CHAPTER 20

HTML Directive Objects

Thanks to the modern browser's desire to expose all HTML elements to the document object model, we can now (in IE4+ and NN6) access a variety of objects that represent many HTML elements that are normally invisible to the human viewer of a page. These elements are called *directive elements* because they predominantly contain instructions for the browser — instructions that direct the browser to locate associated content on the page, link in external specifications, treat content as executable script statements, and more.

As you browse through the objects of this chapter, you may wonder why they have so many properties that normally indicate that the elements occupy space on the rendered page. After all, how can a META element have dimension or position on the page when it has no renderable content? The reason is that modern browsers internally employ some form of object-oriented behavior that lets all HTML elements — rendered or not — inherit the same set of properties, methods, and event handlers that any generic element has (see Chapter 15). The logical flaw is that unrendered elements can have properties and methods that don't genuinely apply to them. In such cases, their property values may be zero, an empty string, or an empty array. Yet the properties and methods exist in the objects just the same. Therefore, despite the large number of objects covered in this chapter, there are relatively few properties and methods that are not shared already with all HTML elements (as covered in Chapter 15).

In This Chapter

Accessing non-displayed element objects

Linking operating-system specific style sheet definitions

HTML, HEAD, LINK, TITLE, META, BASE, and SCRIPT elements

HTML Element Object

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

Properties	***Methods***	***Event Handlers***
`version`		

Syntax

Accessing HTML element object properties or methods:

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

About this object

The HTML element is the big wrapper around all other elements of the page. In the object tree, the HTML element sits between the all-encompassing `document` object and the element's most common children, the HEAD and BODY elements. Other than one deprecated property (`version`), the HTML element object offers nothing of importance to the scripter — with one possible exception. When your script needs to use methods on the child nodes of the HTML element, you must invoke most of those methods from the point of view of the HTML element. Therefore, you should know how to create a reference to the HTML element object (shown in the preceding "Syntax" section) just in case you need it.

Property

`version`

Value: String Read-Only

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

The `version` property is an artifact of an "ancient" way an HTML document used to specify the HTML version of its content. These days, the preferred way to declare the HTML version for a document is through a Document Type Declaration (DTD) statement that precedes the `<HTML>` tag. An example of a modern DTD statement that accommodates HTML 4 plus deprecated elements and attributes as well as frameset support is

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
                  "http://www.w3.org/TR/REC-html40/frameset.dtd">
```

See `http://www.w3.org/TR/REC-html40/struct/global.html#h-7.2` for several other possibilities. A DTD statement does not affect the `version` property of an HTML element object.

Related Items: None.

HEAD Element Object

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

Properties	***Methods***	***Event Handlers***
`profile`		

Syntax

Accessing HEAD element object properties or methods:

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

About this object

The purpose of the HEAD element is primarily to act as a container for most of the other HTML directive elements. Other than as a reference point to the child elements nested within, the HEAD element object rarely comes into play when scripting a document.

Properties

`profile`

Value: String Read-Only

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

The `profile` property is the script version of the optional `PROFILE` attribute of a HEAD element. While the attribute and property are supported in NN6 (that is, they exist), they are not used in practice yet. You can find details about the attribute at `http://www.w3.org/TR/REC-html40/struct/global.html#profiles`.

Related Items: META element object.

BASE Element Object

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

Properties	*Methods*	*Event Handlers*
`href`		
`target`		

Syntax

Accessing BASE element object properties or methods:

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

About this object

The BASE element enables the page author to specify a default server directory and/or link target for the entire page. If you omit the BASE element from the HTML, browsers use the current page's path as the base URL and the current window or frame as the default target. Occasionally, a page generated entirely by way of `document.write()` has difficulty establishing the same BASE URL as the document that generates the content, particularly if the primary page is written out by a server script (in Perl or in another language). Including a `<BASE>` tag in the dynamically written new page solves the problem; the new page can fetch images or other external elements via relative URLs within the page.

The two distinctive properties of the BASE element object are rarely scripted, if ever.

Properties

`href`

Value: URL String Read/Write

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

The `href` property is generally an absolute URL to the directory you wish to declare as the default directory for the page. Even though browsers automatically set the BASE HREF to the document's own directory, this object and property do not have any values unless you explicitly set them in a `<BASE>` tag. In IE, changing this property after a page loads causes the page to re-resolve all relative URLs on the page to the new BASE HREF. Therefore, if images have relative URLs assigned to their `src` properties (either by way of the tag attribute or script), a change to the BASE element's `href` property forces the browser to look for those same relative URLs in the new directory. If the files aren't there, then the images show up broken on the page.

On the CD-ROM

Example on the CD-ROM

`target`

Value: String Read/Write

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

The `target` property governs the default window or frame that is to receive any content coming from a server in response to a click on a link or any other element that has its own `TARGET` attribute. Valid values include the name of any frame (as assigned to the `NAME` attribute of the `<FRAME>` tag) or window (as defined by the second attribute of the `window.open()` method). You can also assign standard HTML targets (`_blank`, `_parent`, `_self`, and `_top`) to this property as strings.

On the CD-ROM

Example on the CD-ROM

BASEFONT Element Object

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

Properties	*Methods*	*Event Handlers*
`color`		
`face`		
`size`		

Syntax

Accessing BASEFONT element object properties or methods:

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

About this object

The BASEFONT element enables authors to define a font face, size, and color for an entire section of an HTML document — or the entire document. Although page authors still frequently use the BASEFONT element, font control in modern browsers should fall in the hands of style sheets. (The element is deprecated in

HTML 4.0.) The paradox of this is that the BASEFONT element is accessible as a scriptable object only in browsers that support style sheets. Even so, I recommend avoiding dynamic font changes by way of the BASEFONT element and use scripts to control style sheets instead.

Note

The BASEFONT element has no end tag, so IE's `outerHTML` property consists of all HTML in the document starting with the element itself.

The three distinctive properties of the BASEFONT element object are rarely, if ever, scripted.

Properties

color
face
size

Value: String Read/Write

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

These three properties define the characteristics of font rendering for all content following the element's tag in the document. Color specifications can be hexadecimal triplets or Netscape color names (a list is available at `http://developer.netscape.com/docs/manuals/htmlguid/colortab.htm`). Font faces can include a list of comma-separated font face names. And because this is HTML as opposed to style sheet fonts, the size property is in terms of the 1 through 7 scale of font sizes. You can also use relative sizes (for example, +1).

On the CD-ROM

Example on the CD-ROM

ISINDEX Element Object

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

Properties	***Methods***	***Event Handlers***
`alt`		
`border`		
`checked`		
`complete`		
`dynsrc`		
`form`		
`height`		
`hspace`		
`indeterminate`		
`loop`		
`lowsrc`		
`maxLength`		
`name`		
`prompt`		
`readOnly`		
`size`		
`start`		
`status`		
`value`		
`vrml`		
`vspace`		
`width`		

Syntax

Accessing ISINDEX element object properties or methods:

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

About this object

The ISINDEX element is a holdover from the early beginnings of HTML. It offered the first text input field prior to the addition of FORM and INPUT elements to the HTML specification. IE treats this element as if it were an INPUT element, so ISINDEX takes on all possible INPUT element properties (including those of buttons). This element is deprecated in HTML 4.0 and should not be part of your development vocabulary. Use forms and genuine INPUT elements instead (see Chapters 23–26).

LINK Element Object

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

Properties	***Methods***	***Event Handlers***
`charset`		`onLoad`
`disabled`		
`href`		
`hreflang`		
`media`		
`rel`		
`rev`		
`styleSheet`		
`target`		
`type`		

Syntax

Accessing LINK element object properties or methods:

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

About this object

The LINK element (not to be confused with the A element that is often referred to as a "link" element when it contains an `HREF` attribute pointing to another document) has many potential uses in pointing to external documents that relate to the current document. Its most common usage today is for linking an external style sheet specification to the document. In fact, it's not uncommon for sophisticated site designs to use `document.write()` to generate the `<LINK>` tag so that operating-system specific style sheets are applied to the page. In the following code fragment (which goes inside a document's HEAD element), the page loads a Macintosh-specific style sheet when the page is running on a Macintosh; otherwise, it loads a Windows-specific style sheet:

```
<SCRIPT LANGUAGE="JavaScript">
var isMac = navigator.userAgent.indexOf("Mac") != -1
var linkTagStart = "<LINK REL='stylesheet' TYPE='text/css' HREF='"
var linkTagEnd = ".css'>"
if (isMac) {
    document.write(linkTagStart + "mac" + linkTagEnd
} else {
    document.write(linkTagStart + "windows" + linkTagEnd
}
</SCRIPT>
```

While it may appear that the LINK element can load a variety of content into a page, do not use it for multimedia (in which case you should use the EMBED or OBJECT elements) or external HTML (where you should use an IFRAME element).

Many of the properties of the LINK element object are script representations of HTML 4.0 attributes for the element. However, browsers don't take full advantage of the possibilities available from the LINK element yet. (For example, a browser can provide arrows to the previous and next documents in a series, as specified by the `REV` and `REL` attributes. But so far, no browser implements this.) Properties unique to this object offer scripted access (in various browser versions) to attribute values of the LINK element. Therefore, this chapter does not spend a lot of time on properties that are not in current use.

Properties

charset

Value: String Read/Write

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

The `charset` property advises the browser about the character encoding of the content that will arrive from the external document (assuming you also have the HREF attribute set). Values for this property must match the encoding naming conventions defined in an industry standard registry (`ftp://ftp.isi.edu.innotes/iana/assignments/character-sets`).

disabled

Value: Boolean Read/Write

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

By changing the `disabled` property (default is `false`), you can turn externally linked content on and off. For example, you can define two different style sheet links in a document that has two `<LINK>` tags with one's `DISABLED` attribute set. You can switch between the two style sheets by setting the `disabled` property of one to `true` and the other to `false`.

href

Value: String See Text

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

Another way to swap style sheets is to modify the value of a single LINK element object's `href` property (although the property is read-only in IE4+/Mac and NN6). The property's value is a URL string.

hrefLang

Value: String Read/Write

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

The `hrefLang` property is an advisory for the browser (if the browser takes advantage of it) about the written language used for the content to which the LINK element's `HREF` attribute points. Values for this property must be in the form of the standard language codes (for example, `en-us` for U.S. English).

media

Value: String Read/Write

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

The `media` property (not available in IE4/Mac) is an advisory for the browser about the target output device intended for the content to which the LINK element's `HREF` attribute points. This is an outgrowth of HTML 4.0 efforts to make way for future browsers and content that can be optimized for devices such as printers, handheld computers, and audio digitizers. The W3C specifies a preliminary set of constant string values for this property's equivalent attribute. So far, browsers (at most) recognize `all` (default), `print`, and `screen`.

rel
rev

Value: String Read/Write

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

The rel and rev properties are intended to define relationships in the forward and back directions with respect to the current document. Browsers have yet to exploit most of the potential of these attributes and properties. For the most part, the attributes solely direct the browser to treat the external content as a style sheet definition file.

A long list of values are predefined for these properties, based on the corresponding attribute values specified in HTML 4.0. If the browser does not respond to a particular value, the value is simply ignored. You can string together multiple values in a space-delimited list inside a single string. Accepted values are as follows:

`alternate`	`contents`	`index`	`start`
`appendix`	`copyright`	`next`	`stylesheet`
`bookmark`	`glossary`	`prev`	`subsection`
`chapter`	`help`	`section`	

styleSheet

Value: Object Read-Only

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

When a LINK element loads an external style sheet, the IE-specific styleSheet property of the LINK element object provides scripted access to the style sheet rules that belong to that external file. Use properties of the styleSheet object (see Chapter 30) to access specifics about the imported rules.

target

Value: String Read/Write

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

In the context of using LINK elements to point to other content associated with the current document (for example, the next and previous documents within a series), the `target` property can advise the browser which frame or window to use to display that content. For example, a suitably equipped browser can display a glossary in a separate window. No browsers currently implement these extended features of the LINK element, so the property is provided in browsers only for compatibility with the W3C standards. If the property were truly functional, it would accept values in the form of a string name for a frame or one of the window constants (`_blank`, `_parent`, `_self`, or `_top`).

type

Value: String Read/Write

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

The `type` property specifies the MIME type for the content that will arrive from the external document to which the element's `HREF` attribute points. LINK elements are used primarily for Cascading Style Sheets, so the property value is `text/css`.

Event handlers

onLoad

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

The `onLoad` event handler fires when the external content pointed to by the LINK element's `HREF` attribute completes loading. IE5 for Windows fires this event handler even if the loading does not succeed, so use this event handler with care.

META Element Object

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

Properties	***Methods***	***Event Handlers***
`charset`		
`content`		
`httpEquiv`		
`name`		
`url`		

Syntax

Accessing META element object properties or methods:

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

About this object

In computer terminology, *metadata* usually consists of extra information about the primary data of a document or information collection. In HTML documents, metadata can be additional hidden information about the document, such as the name of the author and keywords. If the browser is suitably equipped, metadata can also include some instructions, such as when to reload the page by itself. META elements add all of this metadata to HTML documents. Both fact and folklore surround the application of META elements within pages. One fact is that Internet search engine robots scour pages for certain kinds of keyword meta tags to help place your page within relevant categories when Web surfers are looking for specific content. More on the folklore side is that browsers always respond to META element wording that prevents browsers from copying pages into the cache — when in fact, this behavior is not universal among browsers.

Complete details about META element usage is beyond the scope of this JavaScript book, but you should be aware of one composition that enables you to set a page to reload itself (or another page) at a fixed time interval. This is especially useful if your page retrieves very timely information from a database. The format is

```
<META HTTP-EQUIV="refresh" CONTENT="n,url=url">
```

n is the number of seconds to delay before reloading the page, and `url` is the complete URL of the page to be reloaded. Note that you can specify any page you like. This allows for a kind of slide show to be sequenced in a freestanding kiosk, as each page's META element points to the next page in the series after a fixed amount of time.

Unique properties for the META element object mimic the HTML attributes for the `<META>` tag. These properties are rarely, if ever, accessed from a script, so I mention them here only briefly.

Properties

charset

Value: String Read/Write

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

The `charset` property advises the browser about the character encoding of the content for the page. Values for this property must match the encoding naming conventions defined in an industry standard registry (`ftp://ftp.isi.edu.innotes/iana/assignments/character-sets`).

content

Value: String Read/Write

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

For many applications of the META element, the `content` property contains the primary value associated with the element. For example, search engines look for a META element whose `NAME` attribute is `"keywords"`. The value of the `CONTENT` attribute is a comma-delimited string of keywords that the search engine reads and indexes in its own database. The `content` property simply represents the `CONTENT` attribute string. Changing the values by script obviously does nothing to alter the tag values of the page on the server.

httpEquiv

Value: String Read/Write

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

A META element can simulate and extend the transmission of server instructions to the browser — instructions that normally arrive in the form of http headers. These header supplements are supplied in META elements via the `HTTP-EQUIV` attribute, which is represented in the object model by the `httpEquiv` property. Common values include `refresh` and `expires`. Each of these also requires a `CONTENT` attribute that provides necessary details for carrying out the instructions. If you assign a string value to the `httpEquiv` property, be sure the `content` property has a suitable string assigned to it.

name

Value: String Read/Write

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

A META element that includes genuine metadata about the page (for example, author or keywords) usually has a `NAME` attribute that identifies what the metadata is (analogous to the name of a name/value pair). The `name` and `content` properties go hand in hand because the content string usually must be in a particular form for an external process (for example, a search engine) to read the data successfully. Values for the `name` attribute are rarely case-sensitive.

url

Value: String Read/Write

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

If a META element needs to point to a document on the Internet for any reason, the URL of that document is assigned to the `URL` attribute of the element. You can modify the value via the `url` property of a META element object. I recommend a complete URL string for the `url` property value.

SCRIPT Element Object

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

Properties	***Methods***	***Event Handlers***
`defer`		
`event`		
`htmlFor`		
`language`		
`src`		
`text`		
`type`		

Syntax

Accessing SCRIPT element object properties or methods:

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

About this object

The `<SCRIPT>` tag is well known to scripters, and modern browsers (IE4+ and NN6) treat the SCRIPT element as an object that, itself, can be scripted. The circularity of this description isn't as far fetched as it sounds. While scripting an existing script is a rarity in practice, it is not out of the question to generate a new SCRIPT element after the page loads. If you use W3C DOM syntax to create a new SCRIPT element, you then need to assign values to the properties that are normally set via the tag's attributes. Thus, scripting a script does make sense.

Unless you have experience with IE's option of binding event handlers to `<SCRIPT>` tags (see Chapter 14), some of the properties described next will be foreign to you. Even so, these properties are now a part of the W3C DOM specification, so they are implemented in NN6 as well.

One property to take special note of is `language`. This property name conflicts slightly with the `language` property that is part of all HTML element objects. The preferred way to specify the language of the script statements inside the element is to set the `TYPE` attribute to a MIME type. Unfortunately, this technique does not distinguish among versions of JavaScript. Also, for backward compatibility, I advise you to continue using the `LANGUAGE` attribute as well because only IE4+ and NN6+ recognize the `TYPE` attribute.

Note

Microsoft developer documentation states that the SCRIPT element object has an `onLoad` event handler. If that assertion is true, then it is broken in IE4 and IE5.

Properties

defer

Value: Boolean Read/Write

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

The default process of loading a page that contains scripts is to wait for any immediate script execution to complete before the rest of the page loads. But if you include a `DEFER` attribute in the tag, modern browsers continue to load the rest of the page without waiting for immediate scripts to run. The `defer` property enables you to inspect or set that property; its default value is `false`. Once a page loads, any changes you make to an existing SCRIPT element's `defer` property has no effect.

event
htmlFor

Value: String Read-Only

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

Modern browsers enable you to bind events to script statements when you specify both a `FOR` and `EVENT` attribute in the `<SCRIPT>` tag. Statements inside the tag execute only when the object named by the `FOR` attribute receives the event named by the `EVENT` attribute. You can examine the `EVENT` attribute by way of the SCRIPT element object's `event` property, and you can view the `FOR` attribute through the `htmlFor` property. Both properties simply mimic whatever values are assigned to their respective attributes, such as `onClick()` and `myDIV`.

language

Value: String Read/Write

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

Use the `language` property to get or set the name of the scripting language specified for a SCRIPT element object. Even though NN and IE browsers default to JavaScript (or some equivalent), the property has no value unless you set the `LANGUAGE` attribute in the `<SCRIPT>` tag. If you must specify a particular version of JavaScript, you can do so by appending the version number immediately after the language name:

```
document.getElementById("myScript").language = "JavaScript1.5"
```

IE accepts several language names as values for this property: `JavaScript`, `JScript`, `VBScript`, and `VBS`. For IE5, `XML` is also accepted.

Also see the `type` property.

src

Value: String Read-Only

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

The `src` property is a string of the URL of an external `.js` script file to be linked into a page. You cannot change this property after you load the external script.

text

Value: String Read-Only

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

The full text of a SCRIPT element is available for reading through the `text` property. While IE5 may give the impression that you can modify this property, the script that loads with the page is what is stored in the browser's memory. Thus, the original script statements continue to work even though the object's property is different.

type

Value: String Read-Only

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

The `TYPE` attribute was added to the `<SCRIPT>` tag in HTML 4.0 to help resolve the conflict that the `LANGUAGE` attribute created for all HTML elements. The value of the attribute (and thus the `type` property) is a MIME type string. For JavaScript, that value is `text/javascript`.

TITLE Element Object

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

Properties	*Methods*	*Event Handlers*
`text`		

Syntax

Accessing TITLE element object properties or methods:

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

About this object

Before the TITLE element was accessible to scripting as an object, the prescribed way to get to the content of the page's `<TITLE>` tag was through the `document.title` property. While that property is still available for backward compatibility, scripts written exclusively for newer browsers should access the `text` property of the TITLE element object. As a useful exercise, you can modify Listing 18-17 (loaded via Listing 18-16) to use the IE4+ or W3C DOM syntax to retrieve and display the document's title.

Property

text

Value: String Read/Write

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

The `text` property represents the text between the start and end tags of the TITLE element object. This is simply a convenience property because the text can be referenced by other ways in IE4+ (`innerText` property), NN6 (`innerHTML`), and W3C DOM (`firstChild.nodeValue`) syntaxes. For backward compatibility with earlier browsers, you can alternatively use the `document.title` property.

Related Items: `document.title` property.

✦ ✦ ✦