

onselectstart – Event

The **onselectstart** event is activated when the object is selected.

Syntax

```
onselectstart ="JavaScript code or call a Javascript function"
```

onstart – Event

The **onstart** event is activated every time the MARQUEE animation element is reactivated.

Syntax

```
onstart ="JavaScript code or call a Javascript function"
```

onstop – Event

The **onstop** event is activated when the user clicks on the "Stop" button on the browser.

Syntax

```
onstop ="JavaScript code or call a Javascript function"
```

onSubmit – Event of the object Form

See also

This event takes place when the user sends the information on a form. For example, the function called might verify the validity of the data placed on the form and prevent forwarding the form if certain information is found to be mistaken.

Syntax

```
onSubmit="JavaScript code or call a Javascript function"
```

Example

```
<FORM ACTION="URL" METHOD=POSTorGET onSubmit="functionName()">
```

onUnload – Event of the object window

See also

This event takes place when the current document is removed from the window to make way for another document or when the browser is closed by the user.

Syntax

```
onUnload="JavaScript code or call a Javascript function"
```

Example

```
<BODY onUnload="functionName">
```

@cc_on – Statement

See also

Activates conditional compilation in the scripting engine.

Syntax

```
@cc_on
```

Note

In order to allow browsers that do not perform conditional compilation to run the rest of the script, you should place the **@cc_on** as a comment:

```
...
/*@cc_on*/
...
```

@if – Statement

See also

This conditional statement executes a group of statements as a function of the value of an expression.

Syntax

```
@if (condition1)
     text1
[@elif (condition2)
     text2]
[@else
     text3]
@end
```

Parameters

`condition1, condition2`	Expressions that can be converted to Boolean values.
`text1`	Text returned if condition1 is **true**.
`text2`	Text returned if condition1 is **false** and condition2 is **true**.
`text3`	Text returned if condition1 and condition2 are **false**.

Note

You can use several **@elif** statements. In that case, they all have to preceded by an **@else** statement.

@set – Statement

See also

This statement creates variables used by the statements of the conditional compilation.

Syntax

```
@set @varname = term
```

Parameters

varname	Valid JScript variable name. Must be preceded by an "@" character at all times.
term	Zero or more unary operators followed by a constant, conditional compilation variable, or parenthesized expression.

Note

Please note that conditional compilation does not accept strings, but only numeric variables and boolean values. Although variables created width **@set** are usually used in conditional compilation statements, they can be used anywhere in JScript code.

Examples of variable statements

```
@set @mavar1 = 10

@set @mavar2 = (@mavar1 * 2)

@set @mavar3 = @_ jscript_build
```

break – Statement

See also

Terminates the current loop, or if in conjunction with a *label*, terminates the associated statement

Terminates the current for or while loop, ignoring the conditional test of the loop. The **break** statement can also be used to terminate switch or label statements.

Syntax

```
break
```

Example

```
for (i=0; i < 10; i++) {
  if (i == 3)   // Stop at 3
    break;
}
```

catch – Statement

See also

Implements error handling for JScript.

Syntax

```
try
  tryStatement
catch(exception)
  catchStatement
```

Parameters

tryStatement	Statement where an error can occur. Can be a compound statement.
exception	Any variable name. The initial value of exception is the value of the thrown error.
catchStatement	Statement to handle errors occurring in the associated **tryStatement**. It can be a compound statement.

comment – Statement

This statement allows the author of a script to insert a comment that will be ignored by the browser. The **comment** statement is very useful for including information that will be useful when the script is reviewed.

There are two ways for including comments in JavaScript code:

Place a comment in a line

```
//
```

Example

```
// This is a comment
```

Place a comment on one or several lines

```
/*  */
```

Example

```
/* This is a comment consisting of
    two lines */
```

continue – Statement

See also

C Stops the current iteration of a loop, and starts a new iteration
This statement stops the iteration of the for or while loops, and ignores the rest of the code in the loop.

Syntax

```
continue
```

Example

```
for (i=0; i < 10; i++) {
  if (i == 3)
    continue;
  r = r + i;
 // in this loop, the number 3 will never be added
}
```

do...while – Statement

See also

Executes the statements until the condition evaluates to false.

Syntax

```
do
     statements
while (condition);
```

Example

```
do {
     i+=1
     document.write(i);
while (i<5);
```

export – Statement

See also

Recognised only by Netscape Navigator, this statement allows a signed script to supply functions, properties and objects to other scripts, signed or not.

Syntax

```
export name1, name2, ..., nameN
export *
```

for – Statement

See also

This statement executes a number of statements for as long as the condition is true. It is used as follows:

Syntax

```
for(initialisation; test; increment)
```

Example

```
for (i=10; i > 0 ; i--)
    r += i;
```

for...in – Statement

See also

This statement allows displaying all the properties of an object.

Syntax

```
for (variable in object) {
   statements
}
```

function – Statement

See also

This statement is used to identify the name of a function and its parameters if necessary.

Syntax

```
function functionName([parameter] [, parameter] [..., parameter]) {
  statements }
```

Example

```
function calculate(Number1, Number2) {
    // Statements...
}
```

if...else – Statement

See also

Executes a group of statements if the specified condition is true. When the condition is false, another group of statements could be executed.

Syntax

```
if (condition) {
   statements1
}
[else {
   statements2
}]
```

import – Statement

See also

Much like the export statement, **`import`** is used to import properties, functions and objects from a signed script that has exported the information.

Syntax

```
import ObjectName.name1, ObjectName.name2, ..., ObjectName.nameN import ObjectName.*
```

label – Statement

See also

Provides a statement with an identifier that lets you refer to it elsewhere in your script.

Syntax

```
label :
     statements
```

Labeled – Statement

See also

This statement provides an identifier for a statement.

Syntax

```
label :
    statement
```

Parameters

`label`	identifier name used to make reference to the targeted statement.
`statement`	Statement associated to the (`label`) identifier. Can be a compound statement.

return – Statement

See also

This statement specifies the value that has to be returned by a function.

Syntax

```
return expression
```

Example

```
function GetPop(country) {
   return country.pop;
}
```

switch – Statement

See also

Enables the execution of one or more statements when a specified expression's value matches a label

Allows a program to evaluate the contents of a variable (expression) and to compare possible constants. when a match is found, a group of statements is executed.

Syntax

```
switch (expression){
   case "constant" :
       statements;
      break;
   case "constant" :
      statements;
      break;
     ...
   default : statements;
}
```

this – Statement

See also

This statement is used to refer to a current object.

Syntax

```
this.property
```

Note

For client versions of JavaScript, **this** makes reference to a window object it is used out of context of a specific object.

Example

```
function software(type, name, version)
{
  this.type = text;
  this.name = word;
  this.version = 97;
```

throw – Statement

See also

This condition generates an error condition that can be handled by try...catch statement.

Syntax

```
throw exception
```

exception can be any expression.

try...catch – Statement

See also

Implements error handling in JScript.

Syntax

```
try
  tryStatement
catch(exception)
  catchStatement
```

Parameters

tryStatement	Statement where an error can occur. Can be a compound statement.
exception	Any variable name. The initial value of exception is the value of the error.
catchStatement	Statement to handle errors occurring in the associated **tryStatement**. statement. Can be a compound statement

var – Statement

See also

This statement is used to declare a variable. A variable can be initiated after this statement.

Syntax

```
var variableName[= value] [..., variableName [= value] ]
```

Example

```
  var Day;
  var Month = "September";
  var Hour, Min, Sec;
```

while – Statement

See also

This statement Executes a statement until a specified condition is met.

Syntax

```
while(condition) {
    statements;
  }
```

Example

```
 Total = 0;
  i = 1;
  while (i <= 10) {
    Total += Note[i];
    i++;
  }
```

with – Statement

See also

Establishes the default object for a statement

This statement establishes a default value for a group of statements. The following statements can then make references to properties and methods without having to mention the object. JavaScript assumes that the properties and methods of the default object defined for the **`with`** statement.

Syntax

```
with (anObject){
    statements
}
```

Example

```
 with (state) {
    name = "France";
    pop = 57.5;
 }
  // instead of
  state.name = "France";
  state.pop = 57.5;
```

?: – Special Operator

See also

Trinary condition. Executes a simple **if…then…else**.

, – Special Operator

See also

Comma. Evaluates two expressions and returns the results of the second expression.

delete – Special Operator

See also

Deletes an object, a property of an object or an element in a table at the specified index.

Syntax

```
delete ObjectName
delete ObjectName.property
delete ObjectName[index]
delete property //  only allowed with a  with statement.
```

Parameters

ObjectName is the name of an object.
property is the property to be deleted.
index is a number representing the index that will be deleted in a table.

instanceOf – Special Operator

See also

Returns a Boolean value indicating if an object is an instance of a particular class.

Syntax

```
result = objet instanceof class
```

Parameters

result represents any variable.
objet represents any object.
class represents any class of a defined object.

Notes

The **instanceof** operator returns **true** if the *object* is an instance of *class*. It returns **false** in the opposite case or when the *object* has a *null* value.

new – Special Operator

See also

The **new** operator is used to create an instance of a particular object by:

- a type of predefined objects who have the constructor function,
  or
- a type of object defined by the user.

Syntax

```
ObjectName = new objectType (param1 [,param2] ...[,paramN])
```

Parameters

`ObjectName` represents the name of the new instance of an object.
`objectType` must be a function that defines a type of object.
`param1...paramN` represent the values of object's properties. These properties are parameters defines by the **objectType** function.

this – Special Operator

See also

This operator is used to make reference to a current object.

Syntax

```
this.property
```

Example :

```
function software(type, name, version)
{
  this.type = browser;
  this.name = navigator;
  this.version = 4.7;
```

typeof – Special Operator

See also

This operator returns a string indicating the type of operand. The operand can be an object, a variable, a string or a keyword, for which the type must be returned.

Syntax

```
typeof operand
typeof (operand)
```

void – Special Operator

See also

The **void** operator specifies an expression to be evaluated without returning a value. The expression to be evaluated is JavaScript code.

Syntax

```
void (expression)
void expression
```

Examples:

```
<A HREF="javascript:void(0)">Clicking here does nothing </A>
<A HREF="javascript:void(document.form.submit())">
```

Creates a hyperlink that submits a form.

Binary operators

See also

Binary operators (*bitwise operators*) treat operands as bit sets (0 and 1) without making reference to their usual representations. They are useful for testing logical conditions and not for mathematical calculation.

Syntax (Examples)

`a ^ b` (XOR)
`a | b` (OR)
`a & b` (AND)
`~a` (NOT)

Where `a` and `b` are expressions.

Operator	Description

`&`	**Bitwise AND** (AND, logical intersection)
	Returns a one in each bit position where the two values are 1.
	Example : `1100 & 1010`, returns `1000`
	`1100`
	`1010`
	`1000`

`^`	**Bitwise XOR** (exclusive OR, logical union)
	Returns a one in each bit position where only one of the two operands is 1.
	Example : `1100 ^ 1010`, returns `0110`
	`1100`
	`1010`
	`0110`

`|`	**Bitwise OR** (inclusive OR, logical union)
	Returns a one in each bit position where the two operands or one of them is 1.
	Example : `1100 | 1010`, returns `1010`
	`1100`
	`1010`
	`1010`

`~`	**Bitwise NOT** (complement)
	Returns the inverted bit at the same position for each of the bits of the operand.
	Example : `1001`, returns `0110`
	`1001`
	`0110`

`<<`	**<< Binary left shift** Shifts the bits in a expression to the left. Example : `14 << 2`, returns `56` 00001110 (14 in binary mode) 00111000 (56 in binary mode)
`>>`	**>> Sign-propagating right shift** Shifts the bits in a expression to the right and propagates the leftmost bit to freed positions. Example : `9 >> 2`, returns `2` 1001 (9 in binary mode) 10 (2 in binary mode)
`>>>`	**>>> Zero-fill right shift** Shifts bits to the right and replaces the left positions that are freed with zeros.

Example : **`19 >>> 2`**, returns **`4`**
10011 (19 in binary mode)
*00*100 (4 in binary mode)

Assignment Operators

See also

The base assignment operator is the equal (=) sign. Other operators are usually shortcuts for standard operations.

Operator	Description
=	**Assignment** Assigns the value of the second operand to the first operand. Example : x = y = z = 0 changes x, y and z to 0 after the operation is completed.
+=	**Addition** Adds two numbers and assigns the result to the first. Example : x += y returns x = x + y
-=	**Subtraction** Subtracts two numbers and assigns the result to the first. Example : x -= y returns x = x - y
*=	**Multiplication** Multiples two numbers and assigns the result to the first. Example : x *= y returns x = x * y
/=	**Division** Divides two numbers and assigns the result to the first. Example : x /= y returns x = x / y
%=	**Arithmetic modulus** Computes the modulus of 2 numbers (the remaining part of a division) and assigns the result to the first. Example : x %= y returns x % y
&=	**Bitwise AND** Performs a bitwise AND and assigns the result to the first operand.
^=	**Bitwise XOR** Performs a bitwise XOR and assigns the result to the first operand.
|=	**Bitwise OR** Performs a bitwise OR and assigns the result to the first operand.
<<=	**Left shift** Performs a left shift and assigns the result to the first operand.
>>=	**Right shift** Performs a right shift and assigns the result to the first operand.
>>>=	**Sign-propagating right shift** Performs a sign-propagating right shift and assigns the result to the first operand..

String Operators

See also

This type of operator concatenates two strings to obtain one value.

Operator	Description
+	**String addition** Concatenates two strings. Example : "string" + "addition", results in string addition.
+=	**String addition (2)**

Concatenate two strings and assigns the result to the first operand.

Example : If the MyString has a value of alpha, MyString += "bet" assigns the value of alphabet to MyString.

Comparison Operators

See also

A comparison operator compares its operands and returns a Boolean value.

Operator	Description
==	**Equal** Returns true if the two operands are equal. Example : 4 == variable1 returns true if *variable1* is 4.
!=	**Not Equal** Returns true if the operands are not equal. Example : 4 == variable1 returns true if *variable1* is anything but 4.
===	**Strict Equal** Returns true if the operands are equal and of the same type. Example : 4 === variable1 returns true if *variable1* is 4.
!==	**Strict Not Equal** Returns true if the operands area not equal and /or not of the same type. Example : 4 !== variable1 returns true if *variable1* is "4" or '4' or 4.
>	**Greater than** Returns true if the left operand is greater than the right operand. Example : 4 > variable1 returns true if *variable1* is greater than 4.
>=	**Greater than or equal to** Returns true if the left operand is greater than or equal to the right operand. Example : 4 >= variable1 returns true if *variable1* is 4 or greater.
<	**Less than** Returns true if the left operand is less than the right operand. Example : 4 < variable1 returns true if *variable1* is less than 4.
<=	**Less than or equal to** Returns true if the left operand is less than or equal to the right operand. Example : 4 <= variable1 returns true if *variable1* is 4 or less.

Logical Operators

See also

Logical operators are typically used with Boolean (logical) values; when they are, they return a Boolean value. However, the **&&** and **||** operators actually return the value of one of the specified operands, so if these operators are used with non-Boolean values, they may return a non-Boolean value.

Operator	Description
&&	**Logical AND** expression1 && expression2 If and only if the two expressions have a value of true, the resulting value is true. If one of the expressions has a value of false, the resulting value will be false.
||	**Logical OR** expression1 || expression2 If one of the two expressions or the two expressions have a value of true, the resulting value will be true. If the two expressions have a value of false, the resulting value will be false.
!	**Logical NOT** ! expression If the expression has a value of true, the resulting value will be false. If the expression has a value of false, the resulting value will be true.

Arithmetic Operators

See also

Arithmetic operators take numerical values as their operands and return a single numerical value. The standard arithmetic operators are addition (+), subtraction (-), multiplication (*), and division (/)

Operator	Description
+	**Addition** Adds two numbers. Example : 2 + 3 is 5.
++	**Increment** Adds 1 to a variable representing a number. Example : a = 2 b = ++a Here, b is 3. a = 2 b = a++ Here b is 2 because the operator is inserted after the expression.
–	**Unary negation, subtraction** As a subtraction operator, subtracts 2 numbers. Example : 5 - 2 is 3. As a unary operator, negates the value of its argument. Example : -2 Indicates a negative number.
– –	**Decrement** Subtracts 1 from a variable representing a number. Example : a = 2 b = --a Here, b is 1. a = 2 b = a-- Here b is 2 because the operator is inserted after the expression.
*	**Multiplication** Multiplies two numbers. Example : 2 * 6 is 12.
/	**Division** Divides two numbers. Example : 21 / 3 is 7.
%	**Arithmetic Modulus** Computes the integer remainder of dividing 2 numbers. Example : 20 % 3 is 2 (20 / 3 = 6, remainder 2).

Introduction to AceHTML's JavaScript 1.3 and JScript 5 Reference

AceHTML's JavaScript 1.3 and JScript 5 Reference presents a set of elements that are part of these scripting languages. However, the reference cannot replace a programming course or a tutorial. AceHTML's JavaScript 1.3 and JScript 5 Reference is a quick and easy reference tool of these languages for people who want to verify the code in their scripts while they are developing their pages and during the testing period of said material.

The following documents served as a reference as well as for verifying the information contained in AceHTML's references :

- **JavaScript 1.3 Language Reference**
  Client-Side JavaScript Guide, Netscape (developer.netscape.com)
  http://developer.netscape.com/docs/manuals/js/client/jsguide/index.htm
- **Microsoft's JScript 5 Reference**
  http://www.microsoft.com/france/scripting/jscript/download/jsdoc.exe

We encourage anyone who is interested in learning JavaScript and JScript to visit the referred sites.

Conventions:

- The majority of the help items have useful links to the following:
  - **Properties, Methods**, and **Events** (links to related elements).
  - **See also** (items related to the current item);
- The presence of the  icon in the titles indicates that said items refers to Microsoft's JScript language only.