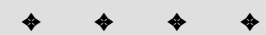
CHAPTER 32

Embedded Objects

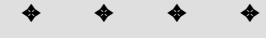


In This Chapter

Using EMBED element objects

Exploring the OBJECT element object

Understanding the unusual PARAM element



In addition to the typical content that you see in Web pages — primarily text and images — you can embed other kinds of content into the page. Such embedded content usually requires the powers of additional software, such as plug-in players or other external code processors, to load and display the content. All of this external content is added to a page by one of three HTML elements: APPLET, EMBED, or OBJECT. In the HTML 4.0 standard, the APPLET element, which was intended originally for loading Java applets, is deprecated in favor of the newer OBJECT element. An OBJECT element is intended to be more extensible, meaning that it has enough attributes and power to summon the Java virtual machine if the incoming code is a Java applet, or run an ActiveX program (in IE for Windows, that is). The EMBED element is commonly used to display a plug-in control panel directly in the document, rather than having the panel appear in a separate window.

In all cases, when a visual element is embedded via any of these elements, the control panel or applet occupies a segregated rectangular space on the page and generally confines its activities to that rectangle. But in many cases, JavaScript can also interact with the content or the player, allowing your scripts to extend themselves with powers for actions, such as controlling audio playback or the operation of a Java applet.

This chapter's primary focus is not on the content and players that you can control as it is on the HTML element objects that load the content or players into the page in the first place. Most of the properties represent nothing more than scriptable access to the element HTML attributes. The property descriptions in this chapter are therefore not extensive. Online HTML references (including the W3C HTML 4.0 specification and the Microsoft Developer Network documentation) should fill in the attribute value information quite well. In practice, scripts have very little interaction with these element objects, but if you ever need to know what's scriptable, you'll find that information here. As for controlling applets and plug-ins, you can find information about that in Chapter 44.

APPLET Element Object

For HTML element properties, methods, and event handlers, see Chapter 15.

Properties	*Methods*	*Event Handlers*
align	(Applet methods)	onCellChange
alt		onDataAvailable
altHTML		onDatasetChanged
archive		onDatasetComplete
code		onLoad
codeBase		onRowEnter
height		onRowExit
hspace		onRowsDelete
name		onRowsInserted
object		onScroll
vspace		
width		
(Applet variables)		

Syntax

Accessing APPLET element object properties or methods:

```
(NN3+/IE4+)    [window.]document.appletName.property | method([parameters])
(NN3+/IE4+)    [window.]document.applets[index].property | method([parameters])
(IE4+)         [window.]document.all.appletID.property | method([parameters])
(IE5+/NN6)     [window.]document.getElementById("appletID").property |
               method([parameters])
```

About this object

Starting with NN3 and IE4, Java applets are treated as scriptable objects. While IE4+ treats both the applet and the APPLET element as objects, NN3 and NN4 offered access to only one property of the APPLET element object (name). In NN6, however, many more APPLET object properties are also scriptable.

The fact that the applet, itself, can expose public instance variables and public methods as properties and methods of the `applet` object means that the scriptable characteristics of an `applet` object are highly dependent upon the way the applet was written. You can learn more about how to compose an applet that exposes its innards to JavaScript in Chapter 44.

Perhaps the most important point to remember about accessing applets is that you must have them loaded and running before you can address them as objects. Although you cannot query an applet to find out whether it's loaded (as you can with an image), you can rely on the `onLoad` event handler of a window to fire only when all applets in the window are loaded and running (with the occasional version- or platform-specific bug in frames, as described in the `window.onLoad` event handler discussion in Chapter 16). IE4+ also features an `onLoad` event handler for the APPLET element directly, but applets tend to be the last things to load on a page. Therefore, you won't be able to use an applet embedded in a document to help you create the HTML content of that page as it loads, but an applet can provide content for new documents or for modifiable elements of a page. With the highly dynamic object models of IE4+ and NN6, this can lead to all kinds of possibilities.

Java applets have also been used to maintain contact with a server after the page has loaded by way of a servlet running on the server. A servlet allows the applet to query or be refreshed with instantaneously updated information without having to reload the page. Of course, getting a sophisticated applet to run in a wide range of browsers and operating systems is a challenge unto itself.

A large set of event handlers for this element (all but `onLoad` and `onScroll`) is related to the application of IE/Windows data binding for PARAM elements nested inside an APPLET element. These events fire when a variety of actions occur to the data source or recordset associated with the applet. For more about applying data binding to an APPLET element, see `http://msdn.microsoft.com/workshop/author/databind/dataconsumer.asp`.

Properties

align

Value: String Read/Write

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The `align` property controls either the horizontal or vertical alignment of the element with regard to surrounding content. String values of `left` or `right` cause the applet rectangle to cling to the left or right edges of its next outermost positioning context. String values of `absbottom`, `absmiddle`, `baseline`, `bottom`, `middle`, `texttop`, or `top` influence the vertical alignment with respect to adjacent text, with the same kind of results as corresponding values of the `style.verticalAlign` property.

Related Items: `style.verticalAlign` property.

alt

Value: String Read/Write

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓					

The `alt` property represents the `ALT` attribute, which should contain text that displays in the browser in the event that the applet does not load or the user has Java turned off in the browser preferences. This information should be set as the APPLET element's attribute, because assigning text to the property after the applet attempts to load does not insert the text into the page.

Related Items: `altHTML` property.

altHTML

Value: String Read/Write

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility							✓	✓	✓

The `altHTML` property is supposed to provide an APPLET element with HTML content to render if the applet doesn't load. In practice, assigning an HTML string to this property has no effect on an APPLET element.

Related Items: `alt` property.

archive

Value: String Read/Write

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓					

The `archive` property represents the `ARCHIVE` attribute, which points to the URL of a compressed (`.zip`) file containing Java class files needed for the applet. The archive must include the class file that is assigned to the `CODE` attribute to get the applet loaded and started.

Related Items: `code` property.

code

Value: String Read-Only

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The `code` property is the URL string of the Java class file that is to begin loading the applet (or the property may be the entire applet if it consists of a single class file). You cannot change the code assigned to an applet after the element has loaded (even if the applet code did not load successfully).

Related Items: `codeBase` property.

codeBase

Value: String Read-Only

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The `codeBase` property is the string of the path on the server to the Java class file that is to begin loading the applet (or the property may be the entire applet if it consists of a single class file). The actual Java class file name is not part of the `codeBase` property.

Related Items: `code` property.

height
width

Value: Integer Read/Write

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The `height` and `width` properties represent the `HEIGHT` and `WIDTH` attributes of the APPLET element. While these values should be set via attributes in the tag, these properties can adjust the size of the applet after the fact in IE5+.

Related Items: `hspace`, `vspace` properties.

hspace
vspace

Value: Integer Read/Write

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The `hspace` and `vspace` properties represent the `HSPACE` and `VSPACE` attributes of the APPLET element, which control the number of pixels of transparent padding around the APPLET element on the page. While these values should be set via attributes in the tag, these properties can adjust the size of the applet padding after the fact in IE5+.

Related Items: `height`, `width` properties.

name

Value: String Read-Only

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The `name` property represents the `NAME` attribute, a holdover from the early implementations of the APPLET element before `ID` attributes were used to identify elements. The value assigned to the `NAME` attribute is the name you can use to reference applets in all browsers that support accessing applets: `document.`*`appletName`*.

object

Value: String Read-Only

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓					

The `object` property represents the `OBJECT` attribute, which, according to the W3C HTML standard, points to the URL of a serialized (that is, "saved") version of the applet's current state. This attribute, and thus the associated property, may not be fully implemented in NN6.

Related Items: `code` property.

vspace

See `hspace`.

width

See `height`.

OBJECT Element Object

For HTML element properties, methods, and event handlers, see Chapter 15.

Properties	*Methods*	*Event Handlers*
align	(Object methods)	onCellChange
alt		onDataAvailable
altHTML		onDatasetChanged
archive*		onDatasetComplete
BaseHref		onLoad
border*		onRowEnter
classid		onRowExit
code		onRowsDelete
codeBase		onRowsInserted
codeType		onScroll
contentDocument*		
data*		
declare*		
form*		
height		
hspace		
name		
object		
standby*		
type		
useMap*		
vspace		
width		
(Object variables)		

*See Text.

Syntax

Accessing OBJECT element object properties or methods:

```
(IE4+)          [window.]document.all.objectID.property | method([parameters])
(IE5+/NN6)      [window.]document.getElementById("objectID").property |
                method([parameters])
```

About this object

The OBJECT element is intended to be the primary way to add external content (that is, content that the browser itself does not render) to a page. For example, IE/Windows uses it to load ActiveX controls (whether from the server or locally). The OBJECT element is also destined to replace usage of the APPLET and EMBED elements.

As with the APPLET element object, scripts can frequently control the programs and plug-ins that get loaded into the browser through the OBJECT tag. Chapter 44 shows you how to do that for common objects. The property listings here are merely for the properties of the element, most of which mimic the attributes available for the OBJECT element. Even though the properties are exposed, they are very rarely scripted, except perhaps to adjust the size of the space occupied by a media controller. Most properties are read-only after their values are set by attributes in the element's tag. But if your scripts are creating the OBJECT element anew, then scripts can set the property values the first time to initialize the object.

In the list of properties that begins this object's coverage, several are marked with an asterisk (*). These properties are defined in the W3C DOM Level 2 specification, and placeholders are included in the NN6 code. But as of this writing, there is no indication that these properties are "connected."

A large set of event handlers for this element (all but `onLoad` and `onScroll`) is related to the application of IE/Windows data binding for PARAM elements nested inside an OBJECT element. These events fire when a variety of actions occur to the data source or recordset associated with the program associated with the element. For more about applying data binding to an OBJECT element, see `http://msdn.microsoft.com/workshop/author/databind/dataconsumer.asp`.

Properties

align

Value: String Read/Write

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The `align` property controls either the horizontal or vertical alignment of the element with regard to surrounding content. String values of `left` or `right` cause the object rectangle to cling to the left or right edges of its next outermost positioning context. String values of `absbottom`, `absmiddle`, `baseline`, `bottom`, `middle`, `texttop`, or `top` influence the vertical alignment with respect to adjacent text, with the same kind of results as corresponding values of the `style.verticalAlign` property.

Related Items: `style.verticalAlign` property.

alt

Value: String Read/Write

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓					

The `alt` property represents the `ALT` attribute, which should contain text that displays in the browser in the event that the object or its data do not load. This information should be set as the OBJECT element's attribute, because assigning text to the property after the object attempts to load does not insert the text into the page.

Related Items: `altHTML` property.

altHTML

Value: String Read/Write

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility							✓	✓	✓

The `altHTML` property is supposed to provide an OBJECT element with HTML content to render if the object doesn't load. In practice, assigning an HTML string to this property has no effect on an OBJECT element.

Related Items: `alt` property.

BaseHref

Value: String Read-Only

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility							✓	✓	✓

The `BaseHref` property returns the full URL path to the current document.

Related Items: None.

classid

Value: String Read-Only

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility							✓	✓	✓

The `classid` property represents the `CLASSID` attribute of the OBJECT element. IE for Windows uses this attribute to assign the Globally Unique ID (GUID) of an ActiveX control. For example, to load a (nearly) invisible Windows Media Player object into a page, the HTML is as follows:

```
<OBJECT ID="medPlayer" WIDTH="1" HEIGHT="1"
CLASSID="CLSID:22d6f312-b0f6-11d0-94ab-0080c74c7e95"
CODEBASE="#Version=1,0,0,0">
```

If your script then accesses the `classid` property of the `medPlayer` object, the value returned is the complete string as assigned to the attribute:

```
CLSID:22d6f312-b0f6-11d0-94ab-0080c74c7e95
```

Note that the `CLSID:` prefix is also part of the string value. Even if the object does not load (for example, because the object is missing or an error is in the long `CLASSID` string), the property value reports the value as assigned to the attribute.

The HTML 4.0 specification indicates that the `CLASSID` attribute be used for any kind of external class files, including Java applets. But in practice, IE wants applet URLs supplied to the `CODE` attribute (a non-HTML 4.0 attribute).

Related Items: `code` property.

code

Value: String Read-Only

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The `code` property is the URL string of a Java class file that is to begin loading the applet (or the property may be the entire applet if it consists of a single class file). You cannot change the code assigned to an applet after the element has loaded (even if the applet code did not load successfully).

Related Items: `codeBase` property.

codeBase

Value: String Read-Only

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The codeBase property is the string of the path on the server to the source of the applet or ActiveX control referenced by the CLASSID or CODE attributes. IE4+ also uses the CODEBASE attribute to specify a minimum version of control that is to load, if the attribute is available. This facet is discussed in Chapter 28's coverage of plug-in detection for IE/Windows.

Related Items: code property.

codeType

Value: String Read-Only

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The codeType property is a string of the MIME type of whatever object is pointed to by the CODE attribute value.

Related Items: type property.

height
width

Value: Integer Read/Write

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The height and width properties represent the HEIGHT and WIDTH attributes of the OBJECT element. While these values should be set via attributes in the tag, these properties can adjust the size of the embedded element after the fact in IE5+.

Related Items: hspace, vspace properties.

hspace
vspace

Value: Integer Read/Write

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The `hspace` and `vspace` properties represent the `HSPACE` and `VSPACE` attributes of the OBJECT element, which control the number of pixels of transparent padding around the OBJECT element on the page. While these values should be set via attributes in the tag, these properties can adjust the size of the padding around the element after the fact in IE5+.

Related Items: `height`, `width` properties.

name

Value: String Read-Only

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The `name` property represents the `NAME` attribute of the OBJECT element. The better form is to assign an ID to the OBJECT element and use accepted reference syntax for element ids.

Related Items: None.

object

Value: External Object Read-Only

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The object property returns a reference to the object contained by the OBJECT element. This property is essential if the program running inside the OBJECT element has the same property or method names as the OBJECT element itself. For example, consider a Java applet loaded into the OBJECT element as follows:

```
<OBJECT CODE="coolApplet" ID="myAPPLET" ... >
```

If the applet code contained a public variable called height, an attempt to read or write that property through the OBJECT element will cause the element's properties to be read, and not the applet's properties. Therefore, if you insert the object property in the reference, the script reaches into the applet object for the property:

```
document.getElementById("myAPPLET").object.height = 40
```

If there is no ambiguity between element and object property and method names, the browser looks first at the element and then the object to find a match.

Related Items: None.

type

Value: String Read-Only

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The type property represents the TYPE attribute of the OBJECT element, which, in theory anyway, is intended to warn the browser about the MIME type of data that is to be loaded into the object's process. I say "in theory" because the HTML 4.0 specification links the TYPE attribute to the DATA attribute, which points to the data to be loaded to support whatever program code is loaded via the CLASSID or CODE attribute. But through IE5.5, there is no support for the DATA attribute.

Related Items: codeType property.

vspace

See hspace.

width

See height.

EMBED Element Object

For HTML element properties, methods, and event handlers, see Chapter 15.

Properties	*Methods*	*Event Handlers*
`align`	(Object methods)	`onLoad`
`height`		`onScroll`
`hidden`		
`name`		
`pluginspage`		
`src`		
`units`		
`width`		
(Object variables)		

Syntax

Accessing EMBED element object properties or methods:

```
(IE4+)        [window.]document.all.objectID.property | method([parameters])
(IE5+/NN6)    [window.]document.getElementById("objectID").property |
              method([parameters])
```

About this object

An EMBED element is a carryover from the early browser days. Although never adopted by the W3C HTML standard, the EMBED element has been used in NN and IE as a way to embed non-native content (for example, sounds, video clips, and custom MIME types for plug-ins, such as Shockwave) into a page. What gets embedded into the page is the controller or viewer for whatever kind of data the EMBED element points to (via the SRC attribute).

The EMBED element is far less sophisticated than the OBJECT element, but current browsers continue to support it. If you have been using the EMBED element in previous applications, it may be a good idea to start gravitating toward the OBJECT element. For backward compatibility purposes, nesting an EMBED element inside an OBJECT element is not uncommon, both of which attempt to load the same content

and plug-in. Browsers that know about the OBJECT element will load the content that way; older browsers will use the EMBED element and its attributes and parameters.

Because an EMBED element loads a plug-in (including ActiveX control types of plug-ins in IE/Windows), you can reference the plug-in's properties and methods through the EMBED object's reference.

Properties

align

Value: String Read/Write

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The `align` property controls either the horizontal or vertical alignment of the element with regard to surrounding content. String values of `left` or `right` cause the object rectangle to cling to the left or right edges of its next outermost positioning context. String values of `absbottom`, `absmiddle`, `baseline`, `bottom`, `middle`, `texttop`, or `top` influence the vertical alignment with respect to adjacent text, with the same kind of results as corresponding values of the `style.verticalAlign` property.

Related Items: `style.verticalAlign` property.

height
width

Value: Integer Read/Write

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The `height` and `width` properties represent the `HEIGHT` and `WIDTH` attributes of the EMBED element. While these values should be set via attributes in the tag, these properties can adjust the size of the element after the fact in IE5+.

Related Items: None.

hidden

Value: Boolean Read/Write

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility							✓	✓	✓

The `hidden` property represents the `HIDDEN` attribute of the EMBED element. When an EMBED element is hidden, neither controller nor the content is shown. Application of this element in modern browsers should use style sheets to hide and show the element.

Related Items: `style.visibility` property.

name

Value: String Read-Only

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The `name` property represents the `NAME` attribute of the EMBED element. The better form is to assign an ID to the EMBED element and use accepted reference syntax for element ids.

Related Items: None.

pluginspage

Value: String Read-Only

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility							✓	✓	✓

The `pluginspage` property represents the `PLUGINSPAGE` attribute of the EMBED element. This attribute is a URL that gets applied to a link in the browser if the plug-in associated with the external file's MIME type cannot be found on the client.

Related Items: None.

src

Value: String Read/Write

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility				✓			✓	✓	✓

The `src` property represents the `SRC` attribute of the EMBED element. This attribute points to the external file that is to be loaded into the browser via the associated plug-in. Scripts can assign a new URL string to this property to load a different file into the current plug-in.

Related Items: None.

units

Value: String Read-Only

	NN2	NN3	NN4	NN6	IE3/J1	IE3/J2	IE4	IE5	IE5.5
Compatibility							✓	✓	✓

The `units` property returns the unit of measure assigned with the length value of the `height` and `width` properties. In IE4, this property returned only `px`. The property does not appear to be connected in IE5.5, so it is probably deprecated in IE.

Related Items: `height`, `width` properties.

The Odd Case of the PARAM Element

HTML pages pass parameters to Java applets, plug-ins, and ActiveX controls by way of PARAM elements that are nested inside APPLET, EMBED, and OBJECT elements. Although a PARAM element object is defined by the W3C DOM Level 2 specification, it does not show up on some browsers' radar when you try to reference the PARAM element by itself. Even assigning an ID to a PARAM element or using `document.getElementsByTagName("PARAM")` fail to allow references to access an individual PARAM element object. At most, you can retrieve the `innerHTML` property of the surrounding element. But even here, the values returned may not necessarily be precisely the HTML you specify in the document.

In practice, this limitation is not particularly important. For one thing, even if you could access the PARAM elements of an embedded object or program, attempts to modify the values would be wasted: Those values are read at load time only. Secondly, a well-designed plug-in, applet, or ActiveX control will provide its own properties or methods to retrieve the current settings of whatever properties are initialized via the PARAM elements.

✦ ✦ ✦