New Page Object Model Objects

Visit the Office Developer Center on the Microsoft Developer Network Web site for the latest information about programming with Microsoft Office FrontPage 2003, including product news, technical articles, downloads, and samples.

The following table lists objects added to the Office FrontPage 2003 object model.

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DynamicTemplateState</td>
<td>Represents an object that stores mapping information related to a Dynamic Web Template.</td>
</tr>
<tr>
<td>FPHTMLTemplateRegionElement</td>
<td>Represents a region in an HTML document that corresponds to a region in a Dynamic Web Template.</td>
</tr>
<tr>
<td>FPHTMLWebPartElement</td>
<td>Represents a Web part in an HTML document.</td>
</tr>
<tr>
<td>FPHTMLWebPartZoneElement</td>
<td>Represents a Web Part zone in an HTML document.</td>
</tr>
<tr>
<td>FPHTMLXSLElement</td>
<td>Represents an Extensible Stylesheet Language (XSL) element for an XSL Data View Web Part.</td>
</tr>
<tr>
<td>FPHTMLXSLWebPartElement</td>
<td>Represents an Extensible Stylesheet Language (XSL) Data View Web Part.</td>
</tr>
<tr>
<td>FPLLayoutRegion</td>
<td>Represents a two dimensional region in a document that is specified as containing special table cell formatting, referred to as layout cells, layout tables, and cell formatting tables.</td>
</tr>
<tr>
<td>FPLLayoutTablesAndCells</td>
<td>Represents a collection of FPLLayoutRegion objects.</td>
</tr>
<tr>
<td><strong>IFPLayoutRegion</strong></td>
<td>special table cell formatting, referred to as layout cells, layout tables, and cell formatting tables.</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>IFPLayoutTablesAndCells</strong></td>
<td>Represents a collection of <strong>IFPLayoutRegion</strong> objects.</td>
</tr>
<tr>
<td><strong>IHTMLTemplateRegionElement</strong></td>
<td>Represents a region in an HTML document that corresponds to a region in a Dynamic Web Template.</td>
</tr>
<tr>
<td><strong>IHTMLWebPartElement</strong></td>
<td>Represents a Web part in an HTML document.</td>
</tr>
<tr>
<td><strong>IHTMLWebPartZoneElement</strong></td>
<td>Represents a Web Part zone in an HTML document.</td>
</tr>
<tr>
<td><strong>IHTMLXSLElement</strong></td>
<td>Represents an Extensible Stylesheet Language (XSL) element for an XSL Data View Web Part.</td>
</tr>
<tr>
<td><strong>IHTMLXSLWebPartElement</strong></td>
<td>Represents an Extensible Stylesheet Language (XSL) Data View Web Part.</td>
</tr>
<tr>
<td><strong>SearchInfo</strong></td>
<td>Provides access to programmatic search and replace functionality to pages in Microsoft FrontPage.</td>
</tr>
</tbody>
</table>
New Page Object Model Properties  
(Alphabetical List)

Visit the Office Developer Center on the Microsoft Developer Network Web site for the latest information about programming with Microsoft Office FrontPage 2003, including product news, technical articles, downloads, and samples.

The following table lists properties added to the Office FrontPage 2003 object model (sorted alphabetically).

<table>
<thead>
<tr>
<th>New Property</th>
<th>Object(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>allowCustomization</td>
<td>FPHTMLWebPartZoneElement, IHTMLWebPartZoneElement</td>
</tr>
<tr>
<td>allowPersonalization</td>
<td>FPHTMLWebPartZoneElement, IHTMLWebPartZoneElement</td>
</tr>
<tr>
<td>assembly</td>
<td>FPHTMLXSLWebPartElement, IHTMLWebPartElement</td>
</tr>
<tr>
<td>behaviorElement</td>
<td>FPHTMLDocument, IFPDocument</td>
</tr>
<tr>
<td>cellFormattingID</td>
<td>FPLLayoutRegion, IFPLayoutRegion</td>
</tr>
<tr>
<td>containerWidth</td>
<td>FPHTMLWebPartZoneElement, IHTMLWebPartZoneElement</td>
</tr>
<tr>
<td>dynamicTemplate</td>
<td>IFPDocument</td>
</tr>
<tr>
<td>Find</td>
<td>SearchInfo</td>
</tr>
<tr>
<td>frameType</td>
<td>FPHTMLWebPartZoneElement, IHTMLWebPartZoneElement</td>
</tr>
<tr>
<td>InternalRep</td>
<td>DynamicTemplateState</td>
</tr>
<tr>
<td>isCell</td>
<td>FPLLayoutRegion, IFPLayoutRegion</td>
</tr>
<tr>
<td>isTable</td>
<td>FPLLayoutRegion, IFPLayoutRegion</td>
</tr>
<tr>
<td>layoutTablesAndCells</td>
<td>FPHTMLDocument, IFPDocument</td>
</tr>
<tr>
<td>lockLayout</td>
<td>FPHTMLWebPartZoneElement, IHTMLWebPartZoneElement</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>namespace</td>
<td>FHTMLWebPartElement, FHTMLXSLWebPartElement, IHTMLWebPartElement</td>
</tr>
<tr>
<td>ncssBdo</td>
<td>IFPStyleState</td>
</tr>
<tr>
<td>orientation</td>
<td>FHTMLWebPartZoneElement, IHTMLWebPartZoneElement</td>
</tr>
<tr>
<td>QueryContents</td>
<td>SearchInfo</td>
</tr>
<tr>
<td>ReplaceWith</td>
<td>SearchInfo</td>
</tr>
<tr>
<td>SkipOnQuery</td>
<td>DynamicTemplateState</td>
</tr>
<tr>
<td>templateRegions</td>
<td>FHTMLDocument, IFPDocument</td>
</tr>
<tr>
<td>webParts</td>
<td>FHTMLDocument, IFPDocument</td>
</tr>
<tr>
<td>webPartZones</td>
<td>FHTMLDocument, IFPDocument</td>
</tr>
<tr>
<td>zoneId</td>
<td>FHTMLWebPartZoneElement, IHTMLWebPartZoneElement</td>
</tr>
<tr>
<td>zoneTitle</td>
<td>FHTMLWebPartZoneElement, IHTMLWebPartZoneElement</td>
</tr>
</tbody>
</table>
New Page Object Model Properties (by Object)

Visit the Office Developer Center on the Microsoft Developer Network Web site for the latest information about programming with Microsoft Office FrontPage 2003, including product news, technical articles, downloads, and samples.

The following table lists properties added to the Office FrontPage 2003 object model (sorted by object name).

<table>
<thead>
<tr>
<th>Object</th>
<th>New Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>DynamicTemplateState</td>
<td>InternalRep, SkipOnQuery</td>
</tr>
<tr>
<td></td>
<td>behaviorElement, layoutTablesAndCells</td>
</tr>
<tr>
<td>FPHTMLDocument</td>
<td>templateRegions, webParts</td>
</tr>
<tr>
<td></td>
<td>webPartZones</td>
</tr>
<tr>
<td>FPHTMLWebPartElement</td>
<td>assembly, namespace</td>
</tr>
<tr>
<td></td>
<td>allowCustomization, allowPersonalization</td>
</tr>
<tr>
<td>FPHTMLWebPartZoneElement</td>
<td>containerWidth, frameType, lockLayout</td>
</tr>
<tr>
<td></td>
<td>orientation, webParts, zoneId, zoneTitle</td>
</tr>
<tr>
<td>FPHTMLXSLWebPartElement</td>
<td>assembly, namespace</td>
</tr>
<tr>
<td>FPLayoutRegion</td>
<td>cellFormattingID, isCell, isTable</td>
</tr>
<tr>
<td></td>
<td>behaviorElement, dynamicTemplate,</td>
</tr>
<tr>
<td>IFPDocument</td>
<td>layoutTablesAndCells, templateRegions</td>
</tr>
<tr>
<td></td>
<td>webParts, webPartZones</td>
</tr>
<tr>
<td>IFPLayoutRegion</td>
<td>cellFormattingID, isCell, isTable</td>
</tr>
<tr>
<td>IFPStyleState</td>
<td>ncssBdo</td>
</tr>
<tr>
<td>IHTMLWebPartElement</td>
<td>assembly, namespace</td>
</tr>
<tr>
<td></td>
<td>allowCustomization, allowPersonalization</td>
</tr>
<tr>
<td>IHTMLWebPartZoneElement</td>
<td>containerWidth, frameType, lockLayout</td>
</tr>
<tr>
<td></td>
<td>orientation, webParts, zoneId, zoneTitle</td>
</tr>
<tr>
<td>SearchInfo</td>
<td>Find, QueryContents, ReplaceWith</td>
</tr>
</tbody>
</table>
New Page Object Model Methods (Alphabetical List)

Visit the Office Developer Center on the Microsoft Developer Network Web site for the latest information about programming with Microsoft Office FrontPage 2003, including product news, technical articles, downloads, and samples.

The following table lists methods added to the Office FrontPage 2003 object model (sorted alphabetically).

<table>
<thead>
<tr>
<th>New Method</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>addScript</code></td>
<td>FPHTMLDocument, IFPDocument</td>
</tr>
<tr>
<td><code>applyDynamicTemplate</code></td>
<td>FPHTMLDocument, IFPDocument</td>
</tr>
<tr>
<td><code>currentDataNode</code></td>
<td>FPHTMLXSLWebPartElement, IHTMLXSLWebPartElement</td>
</tr>
<tr>
<td><code>dataViewNodeSetIterator</code></td>
<td>FPHTMLXSLWebPartElement, IHTMLXSLWebPartElement</td>
</tr>
<tr>
<td><code>dataViewTemplateBodyExpansion</code></td>
<td>FPHTMLXSLWebPartElement, IHTMLXSLWebPartElement</td>
</tr>
<tr>
<td><code>Find</code></td>
<td>FPHTMLDocument, IFPDocument</td>
</tr>
<tr>
<td><code>findByID</code></td>
<td>FPLayoutTablesAndCells</td>
</tr>
<tr>
<td><code>getBorderColor</code></td>
<td>FPLayoutRegion</td>
</tr>
<tr>
<td><code>getBorderSize</code></td>
<td>FPLayoutRegion</td>
</tr>
<tr>
<td><code>getContentColor</code></td>
<td>FPLayoutRegion</td>
</tr>
<tr>
<td><code>getContentHtml</code></td>
<td>FPLayoutRegion</td>
</tr>
<tr>
<td><code>getContentPadding</code></td>
<td>FPLayoutRegion</td>
</tr>
<tr>
<td><code>getContentType</code></td>
<td>FPLayoutRegion</td>
</tr>
<tr>
<td><code>getContentVAlign</code></td>
<td>FPLayoutRegion</td>
</tr>
<tr>
<td><code>getCorner</code></td>
<td>FPLayoutRegion</td>
</tr>
<tr>
<td><code>getCornerBorderColor</code></td>
<td>FPLayoutRegion</td>
</tr>
<tr>
<td><code>getCornerColor</code></td>
<td>FPLayoutRegion</td>
</tr>
</tbody>
</table>
getCornerHeight  FPLayoutRegion
getCornerImageUrl  FPLayoutRegion
getCornerWidth  FPLayoutRegion
getFooterColor  FPLayoutRegion
getFooterHtml  FPLayoutRegion
getFooterPadding  FPLayoutRegion
getFooterSize  FPLayoutRegion
getFooterVAlign  FPLayoutRegion
getHeaderColor  FPLayoutRegion
getHeaderHtml  FPLayoutRegion
getHeaderPadding  FPLayoutRegion
getHeaderSize  FPLayoutRegion
getHeaderVAlign  FPLayoutRegion
getHeight  FPLayoutRegion
getMargin  FPLayoutRegion

getPartProperty  FPHTMLWebPartElement,  IHTMLWebPartElement
getScript  FPHTMLDocument, IFPDocument
getShadowColor  FPLayoutRegion
getShadowPosition  FPLayoutRegion
getShadowSize  FPLayoutRegion
getShadowSmooth  FPLayoutRegion
getWidth  FPLayoutRegion
InsertInteractiveButton  FPHTMLDocument, IFPDocument
insertLayoutCell  FPLayoutTablesAndCells
insertLayoutTable  FPLayoutTablesAndCells
layoutRegion  FPLayoutTablesAndCells
optimizeHTML  FPHTMLDocument, IFPDocument

parentNodeSetIterator  FPHTMLXSLElement,  IHTMLXSLElement

parentTemplateBodyExpansion  FPHTMLXSLElement,  IHTMLXSLElement
parentWebPart
parseCodeChanges
reapplyScript
regenerateDataView
removePartProperty
removeScript
save
SetBodyMapping
setBorderColor
setBorderSize
setContentColor
setContentHtml
setContentPadding
setContentType
setContentVAlign
setCorner
setCornerBorderColor
setCornerColor
setCornerHeight
setCornerImageUrl
setCornerWidth
setFooterColor
setFooterHtml
setFooterPadding
setFooterSize
setFooterVAlign
setHeaderColor
setHeaderHtml
setHeaderPadding
setHeaderSize
<table>
<thead>
<tr>
<th>Method</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>setHeaderVAlign</td>
<td>FPLayoutRegion</td>
</tr>
<tr>
<td>SetHeadMapping</td>
<td>DynamicTemplateState</td>
</tr>
<tr>
<td>setHeight</td>
<td>FPLayoutRegion</td>
</tr>
<tr>
<td>setMargin</td>
<td>FPLayoutRegion</td>
</tr>
<tr>
<td>setPartProperty</td>
<td>FPLayoutRegion, FPHTMLWebPartElement, IHTMLWebPartElement</td>
</tr>
<tr>
<td>setShadowColor</td>
<td>FPLayoutRegion</td>
</tr>
<tr>
<td>setShadowPosition</td>
<td>FPLayoutRegion</td>
</tr>
<tr>
<td>setShadowSize</td>
<td>FPLayoutRegion</td>
</tr>
<tr>
<td>setShadowSmooth</td>
<td>FPLayoutRegion</td>
</tr>
<tr>
<td>setWidth</td>
<td>FPLayoutRegion</td>
</tr>
<tr>
<td>updateDynamicTemplate</td>
<td>FPHTMLDocument, IFPDocument</td>
</tr>
<tr>
<td>writeFile</td>
<td>FPHTMLDocument, IFPDocument</td>
</tr>
<tr>
<td>xmlTree</td>
<td>FPHTMLXSLWebPartElement, IHTMLXSLWebPartElement</td>
</tr>
<tr>
<td>xslInstruction</td>
<td>FPHTMLXSLWebPartElement, IHTMLXSLWebPartElement</td>
</tr>
<tr>
<td>xslNode</td>
<td>FPHTMLXSLElement, IHTMLXSLElement</td>
</tr>
<tr>
<td>xslNodeSetIterator</td>
<td>FPHTMLXSLWebPartElement, IHTMLXSLWebPartElement</td>
</tr>
<tr>
<td>xslTemplateBody</td>
<td>FPHTMLXSLWebPartElement, IHTMLXSLWebPartElement</td>
</tr>
<tr>
<td>xslTree</td>
<td>FPHTMLXSLWebPartElement, IHTMLXSLWebPartElement</td>
</tr>
</tbody>
</table>
New Page Object Model Methods (by Object)

Visit the Office Developer Center on the Microsoft Developer Network Web site for the latest information about programming with Microsoft Office FrontPage 2003, including product news, technical articles, downloads, and samples.

The following table lists methods added to the Office FrontPage 2003 object model (sorted by object name).

<table>
<thead>
<tr>
<th>Object</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>DynamicTemplateState</td>
<td>SetBodyMapping, SetHeadMapping, addScript, applyDynamicTemplate, Find,</td>
</tr>
<tr>
<td></td>
<td>getScript, InsertInteractiveButton, optimizeHTML, parseCodeChanges,</td>
</tr>
<tr>
<td></td>
<td>reapplyScript, removeScript, save, updateDynamicTemplate, writeFile</td>
</tr>
<tr>
<td>FPHTMLDocument</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getPartProperty, removePartProperty, setPartProperty</td>
</tr>
<tr>
<td>FPHTMLWebPartElement</td>
<td></td>
</tr>
<tr>
<td></td>
<td>parentNodeSetIterator, parentTemplateBodyExpansion, parentWebPart,</td>
</tr>
<tr>
<td></td>
<td>xslNode, currentDataNode, DataViewNodeSetIterator, DataViewTemplateBodyExpansion,</td>
</tr>
<tr>
<td>FPHTMLXSLElement</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getPartProperty, regenerateDataView, removePartProperty, setPartProperty,</td>
</tr>
<tr>
<td></td>
<td>xmlTree, xslInstruction, xslNodeSetIterator, xslTemplateBody, xslTree</td>
</tr>
<tr>
<td>FPHTMLXSLWebPartElement</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBorderColor, getBorderSize, getContentType, getContentColor,</td>
</tr>
<tr>
<td></td>
<td>getContentHtml, getContentPadding, getContentType, getContentVAlign,</td>
</tr>
<tr>
<td></td>
<td>getCorner,</td>
</tr>
</tbody>
</table>
FPLayoutRegion

FPLayoutTablesAndCells
findByID, insertLayoutCell, insertLayoutTable, layoutRegion
addScript, applyDynamicTemplate, Find, getScript, InsertInteractiveButton

IFPDocument
optimizeHTML, parseCodeChanges, reapplyScript, removeScript, save, updateDynamicTemplate, writeFile

IHTMLWebPartElement
getPartProperty, removePartProperty, setPartProperty

IHTMLXSLElement
parentNodeSetIterator, parentTemplateBodyExpansion, parentWebPart, xslNode

currentDataNode,
IHTMLXSLWebPartElement
dataViewNodeSetIterator, dataViewTemplateBodyExpansion, getPartProperty, regenerateDataView, xmlTree, xslInstruction, xslNodeSetIterator, xslTemplateBody, xslTree
FPHTMLFiltersCollection Collection

The **FPHTMLFiltersCollection** collection is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
FPHTMLStyleSheetsCollection Object

FPHTMLStyleSheetsCollection

Represents a collection of cascading style sheets attached to an HTML document. See also the IHTMLStyleSheetsCollection object.
Using the FPHTMLStyleSheetsCollection object

Use the `styleSheets` property to return an `FPHTMLStyleSheetsCollection` collection that represents a collection of all the cascading style sheets attached to an HTML document. Use the `Item` method to return an individual `FPHTMLStyleElement` object that accesses a specific cascading style sheet, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first cascading style sheet in the active document.

```vba
Dim objStyleSheet As FPHTMLStyleSheetsCollection
Set objStyleSheet = ActiveDocument.styleSheets
```
FPLayoutTablesAndCells Collection

FPLayoutTablesAndCells < IFPLLayoutRegion

Represents a collection of FPLayoutRegion objects.
Using the FPLayoutTablesAndCells Collection

Use the layoutTablesAndCells property to return a collection of all the FPLayoutRegion objects in a document.
IFPLayoutTablesAndCells Collection

Represents a collection of IFPLayoutRegion objects. The IFPLayoutTablesAndCells collection provides access to a limited number of properties and methods related to the ELE element. For access to all properties and methods, use the FPLayoutTablesAndCells object.
Using the IFPLayoutTablesAndCells Collection

Use the layoutTablesAndCells property to return a collection of all the IFPLayoutRegion objects in a document.
IHTMLAreasCollection Collection

Multiple objects IHTMLAreasCollection

Represents a collection of AREA elements for a specified MAP element in an HTML document. The IHTMLAreasCollection collection contains individual FPHTMLAreaElement or IHTMLAreaElement objects.

See also the FPHTMLMapElement and IHTMLMapElement objects.
Using the IHTMLAreasCollection Collection

Use the **areas** property to return a collection of AREA elements for a MAP element. The following example accesses the AREA elements for the first MAP element in the active document.

```
Dim objMap As FPHTMLMapElement
Dim objAreas As IHTMLAreasCollection

Set objMap = ActiveDocument.all.Item("map").Item(0)
Set objAreas = objMap.areas
```
IHTMLElementCollection Collection

Multiple objects `IHTMLElementCollection`

Represents a collection of elements in an HTML document.
Using the IHTMLElementCollection collection

Use the `all` property of the `FPHTMLDocument` or `IHTMLDocument2` object to return a collection of all elements in a document, including elements in the HEAD element. The following example accesses all elements in the active document.

```vbnet
Dim objElements As IHTMLElementCollection
Set objElements = ActiveDocument.all
```

Use the `all` property of the `FPHTMLBody` or `IHTMLBodyElement` object to return a collection of all elements in the BODY element of a document. The following example accesses all elements in the body of active document.

```vbnet
Dim objElements As IHTMLElementCollection
Set objElements = ActiveDocument.body.all
```

Use the `tags` method to return a collection of a specific elements. The following example accesses all P elements in the active document.

```vbnet
Dim objParagraphs As IHTMLElementCollection
Set objParagraphs = ActiveDocument.body.all.tags("p")
```
**IHTMLFramesCollection2 Object**

Multiple objects **IHTMLFramesCollection2**

Represents a collection of FRAME elements in an HTML document.
Using the IHTMLFramesCollection2 Object

Use the frames property to return an IHTMLFramesCollection2 object. The following example accesses the collection of FRAME elements in the active document and then uses the Item method to access the first FRAME element in the collection.

Dim objFrames As IHTMLFramesCollection2
Dim objFrame As FPHTMLFrameElement

Set objFrames = ActiveDocument.frames
Set objFrame = objFrames.Item(0)
IHTMLStyleSheetRulesCollection

Collection

Multiple objects

IHTMLStyleSheetRulesCollection

IHTMLStyleSheetRule

IHTMLRuleStyle

Represents a collection of cascading style sheet (CSS) rules.
Using the IHTMLStyleSheetRulesCollection object

Use the rules property to return an IHTMLStyleSheetRulesCollection collection that represents a collection of all the cascading style sheet rules in an attached CSS. Use the Item method to return an IHTMLStyleSheetRule object that accesses a specific CSS rule, referenced by ordinal number. The following example accesses the collection of CSS rules for the first CSS attached to the active document.

Dim objCSS As FPHTMLStyleSheet
Dim objRules As IHTMLStyleSheetRulesCollection

Set objCSS = ActiveDocument.styleSheets(0)
Set objRules = objCSS.rules
IHTMLStyleSheetsCollection Object

Multiple objects IHTMLStyleSheetsCollection

Represents a collection of cascading style sheets attached to an HTML document. The IHTMLStyleSheetsCollection object provides access to a limited number of properties and methods related to a collection of cascading style sheets. For access to all properties and methods, use the FPHTMLStyleSheetsCollection object.
Using the IHTMLStyleSheetsCollection object

Use the `styleSheets` property to return an `IHTMLStyleSheetsCollection` collection that represents a collection of all the cascading style sheets attached to an HTML document. Use the `Item` method to return an individual `FPHTMLStyleElement` object that accesses a specific cascading style sheet, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first cascading style sheet in the active document.

```vbnet
Dim objStyleSheet As IHTMLStyleSheetsCollection
Set objStyleSheet = ActiveDocument.styleSheets
```
**DynamicTemplateState Object**

**DynamicTemplateState**

Represents an object that stores mapping information related to a Dynamic Web Template.

**Note** The **DynamicTemplateState** object is an in-memory object only and doesn't correspond to any Microsoft FrontPage application element or to an HTML element. Information about an attached Dynamic Web Template and associated regions within a document are stored in COMMENT elements within a document. Use the **templateRegions** property to access the Dynamic Web Template regions in a document.
Using the DynamicTemplateState object

Use the CreateDynamicTemplateState method of the FrontPage Application object to create a new DynamicTemplateState object. The following example creates a new DynamicTemplateState object.

Dim objState As DynamicTemplateState
Set objState = Application.CreateDynamicTemplateState

Use the SetBodyMapping and SetHeadMapping methods to specify how to map regions in a document where the regions in a Dynamic Web Template are different from the existing regions in the document. Then use the ApplyDynamicTemplate method of the WebFile or FPHTMLDocument object to attach a Dynamic Web Template to a document.

The following example creates a new DynamicTemplateState object, specifies region mapping, and then attaches the specified Dynamic Web Template to the specified file.

Dim objState As DynamicTemplateState
Dim objFile As WebFile
Set objState = Application.CreateDynamicTemplateState
Set objFile = ActiveWeb.LocateFile("home.htm")
objState.SetBodyMapping "", "template.dwt", "", "A1"
objState.SetHeadMapping "", "template.dwt", "", "metadata"
objFile.ApplyDynamicTemplate "template.dwt", objState

Use the UpdateDynamicTemplate method to update a page that is attached to a Dynamic Web Template to apply any changes that were made to the Dynamic Web Template file. The following example updates the Dynamic Web Template references in the active document.

Dim objState As DynamicTemplateState
Set objState = Application.CreateDynamicTemplateState
If ActiveDocument.DynamicTemplate <> "" Then _
    ActiveDocument.UpdateDynamicTemplate objState
End Sub
FPHTMLAnchorElement Object

Multiple objects

FPHTMLAnchorElement

Represents a specified bookmark in a page. Bookmarks are represented by A elements that use the name attribute. See also IHTMLAnchorElement object.

Note  A elements that use the href attribute are links. For information about accessing links, see the FPHTMLLinkElement and IHTMLLinkElement objects.
Using the FPHTMLAnchorElement object

Use the anchors property to return an IHTMLElementCollection object that represents all the anchors in a document. Use the Item method to return an FPHTMLAnchorElement object. The following example returns a String array containing the names of all the bookmarks in the specified document.

Function GetBookmarks(objDoc As FPHTMLDocument) As String()
    Dim strTemp() As String
    Dim intCount As Integer

    If objDoc.anchors.Length > 0 Then
        ReDim strTemp(objDoc.anchors.Length - 1)

        For intCount = 0 To objDoc.anchors.Length - 1
            strTemp(intCount) = objDoc.anchors.Item(intCount).Name
        Next

        GetBookmarks = strTemp
    End If
End Function
FPHTMLAreaElement Object

Represents an AREA element. AREA elements are contained within MAP elements in an HTML document. Use the FPHTMLAreaElement object to specify the coordinates and shape of an AREA element as well as other attributes of an AREA element. See also the IHTMLAreaElement object.
Using the FPHTMLAreaElement object

Use the `areas` property of an `FPHTMLMapElement` or `IHTMLMapElement` object to return the `IHTMLAreasCollection` object of a `MAP` element. Use the `Item` method to return an `FPHTMLAreaElement` object. The following example returns a string array containing the values of the `href` property, which is equivalent to a hyperlink, for all the `FPHTMLAreaElement` objects in the specified `FPHTMLMapElement` object.

```vba
Function GetAreaHREF(objMap As FPHTMLMapElement) As String()
    Dim objArea As FPHTMLAreaElement
    Dim strAreas() As String
    Dim intCount As Integer

    ReDim strAreas(objMap.areas.Length - 1)

    For intCount = 0 To objMap.areas.Length - 1
        Set objArea = objMap.areas.Item(intCount)
        strAreas(intCount) = objArea.href
    Next

    GetAreaHREF = strAreas
End Function
```

Use the `href`, `Shape`, and `coords` properties to specify the appearance and behavior of an AREA element. The following example takes arguments that specify the shape, hyperlink behavior, and coordinates of the specified `FPHTMLAreaElement` object.

```vba
Sub SetArea(objArea As FPHTMLAreaElement, iX1 As Integer, _
    iY1 As Integer, iX2 As Integer, iY2 As Integer, _
    strHREF As String, strShape As String)
    Dim strCoords As String

    strCoords = iX1 &"," & iY1 &"," & iX2 &"," & iY2

    With objArea
        .href = strHREF
        .Shape = strShape
        .coords = strCoords
    End With
End Sub
```
FPHTMLBaseElement Object

FPHTMLBaseElement Multiple objects

Represents the BASE element of an HTML document. Use the FPHTMLBaseElement object to specify the base URL for all relative URLs in a Web page. You can also use the FPHTMLBaseElement object to specify the base target window for all hyperlinks that do not specify a target window. See also the IHTMLBaseElement object.
Using the FPHTMLBaseElement object

Use the `Item` method to return an `FPHTMLBaseElement` object. The following example takes an `FPHTMLDocument` object and a `String` that represents the target window all hyperlinks in the document will use unless otherwise specified.

```vba
Function SetBaseTarget(objDoc As FPHTMLDocument, _
    strTarget As String) As FPHTMLBaseElement
    Dim objHead As IHTMLElement
    If objDoc.all.tags("base").Length <= 0 Then
        Set objHead = objDoc.all.tags("head").Item(0)
        objHead.insertAdjacentHTML "beforeend", "<Base id="basetarg"
    Set SetBaseTarget = objHead.all.tags("base").Item("basetarge"
Else
    Set SetBaseTarget = objHead.all.tags("base").Item(0)
End If
    SetBaseTarget.target = strTarget
End Function
```

Use the following example to call the preceding function.

```vba
Sub CallSetBaseTarget()
    Call SetBaseTarget(ActiveDocument, "_blank")
End Sub
```
FPHTMLBaseFontElement Object

FPHTMLBaseFontElement

Multiple objects

Represents the BASEFONT element in an HTML document. Using the FPHTMLBaseFontElement object, you can specify base font characteristics such as name, size, and color. See also the IHTMLBaseFontElement object.
Using the FPHTMLBaseFontElement object

Use the Item method to return an FPHTMLBaseFontElement object. The following example function takes an FPHTMLDocument object, one required String (representing the name of the font), and one optional String (representing the size of the font). The function then changes the base font's face and size attributes and returns an FPHTMLBaseFontElement object that represents the BASEFONT element in the specified document.

Function SetBasefont(objDoc As FPHTMLDocument, strFontFace As String Optional strFontSize As String) As FPHTMLBaseFontElement
Dim objBody As FPHTMLBody
Dim objTemp As FPHTMLBaseFontElement

Set objBody = objDoc.body

If objBody.all.tags("basefont").Length <= 0 Then
    objBody.insertAdjacentHTML "afterbegin", "<Basefont>

    Set objTemp = objBody.all.tags("basefont").Item(0)
Else
    Set objTemp = objBody.all.tags("basefont").Item(0)
End If

With objTemp
    .face = strFontFace
    If Len(strFontSize) > 0 Then .Size = strFontSize
End With

Set SetBasefont = objTemp
End Function

Use the following example to call the preceding function.

Sub CallSetBasefont()
    Call SetBasefont(ActiveDocument, "tahoma", "5")
End Sub
FPHTMLBGsound Object

FPHTMLBGsound Multiple objects

Represents the BGSOUND element in an HTML document. The BGSOUND element plays a sound file in the background when a document is loaded into a browser. Use the FPHTMLBGsound object to specify the location of an audio file and looping characteristics of the background sound. See also the IHTMLBGsound object.
Using the FPHTMLBGsound object

Use the **Item** method to return an **FPHTMLBGsound** object. The following example inserts the BGSOUND element into the specified document and sets the **src** and **loop** attributes according to the values passed into the function.

```vbscript
Function InsertBGSound(objdoc As FPHTMLDocument, strSRC As String, _
Optional intLoops As Integer) As Boolean
    Dim objBGSound As FPHTMLBGsound
    Dim intNumber As Integer
    Dim objHead As IHTMLElement

    On Error GoTo InsertBGSoundError

    intNumber = objdoc.body.all.Length
    Set objHead = objdoc.all.tags("head").Item(0)

    objHead.insertAdjacentHTML "beforeend", _
        "<BGSOUND id=""bgsound" & intNumber & "">"
    Set objBGSound = objdoc.all.tags("bgsound")
        .Item(CVar("bgsound" & intNumber))

    With objBGSound
        .src = strSRC
        If intLoops <> 0 Then
            .loop = intLoops
        Else
            .loop = "infinite"
        End If
    End With

    InsertBGSound = True

ExitFunction:
    Exit Function

InsertBGSoundError:
    InsertBGSound = False
    GoTo ExitFunction
End Function
```

Use the following example to call the preceding function. This example assumes that you have a sound file called "song.avi" in your Web site in a subfolder called "Sounds." To see the BGSOUND element as it would function in a
browser, replace the String ".../sounds/song.avi" with the path and filename of a valid sound file in your Web site.

Sub CallInsertBGSound()
    MsgBox InsertBGSound(ActiveDocument, "../sounds/song.avi", 5)
End Sub
FPHTMLBlockElement Object

Represented by the BLOCKQUOTE element in an HTML document. See also the IHTMLBlockElement object.
Using the FPHTMLBlockElement object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents all the BLOCKQUOTE elements in a document. Use the **Item** method to return an **FPHTMLBlockElement** object that accesses a specific BLOCKQUOTE element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first BLOCKQUOTE element in the active document.

Dim objBlock As FPHTMLBlockElement

Set objBlock = ActiveDocument.all.tags("blockquote").Item(0)
**FPHTMLBody Object**

FPHTMLBody \(^{L}\) Multiple objects

Represents the BODY element in an HTML document. See also the IHTMLBodyElement object.
Using the FPHTMLBody object

Use the **Body** property to return an **FPHTMLBody** object. The following example creates an **FPHTMLBody** object that represents the BODY element in the active document.

```vbscript
Dim objBody As FPHTMLBody
Set objBody = ActiveDocument.body
```
FPHTMLBRElement Object

Multiple objects

Represents the BR element in an HTML document. See also the IHTMLBRElement object.
Using the FPHTMLBRElement object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents all the BR elements in a document. Use the **Item** method to return an **FPHTMLBRElement** object that accesses a specific BR element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first BR element in the active document.

```vba
Dim objBreak As FPHTMLBRElement
Set objBreak = ActiveDocument.all.tags("br").Item(0)
```
FPHTMLButtonElement Object

FPHTMLButtonElement

Multiple objects

Represents the BUTTON element in an HTML document. See also the IHTMLButtonElement object.
Using the FPHTMLButtonElement object

Use the tags method to return an IHTMLElementCollection collection that represents all the BUTTON elements in a document. Use the Item method to return an FPHTMLButtonElement object that accesses a specific BUTTON element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first BUTTON element in the active document.

Dim objButton As FPHTMLButtonElement

Set objButton = ActiveDocument.all.tags("button").Item(0)
FPHTMLCMimeTypes Object

The **FPHTMLCMimeTypes** object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
FPHTMLElement Object

The FPHTMLElement object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
FPHTMLCOpsProfile Object

The FPHTMLCOpsProfile object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
FPHTMLCPlugins Object

The **FPHTMLCPlugins** object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
FPHTMLDDElement Object

FPHTMLDDElement - Multiple objects

Represents a DD element in an HTML document. See also the IHTMLDDElement object.
Using the FPHTMLDDElement Object

Use the `tags` method to return an `IHTMLElementCollection` collection that represents all the DD elements in a document. Use the `Item` method to return an `FPHTMLDDElement` object that accesses a specific DD element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first DD element in the active document.

```
Dim objDescr As FPHTMLDDElement

Set objDescr = ActiveDocument.all.tags("dd").Item(0)
```
**FPHTMLDialog Object**

The **FPHTMLDialog** object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
FPHTMLDivElement Object

Multiple objects

Represents a DIV element in an HTML document. See also the IHTMLDivElement object.
Using the FPHTMLDivElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents all the DIV elements in a document. Use the **Item** method to return an **FPHTMLDivElement** object that accesses a specific DIV element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first DIV element in the active document.

```vba
Dim objDiv As FPHTMLDivElement
Set objDiv = ActiveDocument.all.tags("div").Item(0)
```
FPHTMLDivPosition Object

The **FPHTMLDivPosition** object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
FPHTMLDListElement Object

FPHTMLDListElement

Multiple objects

Represents a DL element in an HTML document. See also the IHTMLDListElement object.
Using the FPHTMLDListElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents all the DL elements in a document. Use the **Item** method to return an **FPHTMLDListElement** object that accesses a specific DL element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first DL element in the active document.

```vba
Dim objDef As FPHTMLDListElement
Set objDef = ActiveDocument.all.tags("dl").Item(0)
```
FPHTMLDocument Object

FPHTMLDocument

Multiple objects

Represents the active HTML document. The FPHTMLDocument object contains all property information related to the HTML in a Web page such as background color, link color, and style sheet as well as methods that act on the data such as the **CreateElement** and **ExecCommand** methods.
Using the FPHTMLDocument object

Use the **ActiveDocument** property of the **Application** object or the **Document** property of the **PageWindowEx** object to return an **FPHTMLDocument** object. The following example modifies the background color of the current document.

```vba
Sub Document()
    Dim objDoc As FPHTMLDocument
    Dim objWindow As PageWindowEx

    'Create a reference to the active page window.
    Set objWindow = ActivePageWindow

    'Create a reference to the currently open document.
    Set objDoc = objWindow.Document

    objDoc.bgColor = "yellow"
End Sub
```

**Note**  There is no way to access the contents of a Web page using Microsoft Visual Basic for Applications in Microsoft FrontPage without first opening the document and setting it as the active document using either the **ActivePageWindow** property, as shown in the preceding example, or the **ActiveDocument** property.
Remarks

The PageWindowEx object, in the FrontPage Web Object Model, contains both an ActiveDocument property and a Document property. Both properties return an FPHTMLDocument object. However, when a FRAMESET element is contained within a page, the Document property returns the page containing the FRAMESET element, and the ActiveDocument property returns the page referenced in the SRC attribute of the FRAME element where the insertion point currently resides.
**FPHTMLDTElement Object**

Multiple objects

Represents a DT element in an HTML document. See also the **IHTMLDTElement** object.
Using the FPHTMLDTElement Object

Use the tags method to return an IHTMLElementCollection collection that represents all the DT elements in a document. Use the Item method to return an FPHTMLDTElement object that accesses a specific DT element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first DT element in the active document.

Dim objTerm As FPHTMLDTElement

Set objTerm = ActiveDocument.all.tags("dt").Item(0)
FPHTMLEmbed Object

FPHTMLEmbed Multiple objects

Represents an EMBED element in an HTML document. See also the IHTMLEmbedElement object.
Using the FPHTMLEmbed Object

Use the **embeds** or **plugins** property to return an **IHTMLElementCollection** collection that represents a collection of all the EMBED elements in a document. Use the **Item** method to return an **FPHTMLEmbed** object that accesses a specific EMBED element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first EMBED element in the active document.

```
Dim objEmbed As FPHTMLEmbed
Set objEmbed = ActiveDocument.embeds.Item(0)
```
FPHTMLFieldSetElement Object

FPHTMLFieldSetElement Multiple objects

Represents a FIELDSET element in an HTML document. See also the IHTMLFieldSetElement object.
Using the FPHTMLFieldSetElement Object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the FIELDSET elements in a document. Use the Item method to return an FPHTMLFieldSetElement object that accesses a specific FIELDSET element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first FIELDSET element in the active document.

Dim objField As FPHTMLFieldSetElement
Set objField = ActiveDocument.all.tags("fieldset").Item(0)
FPHTMLFontElement Object

FPHTMLFontElement Multiple objects

Represents a FONT element in an HTML document. See also the IHTMLFontElement object.
Using the FPHTMLFontElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all FONT elements in a document. Use the **Item** method to return an **FPHTMLFontElement** object that accesses a specific FONT element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first FONT element in the active document.

```vbnet
Dim objFont As FPHTMLFontElement
Set objFont = ActiveDocument.all.tags("font").Item(0)
```
FPHTMLFormElement Object

FPHTMLFormElement  Multiple objects

Represents a FORM element in an HTML document. See also the IHTMLFormElement object.
Using the FPHTMLFormElement Object

Use the **forms** property to return an **IHTMLElementCollection** collection that represents a collection of all the FORM elements in a document. Use the **Item** method to return an **FPHTMLFormElement** object that accesses a specific FORM element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first FORM element in the active document.

```vbscript
Dim objForm As FPHTMLFormElement
Set objForm = ActiveDocument.forms.Item(0)
```
FPHTMLFrameBase Object

The FPHTMLFrameBase object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
FPHTMLFrameElement Object

FPHTMLFrameElement Multiple objects

 Represents a FRAME element in an HTML document. See also the IHTMLFrameElement object.
Using the FPHTMLFrameElement Object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all FRAME elements in a document. Use the Item method to return an FPHTMLFrameElement object that accesses a specific FRAME element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first FRAME element in the active document.

Dim objFrame As FPHTMLFrameElement

Set objFrame = ActiveDocument.all.tags("frame").Item(0)
FPHTMLFrameSetSite Object

FPHTMLFrameSetSite (Multiple objects)

Represents a FRAMESET element in an HTML document. See also the IHTMLFrameSetElement object.
Using the FPHTMLFrameSetSite Object

Use the `tags` method to return an `IHTMLElementCollection` collection that represents the collection of FRAMESET elements in a document. Use the `Item` method to access an `FPHTMLFrameSetSite` object that accesses a specific FRAMESET element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first FRAMESET element in the active document.

```vba
Dim objFrameSet As FPHTMLFrameSetSite
Set objFrameSet = ActiveDocument.all.tags("frameset").Item(0)
```
FPHTMLFrontPageBotElement Object

Multiple objects

FPHTMLFrontPageBotElement

Represents a Microsoft FrontPage web component. See also the IHTMLFrontPageBotElement and IHTMLFrontPageBotElement2 objects.
Using the FPHTMLFrontPageBotElement object

Use the `tags` method to return an `IHTMLElementCollection` collection that represents a collection of all the FrontPage components in a document. Use the `Item` method to access an `FPHTMLFrontPageBotElement` object that accesses a specific FrontPage component, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first FrontPage component in the active document.

```vbnet
Dim objWebbot As FPHTMLFrontPageBotElement
Set objWebbot = ActiveDocument.all.tags("webbot").Item(0)
```
FPHTMLHeaderElement Object

FPHTMLHeaderElement  Multiple objects

Represents headings in an HTML document. Headings are represented using the H1 through H6 elements. See also the IHTMLHeaderElement.
Using the FPHTMLHeaderElement Object

Use the `tags` method to return an `IHTMLElementCollection` collection that represents a collection of all the headings of a specified type in a document. For example, use `tags("h1")` to return a collection of all H1 elements in a document and use `tags("h2")` to return a collection of all H2 elements in a document.

Use the `Item` method to return an `FPHTMLHeaderElement` object that accesses a specific heading element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first H1 element in the active document.

```vba
Dim objHeading As FPHTMLHeaderElement

Set objHeading = ActiveDocument.all.tags("h1").Item(0)
```
FPHTMLHistory Object

The **FPHTMLHistory** object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
FPHTMLHRElement Object

FPHTMLHRElement Multiple objects

Represents an HR element in an HTML document. See also the IHTMLHRElement object.
Using the FPHTMLHRElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all the HR elements in a document. Use the **Item** method to return an **FPHTMLHRElement** object that accesses a specific HR element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first HR element in the active document.

```vbnet
Dim objLine As FPHTMLHRElement
Set objLine = ActiveDocument.all.tags("hr").Item(0)
```
FPHTMLIFrame Object

FPHTMLIFrame Multiple objects

Represents an IFRAME element in an HTML document. See also the IHTMLIFrameElement object.
Using the FPHTMLIFrame Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all the IFRAME elements in a document. Use the **Item** method to return an **FPHTMLIFrame** object that accesses a specific IFRAME element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first IFRAME element in the active document.

```vbscript
Dim objIFrame As FPHTMLIFrame
Set objIFrame = ActiveDocument.all.tags("iframe").Item(0)
```
FPHTMLImageElementFactory Object

The FPHTMLImageElementFactory object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
FPHTMLImg Object

FPHTMLImg Multiple objects

Represents an individual IMG element in an HTML document.
Using the FPHTMLImg Object

Use the images property to return an IHTMLElementCollection collection that represents a collection of all the IMG elements in a document. Use the Item method to return an FPHTMLImg object that accesses a specific IMG element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first IMG element in the active document.

Dim objImage As FPHTMLImg
Set objImage = ActiveDocument.images.Item(0)
FPHTMLInputElement Object

FPHTMLInputElement

Multiple objects

Represents an INPUT element of type "button" in a HTML document. See also the IHTMLInputElement object.
Using the FPHTMLInputButtonElement Object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the INPUT elements in a document. Use the Item method to return an FPHTMLInputButtonElement object that accesses a specific INPUT element, referenced by ordinal number or by the value of the id attribute. Use the type property to determine the type of INPUT element.

**Note** The type property is not a member of the IHTMLElement object; however, it will return valid results for the value of the type attribute for an element, which in this case is "button".

The following example places the first INPUT element in the active document into an IHTMLElement object variable, then uses the type property to determine the type of INPUT element, and places the variable into an appropriate object variable, providing access to the properties and methods that relate to INPUT elements of the specified type.

```
Dim objInput As IHTMLElement
Dim objButton As FPHTMLInputButtonElement
Dim objFile As FPHTMLInputFileElement
Dim objHidden As FPHTMLInputHiddenElement
Dim objImage As FPHTMLInputImage
Dim objText As FPHTMLInputTextElement

Set objInput = ActiveDocument.all.tags("input").Item(0)

Select Case objInput.Type
    Case "button"
        Set objButton = objInput
    Case "file"
        Set objFile = objInput
    Case "hidden"
        Set objHidden = objInput
    Case "image"
        Set objImage = objInput
    Case "text"
        Set objText = objInput
End Select
```
FPHTMLInputElement Object

FPHTMLInputElement

Multiple objects

Represents an INPUT element of type "file" in an HTML document. See also the IHTMLInputElement object.
Using the FPHTMLInputElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all the INPUT elements in a document. Use the **Item** method to return an **FPHTMLInputElement** object that accesses a specific INPUT element, referenced by ordinal number or by the value of the **id** attribute. Use the **type** property to determine the type of INPUT element.

**Note** The **type** property is not a member of the **IHTMLElement** object; however, it will return valid results for the value of the **type** attribute for an element, which in this case is "file".

The following example places the first INPUT element in the active document into an **IHTMLElement** object variable, then uses the **type** property to determine the type of INPUT element, and places the variable into an appropriate object variable, providing access to the properties and methods that relate to INPUT elements of the specified type.

```vba
Dim objInput As IHTMLElement
Dim objButton As FPHTMLInputButtonElement
Dim objFile As FPHTMLInputFileElement
Dim objHidden As FPHTMLInputHiddenElement
Dim objImage As FPHTMLInputImage
Dim objText As FPHTMLInputTextElement

Set objInput = ActiveDocument.all.tags("input").Item(0)

Select Case objInput.Type
    Case "button"
        Set objButton = objInput
    Case "file"
        Set objFile = objInput
    Case "hidden"
        Set objHidden = objInput
    Case "image"
        Set objImage = objInput
    Case "text"
        Set objText = objInput
End Select
```
FPHTMLInputElement Multiple objects

Represents an INPUT element of type "hidden" in an HTML document. See also the IHTMLInputElement object.
Using the FPHTMLInputHiddenElement Object

Use the **tags** method to return an **IHTMLInputElementCollection** collection that represents a collection of all the INPUT elements in a document. Use the **Item** method to return an **FPHTMLInputHiddenElement** object that accesses a specific INPUT element, referenced by ordinal number or by the value of the **id** attribute. Use the **type** property to determine the type of INPUT element.

**Note** The **type** property is not a member of the **IHTMLElement** object; however, it will return valid results for the value of the **type** attribute for an element, which in this case is "hidden".

The following example places the first INPUT element in the active document into an **IHTMLElement** object variable, then uses the **type** property to determine the type of INPUT element, and places the variable into an appropriate object variable, providing access to the properties and methods that relate to INPUT elements of the specified type.

```plaintext
Dim objInput As IHTMLElement
Dim objButton As FPHTMLInputButtonElement
Dim objFile As FPHTMLInputFileElement
Dim objHidden As FPHTMLInputHiddenElement
Dim objImage As FPHTMLInputImage
Dim objText As FPHTMLInputTextElement

Set objInput = ActiveDocument.all.tags("input").Item(0)

Select Case objInput.Type
    Case "button"
        Set objButton = objInput
    Case "file"
        Set objFile = objInput
    Case "hidden"
        Set objHidden = objInput
    Case "image"
        Set objImage = objInput
    Case "text"
        Set objText = objInput
End Select
```
FPHTMLInputImage Object

FPHTMLInputImage represents an INPUT element of type "image" in a HTML document. See also the IHTMLInputImage object.
Using the FPHTMLInputElement Object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the INPUT elements in a document. Use the Item method to return a FPHTMLInputElement object that accesses a specific INPUT element, referenced by ordinal number or by the value of the id attribute. Use the type property to determine the type of INPUT element.

Note The type property is not a member of the IHTMLElement object; however, it will return valid results for the value of the type attribute for an element, which in this case is "image".

The following example places the first INPUT element in the active document into an IHTMLElement object variable, then uses the type property to determine the type of INPUT element, and places the variable into an appropriate object variable, providing access to the properties and methods that relate to INPUT elements of the specified type.

```vbnet
Dim objInput As IHTMLElement
Dim objButton As FPHTMLInputElementButtonElement
Dim objFile As FPHTMLInputFileElement
Dim objHidden As FPHTMLInputHiddenElement
Dim objImage As FPHTMLInputImage
Dim objText As FPHTMLInputTextElement

Set objInput = ActiveDocument.all.tags("input").Item(0)

Select Case objInput.Type
    Case "button"
        Set objButton = objInput
    Case "file"
        Set objFile = objInput
    Case "hidden"
        Set objHidden = objInput
    Case "image"
        Set objImage = objInput
    Case "text"
        Set objText = objInput
End Select
```
FPHTMLInputTextElement Object

FPHTMLInputTextElement

Multiple objects

Represents an INPUT element of type "text" in an HTML document. See also the IHTMLInputTextElement object.
Using the FPHTMLInputTextElement Object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the INPUT elements in a document. Use the Item method to return an FPHTMLInputTextElement object that accesses a specific INPUT element, referenced by ordinal number or by the value of the id attribute. Use the type property to determine the type of INPUT element.

Note The type property is not a member of the IHTMLElement object; however, it will return valid results for the value of the type attribute for an element, which in this case is "text".

The following example places the first INPUT element in the active document into an IHTMLElement object variable, then uses the type property to determine the type of INPUT element, and places the variable into an appropriate object variable, providing access to the properties and methods that relate to INPUT elements of the specified type.

```vbscript
Dim objInput As IHTMLElement
Dim objButton As FPHTMLInputButtonElement
Dim objFile As FPHTMLInputFileElement
Dim objHidden As FPHTMLInputHiddenElement
Dim objImage As FPHTMLInputImage
Dim objText As FPHTMLInputTextElement

Set objInput = ActiveDocument.all.tags("input").Item(0)

Select Case objInput.Type
    Case "button"
        Set objButton = objInput
    Case "file"
        Set objFile = objInput
    Case "hidden"
        Set objHidden = objInput
    Case "image"
        Set objImage = objInput
    Case "text"
        Set objText = objInput
End Select
```
FPHTMLIsIndexElement Object

Multiple objects

Represents an ISINDEX element in an HTML document. See also the IHTMLIsIndexElement object.
Using the FPHTMLIsIndexElement Object

Use the tags method to return an IHTMLInputElementCollection collection that represents a collection of all the ISINDEX elements in a document. Use the Item method to return an FPHTMLIsIndexElement object that accesses a specific ISINDEX element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first ISINDEX element in the active document.

Dim objIndex As FPHTMLIsIndexElement

Set objIndex = ActiveDocument.all.tags("isindex").Item(0)
FPHTMLLabelElement Object

FPHTMLLabelElement - Multiple objects

Represents a LABEL element in an HTML document. See also the IHTMLLabelElement object.
Using the FPHTMLLabelElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all the LABEL elements in a document. Use the **Item** method to return an **FPHTMLLabelElement** object that accesses a specific LABEL element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first LABEL element in the active document.

Dim objLabel As FPHTMLLabelElement

Set objLabel = ActiveDocument.all.tags("label").Item(0)
FPHTMLLegendElement Object

Multiple objects

FPHTMLLegendElement

Represents a LEGEND element for a FIELDSET element in an HTML document. See also the IHTMLLegendElement object.
Using the FPHTMLLegendElement Object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the LEGEND elements in a document. Use the Item method to return an FPHTMLLegendElement object that accesses a specific LEGEND element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first LEGEND element in the active document.

Dim objLegend As FPHTMLLegendElement

Set objLegend = ActiveDocument.all.tags("legend").Item(0)
**FPHTMLLIElement Object**

*FPHTMLLIElement* — Multiple objects

Represents a LI element in an HTML document. See also the *IHTMLLIElement* object.
Using the FPHTMLLIElement Object

Use the `tags` method to return an `IHTMLInputElementCollection` collection that represents a collection of all the LI elements in a document. Use the `Item` method to return an `FPHTMLLIElement` object that accesses a specific LI element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first LI element in the active document.

```vbnet
Dim objItem As FPHTMLLIElement
Set objItem = ActiveDocument.all.tags("li").Item(0)
```

The following example accesses the first LI element in the first ordered list in the active document.

```vbnet
Dim objList As FPHTMLULListElement
Dim objItem As FPHTMLLIElement

Set objList = ActiveDocument.all.tags("ul").Item(0)
Set objItem = objList.all.tags("li").Item(0)
```
FPHTMLLinkElement Object

FPHTMLLinkElement

Multiple objects

Represents a LINK element in an HTML document. See also the IHTMLLinkElement object.
Using the FPHTMLLinkElement object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the LINK elements in a document. Use the Item method to return an FPHTMLLinkElement object that accesses a specific LINK element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first LINK element in the active document.

Dim objLink As FPHTMLLinkElement
Set objLink = ActiveDocument.all.tags("link").Item(0)

Note The FPHTMLLinkElement object cannot be accessed using the links property. The links property applies to hyperlinks in a document.
FPHTMLListElement Object

The **FPHTMLListElement** object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**FPHTMLLocation Object**

FPHTMLLocation

The **FPHTMLLocation** object contains members that provide information about the current URL.
Using the IHTMLLocation object

Use the `location` property to return an `FPHTMLLocation` object. Use the `href` and `pathname` properties of the `FPHTMLLocation` object to return the path and filename of a specified document. Use the `reload` method to reload the page. The following example reloads the specified document.

**Note** When you use the `reload` method on a saved document that has been changed but not resaved, a message appears asking if the user wants to revert to the previously saved version.

```vba
Sub ReloadDocument()
    Dim objLocation As FPHTMLLocation
    Set objLocation = ActiveDocument.Location
    objLocation.reload
End Sub
```
FPHTMLMapElement Object

FPHTMLMapElement Multiple objects

Represents a MAP element in an HTML document. See also IHTMLMapElement object.
Using the FPHTMLMapElement object

Use the **Item** method to return a specific **FPHTMLMapElement** object. The following example accesses the first MAP element in the active document.

```vbscript
Dim objMap As FPHTMLMapElement
Set objMap = ActiveDocument.all.tags("map").Item(0)
```
FPHTMLMarqueeElement Object

Multiple objects

Represents a MARQUEE element in an HTML document. See also the IHTMLMarqueeElement object.
Using the FPHTMLMarqueeElement Object

Use the `tags` method to return an `IHTMLElementCollection` collection that represents a collection of all the MARQUEE elements in a document. Use the `Item` method to return an `FPHTMLMarqueeElement` object that accesses a specific MARQUEE element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first MARQUEE element in the active document.

```vba
Dim objMarquee As FPHTMLMarqueeElement

Set objMarquee = ActiveDocument.all.tags("marquee").Item(0)
```
FPHTMLMetaElement Object

FPHTMLMetaElement Multiple objects

Represents a META element in an HTML document. See also the IHTMLMetaElement object.
Using the FPHTMLMetaElement Object

Use the `tags` method to return an `IHTMLElementCollection` collection that represents a collection of all the META elements in a document. Use the `Item` method to return an `FPHTMLMetaElement` object that accesses a specific META element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first META element in the active document.

```vba
Dim objMeta As FPHTMLMetaElement
Set objMeta = ActiveDocument.all.tags("meta").Item(0)
```
FPHTMLNavigator Object

FPHTMLNavigator

Multiple objects

Represents the Web browser in Microsoft FrontPage.
Using the FPHTMLNavigator Object

Use the `navigator` property to return an `FPHTMLNavigator` object. The following code accesses the browser for the active document.

```vba
Dim objNav As FPHTMLNavigator
Set objNav = ActiveDocument.parentWindow.navigator
```
FPHTMLNextIdElement Object

The FPHTMLNextIdElement object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
The **FPHTMLNoShowElement** object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
FPHTMLObjectElement Object

FPHTMLObjectElement Represents an OBJECT element in an HTML document. See also the IHTMLObjectElement object.
Using the FPHTMLObjectElement object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the OBJECT elements in a document. Use the Item method to return an FPHTMLObjectElement object that accesses a specific OBJECT element, referenced by ordinal number or by the value of the id attribute.

The following example accesses the first OBJECT element in the active document.

Dim objObject As FPHTMLObjectElement

Set objObject = ActiveDocument.all.tags("object").Item(0)
FPHTMLLOListElement Object

FPHTMLLOListElement

Multiple objects

Represents an OL element in an HTML document. See also the IHTMLLOListElement object.
Using the FPHTMLOLListElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all the OL elements in a document. Use the **Item** method to return an **FPHTMLOLListElement** object that accesses a specific OL element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first OL element in the active document.

Dim objList As FPHTMLOLListElement

Set objList = ActiveDocument.all.tags("ol").Item(0)
FPHTMLOptionButtonElement Object

The FPHTMLOptionButtonElement object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
FPHTMLOptionElement Object

FPHTMLOptionElement

Multiple objects

Represents an OPTION element in an HTML document. See also IHTMLOptionElement object.
Using the FPHTMLOptionElement Object

Use the `options` property to return an `IHTMLElementCollection` collection that represents a collection of all the OPTION elements in a SELECT element. Use the `Item` method to return an `FPHTMLOptionElement` object that accesses a specific OPTION element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first OPTION element of the first SELECT element in the active document.

```vbscript
Dim objSelect As FPHTMLSelectElement
Dim objOption As FPHTMLOptionElement

Set objSelect = ActiveDocument.all.tags("select").Item(0)
Set objOption = objSelect.Options(0)
```
FPHTMLOptionElementFactory Object

The `FPHTMLOptionElementFactory` object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
FPHTMLParaElement Object

Multiple objects

Represents a P element in an HTML document. See also the IHTMLParaElement object.
Using the FPHTMLParaElement Object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the P elements in a document. Use the Item method to return an FPHTMLParaElement object that accesses a specific P element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first P element in the active document.

Dim objPara As FPHTMLParaElement

Set objPara = ActiveDocument.all.tags("p").Item(0)
FPHTMLPhraseElement Object

The **FPHTMLPhraseElement** object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
The **FPHTMLScreen** object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
FPHTMLScriptElement Object

FPHTMLScriptElement

Multiple objects

Represents a SCRIPT element in an HTML document. See also the IHTMLScriptElement object.
Using the `FPHTMLScriptElement` object

Use the `scripts` property to return an `IHTMLElementCollection` collection that represents a collection of all the SCRIPT elements in a document. Use the `Item` method to return an `FPHTMLScriptElement` object that accesses a specific SCRIPT element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first SCRIPT element in the active document.

```vba
Dim objScript As FPHTMLScriptElement
Set objScript = ActiveDocument.Scripts.Item(0)
```
FPHTMLSelectElement Object

FPHTMLSelectElement

Multiple objects

Represents a SELECT element in an HTML document. See also the IHTMLSelectElement object.
Using the FPHTMLSelectElement object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all the SELECT elements in a document. Use the **Item** method to return an **FPHTMLSelectElement** object that accesses a specific SELECT element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first SELECT element in the active document.

Dim objSelect As FPHTMLSelectElement

Set objSelect = ActiveDocument.all.tags("select").Item(0)
FPHTMLSpanElement Object

FPHTMLSpanElement

Multiple objects

Represents a SPAN element in an HTML document. See also the IHTMLSpanElement object.
Using the FPHTMLSpanElement object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the SPAN elements in a document. Use the Item method to return an FPHTMLSpanElement object that accesses a specific SPAN element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first SPAN element in the active document.

Dim objSelect As FPHTMLSpanElement
Set objSelect = ActiveDocument.all.tags("span").Item(0)
FPHTMLSpanFlow Object

The FPHTMLSpanFlow object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
FPHTMLStyle Object

**FPHTMLStyle**

Represents a `style` attribute for the Cascading Style Sheet (CSS) for an HTML element. See also the `IHTMLStyle` object.
Using the FPHTMLStyle object

Use the style property to return an FPHTMLStyle object. The following example accesses the style attribute for the body of the active document.

Dim objStyle As FPHTMLStyle
Set objStyle = ActiveDocument.body.Style
FPHTMLStyleElement Object

FPHTMLStyleElement Multiple objects

Represents a STYLE element in an HTML document. See also the IHTMLStyleElement object.
Using the FPHTMLStyleElement object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the STYLE elements in a document. Use the Item method to return an FPHTMLStyleElement object that accesses a specific STYLE element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first STYLE element in the active document.

Dim objStyle As FPHTMLStyleElement

Set objStyle = ActiveDocument.all.tags("style").Item(0)
FPHTMLStyleSheet Object

FPHTMLStyleSheet

Multiple objects

Represents a cascading style sheet reference in an HTML document. See also the IHTMLStyleSheet object.
Using the FPHTMLStyleSheet object

Use the styleSheets property to return an FPHTMLStyleSheetsCollection collection that represents a collection of all the cascading style sheets referenced in a document. Use the Item method to return an FPHTMLStyleSheet object that accesses a specific cascading style sheet, referenced by ordinal number or by the value of the id attribute. The following example accesses the first cascading style sheet in the active document.

Dim objStyleSheet As FPHTMLStyleSheet

Set objStyleSheet = ActiveDocument.styleSheets.Item(0)
FPHTMLTable Object

FPHTMLTable Multiple objects

 Represents a TABLE element in an HTML document. See also the IHTMLTable object.
Using the FPHTMLTable object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the TABLE elements in a document. Use the Item method to return an FPHTMLTable object that accesses a specific TABLE element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first TABLE element in the active document.

Dim objTable As FPHTMLTable
Set objTable = ActiveDocument.all.tags("table").Item(0)

To change the properties for the TFOOT, THEAD, TR, and TD elements, use the FPHTMLTableSection, FPHTMLTableRow, FPHTMLTableCol, and FPHTMLTableCell objects.
FPHTMLTableCaption Object

FPHTMLTableCaption

Multiple objects

Represents a CAPTION element for a TABLE element in an HTML document. See also the IHTMLTableCaption object.
Using the FPHTMLTableCaption object

Use the `caption` property of the `FPHTMLTable` object to return an `FPHTMLTableCaption` object that represents the CAPTION element for that table. The following example accesses the first CAPTION element in the first TABLE element in the active document.

```vba
Dim objTable As FPHTMLTable
Dim objCaption As FPHTMLTableCaption

Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objCaption = objTable.Caption
```
FPHTMLTableCell Object

Multiple objects

Represents a TD element in an HTML document. See also the IHTMLTableCell object.
Using the FPHTMLTableCell object

Use the **cells** property of the **FPHTMLTableRow** object to return an **IHTMLElementCollection** collection that represents a collection of all the TD elements in a row in a table. Use the **Item** method to return an **FPHTMLTableCell** object that accesses a specific TD element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first TD element in the first row in the first table in the active document.

```vba
Dim objTable As FPHTMLTable
Dim objRow As FPHTMLTableRow
Dim objCell As FPHTMLTableCell

Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objRow = objTable.rows(0)
Set objCell = objRow.cells(0)
```

To return a reference to a specific cell, use the **cellIndex** property of the **FPHTMLTableCell** object. You can add or delete cells using the **insertCell** and **deleteCell** methods. To change or return the contents of a particular cell, use the **innerHTML** or **innerText** property.
FPHTMLTableCol Object

FPHTMLTableCol Multiple objects

Represents a COL element in an HTML document. See also the IHTMLTableCol object.
Using the FPHTMLTableCol object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the COL elements in a TABLE element in a document. Use the Item method to return an FPHTMLTableCol object that accesses a specific COL element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first COL element in the first table in the active document.

```vbnet
Dim objTable As FPHTMLTable
Dim objCol As FPHTMLTableCol

Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objCol = objTable.all.tags("col").Item(0)
```
FPHTMLTableRow Object

FPHTMLTableRow  - Multiple objects

Represents a TR element in an HTML document. See also the IHTMLTableRow object.
Using the FPHTMLTableRow object

Use the `rows` property to return an `IHTMLElementCollection` collection that represents a collection of all the TR elements in a table. Use the `Item` method to return an `FPHTMLTableRow` object that accesses a specific TR element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first TR element in the first table in the active document.

```
Dim objTable As FPHTMLTable
Dim objRow As FPHTMLTableRow

Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objRow = objTable.rows(0)
```

Use the `rowIndex` property of the `FPHTMLTableRow` object to retrieve a reference to a specific row. Use the `insertRow` and `deleteRow` methods to add or remove rows to or from a table.
**FPHTMLTableSection Object**

*FPHTMLTableSection*  
Multiple objects

Represents the THEAD, TFOOT, and TBODY elements inside of a TABLE element in an HTML document. See also the *IHTMLTableSection* object.
Using the FPHTMLTableSection object

Use the **tHead** property of the **FPHTMLTable** object to return an **FPHTMLTableSection** object that represents the THEAD element for a table. The following example accesses the THEAD element for the first table in the active document.

```vba
Dim objTable As FPHTMLTable
Dim objTHead As FPHTMLTableSection
Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objTHead = objTable.tHead
```

Use the **createTHead** method of the **FPHTMLTable** object to add a THEAD element. Use the **deleteTHead** method of the **FPHTMLTable** object to remove a THEAD element.

Use the **tFoot** property of the **FPHTMLTable** object to return an **FPHTMLTableSection** object that represents the TFOOT element for a table. The following example accesses the TFOOT element for the first table in the active document.

```vba
Dim objTable As FPHTMLTable
Dim objTFoot As FPHTMLTableSection
Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objTFoot = objTable.tFoot
```

Use the **createTFoot** method of the **FPHTMLTable** object to add a TFOOT element. Use the **deleteTFoot** method of the **FPHTMLTable** object to remove a TFOOT element.
FPHTMLTemplateRegionElement Object

FPHTMLTemplateRegionElement

Multiple objects

Represents a region in an HTML document that corresponds to a region in a Dynamic Web Template. See also the IHTMLTemplateRegionElement object.
Using the FPHTMLTemplateRegionElement object

Use the `templateRegions` property to return an `IHTMLElementCollection` collection that represents a collection of all the Dynamic Web Template regions in a document. Use the `Item` method to return an individual `FPHTMLTemplateRegionElement` object that accesses a specific Dynamic Web Template region, referenced by ordinal number or by the name of the region. The following example accesses the first Dynamic Web Template region in the active document and sets the `id` attribute equal to the value of the name of the region.

```vba
Dim objRegions As IHTMLElementCollection
Dim objRegion As FPHTMLTemplateRegionElement

Set objRegions = ActiveDocument.templateRegions
Set objRegion = objRegions.Item(0)

objRegion.Id = objRegion.Name
```
FPHTMLTextAreaElement Object

FPHTMLTextAreaElement Multiple objects

Represents a TEXTAREA element in an HTML document. See also the IHTMLTextAreaElement object.
Using the FPHTMLTextareaElement object

Use the `tags` method to return an `IHTMLElementCollection` collection that represents a collection of all the TEXTAREA elements in a document. Use the `Item` method to return an `FPHTMLTextareaElement` object that accesses a specific TEXTAREA element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first TEXTAREA element in the active document.

Dim objTextArea As FPHTMLTextareaElement

Set objTextArea = ActiveDocument.all.tags("textarea").Item(0)
FPHTMLTextElement Object

The FPHTMLTextElement object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
FPHTMLTitleElement Object

FPHTMLTitleElement  Multiple objects

Represents a TITLE element in an HTML document. See also the IHTMLTitleElement object.
Using the FPHTMLTitleElement object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the TITLE elements in a document. Use the Item method to return an FPHTMLTitleElement object that accesses a specific TITLE element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first TITLE element in the active document.

```vba
Dim objTitle As FPHTMLTitleElement
Set objTitle = ActiveDocument.all.tags("title").Item(0)
```

**Note** While there can only be one TITLE element in an HTML document, the tags method by default returns an IHTMLElementCollection. Therefore, you must use the Item method to return a single FPHTMLTitleElement object.

You can set the value of the TITLE element by using the title property.
FPHTMLULListElement Object

FPHTMLULListElement: Multiple objects

Represents a UL element in an HTML document. See also the IHTMLULListElement object.
Using the FPHTMLULListElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all the UL elements in a document. Use the **Item** method to return an **FPHTMLULListElement** object that accesses a specific UL element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first UL element in the active document.

```vba
Dim objList As FPHTMLULListElement
Set objList = ActiveDocument.all.tags("ul").Item(0)
```
FPHTMLUndoTransaction Object

FPHTMLUndoTransaction

Represents the cache where the actions performed by a macro are stored. The FPHTMLUndoTransaction object includes methods that continue or stop the specified transaction.
Using the FPHTMLUndoTransaction Object

Use the createUndoTransaction method of the IFPDocument object to create an FPHTMLUndoTransaction object. The String argument passed in with the createUndoTransaction method is added to the Undo command on the Edit menu when the Commit method is called. The following example creates an FPHTMLUndoTransaction object illustrating the abort and Commit methods.

Private Sub CreateUndoTransaction()
    Dim objDoc As FPHTMLDocument
    Dim objTransaction As FPHTMLUndoTransaction

    On Error GoTo CreateUndoTransactionError

    Set objDoc = ActiveDocument
    Set objTransaction = objDoc . CreateUndoTransaction("Last Macro")

    objDoc.body.insertAdjacentHTML "BeforeEnd", _
    "<b>Added by FP Programmability</b>"

    objTransaction.Commit

    ExitCreateUndoTransaction:
    Exit Sub

CreateUndoTransactionError:
    objTransaction.abort
    GoTo ExitCreateUndoTransaction
End Sub
FPHTMLUnknownElement Object

FPHTMLUnknownElement

Multiple objects

Represents HTML elements for which there are no specific objects. The Hypertext Markup Language (HTML) elements for which there isn't a corresponding object include the following:

ADDRESS  CODE  KBD  SAMP  TT
APPLET   DFN    MENU  SMALL  U
B         DIR    NOBR  STRIKE VAR
BIG       EM     NOEMBED  STRONG
BLINK     HEAD   NOFRAMES  SUB
CENTER    HTML   PARAM  SUP
CITE      I      PRE    TH

See also the IHTMLUnknownElement object.

Note You can also use the IHTMLElement object for unknown elements.
Using the FPHTMLUnknownElement object

When interating through a collection of all elements in a document, elements for which there isn't a corresponding object will return a type name of DispIHTMLUnknownElement. The DispIHTMLUnknownElement object is a hidden object that you cannot access directly. Therefore, you can set these objects to either an FPHTMLUnknownElement or IHTMLUnknownElement object, or you can set them to an IHTMLElement object.
FPHTMLWebPartElement Object

Multiple objects

FPHTMLWebPartElement

Represents a Web part in an HTML document. See also the IHTMLWebPartElement object.

This object is supported only by Web pages or sites that are based on Microsoft Windows SharePoint Services.
Using the FPHTMLWebPartElement Object

Use the `webParts` property of the `FPHTMLDocument` object to return an `IHTMLElementCollection` collection that represents a collection of all the Web Parts within a document. Use the `Item` method to return an `FPHTMLWebPartElement` object that accesses a specific Web Part, referenced by ordinal number. The following example accesses the first Web Part in the active document.

Dim objPart As FPHTMLWebPartElement
Set objPart = ActiveDocument.webParts.Item(0)

Use the `InsertAdjacentHTML` method, as shown in the following example, to insert a Web Part into a document.

Public Sub InsertWebPart()
    Dim strPageDirective As String
    Dim strRegisterDirective As String
    Dim strWebPart As String
    Dim objWebPart As FPHTMLWebPartElement

    strPageDirective = "<%@ Page Language=""C#"" Debug=""true""%>
    strRegisterDirective = "<%@ Register TagPrefix=""WebPartPages"" NameSpace=""Microsoft.SharePoint.WebPartPages"" Assembly=""Microsoft.SharePoint, Version=11.0.0.0, Culture=neutral, PublicKeyToken=71e9bce11e9429c""%>

    strWebPart = "<form runat=""server"">" & vbCrLf & "<WebPartPages:ImageWebPart webpart=""true"" runat=""server"" id=""insertedwebpart"">" & vbCrLf & "</WebPartPages:ImageWebPart>" & vbCrLf & "</form>"
    ActiveDocument.body.insertAdjacentHTML "afterBegin", strWebPart
    Set objWebPart = ActiveDocument.webParts.Item("insertedwebpart")
End Sub
FPHTMLWebPartZoneElement Object

FPHTMLWebPartZoneElement Multiple objects

Represents a Web Part zone in an HTML document. See also the IHTMLWebPartZoneElement object.

This object is supported only by Web pages or sites that are based on Microsoft Windows SharePoint Services.
Using the FPHTMLWebPartZoneElement Object

Use the `webPartZones` property to return an `IHTMLElementCollection` collection that represents a collection of all the Web Part zones in a document. Use the `Item` method to return an `FPHTMLWebPartZoneElement` object that accesses a specific Web Part zone, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first Web Part zone in the active document.

```vba
Dim objZone As FPHTMLWebPartZoneElement
Set objZone = ActiveDocument.webPartZones.Item(0)
```

Use the `InsertAdjacentHTML` method, as shown in the following example, to insert a Web Part Zone into a document.

```vba
Public Sub InsertWebPartZone()
    Dim strWebPartZone As String
    strWebPartZone = "<form runat=""server"">" & vbCrLf & _
        "<WebPartPages:WebPartZone id=""Zone1"" runat=""server"" title=""Zone 1"">" & vbCrLf & _
        "</WebPartPages:WebPartZone>" & vbCrLf & "</form>"
    ActiveDocument.body.insertAdjacentHTML "afterBegin", strWebPartZone
End Sub
```
FPHTMLWindow2 Object

Multiple objects

Represents the window in which an HTML document or a frame within an HTML document is displayed. See also the IHTMLWindow2 object.
Using the FPHTMLWindow2 object

Use the `parentWindow` property of the `FPHTMLDocument` object to return an `FPHTMLWindow2` object. The following example accesses the window in which the active document is displayed.

```vba
Dim objWindow As FPHTMLWindow2
Set objWindow = ActiveDocument.parentWindow
```
FPHTMLWindowProxy Object

FPHTMLWindowProxy

Multiple objects

Represents the browser window or a frame window within a Web page. See also the FPHTMLWindow2 object.
Using the FPHTMLWindowProxy object

Use the parentWindow property of the FPHTMLDocument object or the FrameWindow property of the PageWindowEx object to return an FPHTMLWindowProxy object. The following example accesses the window in which the active document is displayed.

Dim objWindow As FPHTMLWindowProxy

Set objWindow = ActiveDocument.parentWindow
FPHTMLXSLElement Object

FPHTMLXSLElement

Multiple objects

Represents an Extensible Stylesheet Language (XSL) element for an XSL Data View Web Part. See also the IHTMLXSLElement object.

This object is supported only by Web pages or sites that are based on Microsoft Windows SharePoint Services.
Using the FPHTMLXSLElement Object

Use the all property for the FPHTMLXSLWebPartElement object to return an IHTMLElementCollection collection that represents a collection of all the elements for an XSL Data View Web Part. Then use the tags method to return an IHTMLElementCollection collection that represents all of the XSL elements of the specified type (for example, "xsl-template", "xsl-outside-if", "xsl-value-of", etc.) in a Data View Web Part. The following example returns the first xsl:template element in the first Web Part in the active document. This example assumes that the first Web Part in the active document is a Data View Web Part.

Dim objWebPart As FPHTMLXSLWebPartElement
Dim objXSLElement As FPHTMLXSLElement

Set objWebPart = ActiveDocument.WebParts.Item(0)
Set objXSLElement = objWebPart.all.tags("xsl-template").Item(0)
FPHTMLXSLWebPartElement Object

FPHTMLXSLWebPartElement [Multiple objects]

Represents an Extensible Stylesheet Language (XSL) Data View Web Part. See also the IHTMLXSLWebPartElement object.

This object is supported only by Web pages or sites that are based on Microsoft Windows SharePoint Services.
Using the FPHTMLXSLWebPartElement Object

Use the `webParts` property to return an `IHTMLElementCollection` collection that represents a collection of all Web Parts in a document. Use the `Item` method to return an `FPHTMLXSLWebPartElement` object that accesses a specific XSL Data View Web Part, referenced by ordinal number. The following example accesses the first Web Part in the active document. This example assumes that the first Web Part is an XSL Data View Web Part.

```vba
Dim objXSLWebPart As FPHTMLXSLWebPartElement
Set objXSLWebPart = ActiveDocument.webParts.Item(0)
```
FPLayoutRegion Object

FPLayoutRegion  IFPLayoutRegion

Represents a two dimensional region in a document that is specified as containing special table cell formatting, referred to as layout cells, layout tables, and cell formatting tables.
Using the FPLayoutRegion Object

Use the `layoutRegion` method to return a single `FPLayoutRegion` object in a document.
**IDispatchEx Object**

The `IDispatchEx` object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IDTCExtendedServices Object

The `IDTCExtendedServices` object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IFPDocument Object

The IFPDocument object represents a Microsoft FrontPage document. It includes properties and methods that return objects compatible with Microsoft Internet Explorer 4.0 and later, but not with the Internet Explorer 5 Dynamic HTML object model. See also the FPHTMLDocument object and the IHTMLDocument2 object.
Using the IFPDocument object

Use the `ActiveDocument` property to return an `IFPDocument` object. The following example creates an `IFPDocument` object variable, assigns it to the active document, and then uses the `IsDirty` property to see if the page has changed. If the page has changed, the example saves it.

```vbscript
Sub SaveChangedPage()
    Dim objDoc As IFPDocument
    On Error GoTo UnableToSavePage
    Set objDoc = ActiveDocument
    If objDoc.IsDirty = True Then ActivePageWindow.Save
    ExitSub:
    Exit Sub

    UnableToSavePage:
    MsgBox "Unable to save the page. " & vbCrLf & "If you haven't saved the page previously, " & vbCrLf & "you need to save it first before " & vbCrLf & "you can use the Save method."
    GoTo ExitSub
End Sub
```

Use the `DocumentHTML` property to return a `String` that contains the entire HTML and text within the specified document. The following statement retrieves the HTML in the active document, replaces every occurrence of "red" with "green", and writes the changed HTML back to the document.

```vbscript
Sub ViewDocumentHTML()
    Dim objDoc As IFPDocument
    Dim strHTML As String
    Set objDoc = ActiveDocument
    strHTML = objDoc.DocumentHTML
    strHTML = Replace(strHTML, "red", "green")
    objDoc.DocumentHTML = strHTML
End Sub
```
IFPLayoutRegion Object

Represents a two dimensional region in a document that is specified as containing special table cell formatting, referred to as layout cells, layout tables, and cell formatting tables. The IFPLayoutRegion object provides access to a limited number of properties and methods related to layout cells, layout tables, and cell formatting tables. For access to all properties and methods, use the FPLLayoutRegion object.
Using the IFPLayoutRegion Object

Use the layoutRegion method to return a single IFPLayoutRegion object in a document.
IFPStyleLength Object

IFPStyleState | IFPStyleLength

Contains information about the measurement properties of a style attribute associated with a given element or text range on the current page.
Using the IFPStyleLength object

Use the `createStyleLength` method of the `IFPStyleState` object to return an `IFPStyleLength` object. The following example sets the line height of the first paragraph in the active document to 25 pixels.

```vba
Dim objPara As IHTMLElement
Dim objSS As IFPStyleState
Dim objSL As IFPStyleLength

Set objPara = ActiveDocument.all.tags("p").Item(0)
Set objSS = Application.ActiveDocument.createStyleState
Set objSL = objSS.createStyleLength

objSL.units = fpLengthUnitsPX
objSL.Value = 25

With objSS
    .gatherFromElement objPara
    .lineHeight = objSL
    .Apply
End With
```
IFPStyleState Object

Multiple objects IFPStyleState
   IFPStyleLength

Contains information about the styles associated with a given element or text range on the current page. The IFPStyleState object allows you to view and edit any style property associated with a given element. Once style properties are modified, Microsoft FrontPage uses the new style information to render the new text based on the original styles.
Using the IFPStyleState object

Use the **CreateStyleState** property of the **FPHTMLDocument** object to create an **IFPStyleState** object. The following example creates a new **IFPStyleState** object and stores it in a variable called `objSs`.

```vba
Sub NewStyleState()
    'Creates a new style state object
    Dim objSs As IFPStyleState
    Set objSs = Application.ActiveDocument.createStyleState
End Sub
```

Use the **gather** method to associate a style state with a specified element or range. The following example creates an **IFPStyleState** object, associates it with a text range on the active document, and modifies its properties. The resulting text is rendered with the new styles.

```vba
Sub ChangeStyleState()
    'Changes the style state of a given text range
    Dim objSs As IFPStyleState
    Dim objRng As IHTMLTxtRange
    Application.ActiveDocument.body.innerHTML = "Hello Style State World"
    Set objRng = Application.ActiveDocument.body.createTextRange
    Set objSs = Application.ActiveDocument.createStyleState
    objSs.gather objRng
    objSs.fontWeight = 700
    objSs.fontFamily = "arial"
    objSs.Apply
End Sub
```

Use the **Apply** method to apply any changes made to the text range or element.
The `IFrontPageHostedControl` object is a member of the FrontPageHostedControl interfaces. For more information about how to use these when creating a Microsoft ActiveX control to use in Microsoft FrontPage, see the FrontPage Software Development Kit (SDK) on the Microsoft Developer Network (MSDN) Web site.
IFrontPageHostedControl2 Object

The IFrontPageHostedControl2 method is a member of the FrontPageHostedControl interfaces. For more information about how to use these when creating a Microsoft ActiveX control to use in Microsoft FrontPage, see the FrontPage Software Development Kit (SDK) on the Microsoft Developer Network (MSDN) Web site.
IHTMLAnchorElement Object

IHTMLAnchorElement

Represents a bookmark in an HTML document. Bookmarks are represented by A elements that use the name attribute. The IHTMLAnchorElement object provides references to a limited set of properties and methods related to a bookmark in a document. For access to all properties and methods, use the FPHTMLAnchorElement object.

Note  A elements that use the href attribute are hyperlinks. For information about accessing hyperlinks, see the FPHTMLLinkElement and IHTMLLinkElement objects.
Using the IHTMLAnchorElement object

Use the anchors property to return an IHTMLElementCollection object that represents all the bookmarks in a document. Use the Item method to return an IHTMLAnchorElement object that represents a specific bookmark, referenced by ordinal number or by the value of the id attribute. The following example accesses the first bookmark in the active document.

Dim objBookmark As IHTMLAnchorElement
Set objBookmark = ActiveDocument.anchors.Item(0)
IHTMLAreaElement Object

IHTMLAreaElement

Represents an AREA element. AREA elements are contained within MAP elements in an HTML document. Use the IHTMLAreaElement object to specify the coordinates and shape of an AREA element. See also the FPHTMLAreaElement object.
Using the IHTMLAreaElement object

Use the areas property of an IHTMLMapElement or FPHTMLMapElement object to return the IHTMLAreasCollection object of a MAP element. Use the Item method to return an IHTMLAreaElement object. The following example returns a string array containing the values of the href property, which is equivalent to a hyperlink, for all the IHTMLAreaElement objects in the specified FPHTMLMapElement object.

Function GetAreaHREF(objMap As FPHTMLMapElement) As String()
    Dim objArea As IHTMLAreaElement
    Dim strAreas() As String
    Dim intCount As Integer

    ReDim strAreas(objMap.areas.Length - 1)

    For intCount = 0 To objMap.areas.Length - 1
        Set objArea = objMap.areas.Item(intCount)
        strAreas(intCount) = objArea.href
    Next

    GetAreaHREF = strAreas
End Function
IHTMLBaseElement Object

IHTMLBaseElement

Represents the BASE element of an HTML document. Use the IHTMLBaseElement object to specify the base URL for all relative URLs in a Web page. The IHTMLBaseElement object has limited access to properties related to the BASE element. For access to all properties and methods, use the FPHTMLBaseElement object.
Using the IHTMLBaseElement object

Use the **tags** method to return an **IHTMLElementCollection** collection that represent all BASE elements in a document.

Use the **Item** method to return an **ITHMLBaseElement** object that accesses a specific BASE element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first BASE element in the active document.

```
Dim objBase As IHTMLBaseElement
Set objBase = ActiveDocument.all.tags("base").Item(0)
```
IHTMLBaseFontElement Object

IHTMLBaseFontElement

Represents the BASEFONT element in an HTML document. Use the IHTMLBaseFontElement object to specify base font characteristics such as name, size, and color. The IHTMLBaseFontElement object provides access to a limited number of properties and methods for the BASEFONT element. For access to all properties and methods, use the FPHTMLBaseFontElement object.
Using the IHTMLBaseFontElement object

Use the tags method to return an IHTMLElementCollection collection that represents all the BASEFONT elements in a document. Use the Item method to return an IHTMLBaseFontElement object that accesses a specific BASEFONT element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first BASEFONT element in the active document.

Dim objBaseFont As IHTMLBaseFontElement
Set objBaseFont = ActiveDocument.body.all.tags("basefont").Item(0)
IHTMLBGsound Object

IHTMLBGsound

Represents the BGSOUND element in an HTML document. The BGSOUND element plays a sound file in the background when a document is loaded into a browser. The IHTMLBGsound object provides access to a limited number of properties and methods related to the BGSOUND element. For access to all properties and methods, use the FPHTMLBGSound object.
Using the IHTMLBGsound object

Use the tags method to return an IHTMLElementCollection collection that represents all the BGSOUND elements in a document. Use the Item method to access a specific BGSOUND element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first BGSOUND element in the active document.

Dim objSound As IHTMLBGsound

Set objSound = ActiveDocument.all.tags("bgsound").Item(0)
IHTMLBlockElement Object

IHTMLBlockElement

Represents the BLOCKQUOTE element in an HTML document. The IHTMLBlockElement object provides access to a very limited set of properties for the BLOCKQUOTE element. For access to all properties and methods, use the FPHtmlBlockElement object.
Using the IHTMLBlockElement Object

Use the tags method to return an IHTMLElementCollection collection that represents all the BLOCKQUOTE elements in a document. Use the Item method to access a specific BLOCKQUOTE element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first BLOCKQUOTE element in the active document.

Dim objBlock As IHTMLBlockElement

Set objBlock = ActiveDocument.all.tags("blockquote").Item(0)
IHTMLBodyElement Object

IHTMLBodyElement

Represents the BODY element in an HTML document. The IHTMLBodyElement object provides access to a limited number of properties and methods that relate to a BODY element. For access to all properties and methods, use the FPHTMLBody object.
Using the IHTMLBodyElement Object

Use the **Body** property to return an **IHTMLBodyElement** object. The following example creates an **IHTMLBodyElement** object that represents the BODY element in the active document.

```vbscript
Dim objBody As IHTMLBodyElement
Set objBody = ActiveDocument.all.tags("body").Item(0)
```
IHTMLBRElement Object

IHTMLBRElement

Represents the BR element in an HTML document. The IHTMLBRElement object provides limited access to properties and methods related to the BR element. For access to all properties and methods, use the FPHTMLBRElement object.
Using the IHTMLBRElement Object

Use the `tags` method to return an `IHTMLElementCollection` collection that represents all the BR elements in a document. Use the `Item` method to return an `IHTMLBRElement` object that accesses a specific BR element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first BR element in the active document.

```vbnet
Dim objBreak As IHTMLBRElement
Set objBreak = ActiveDocument.all.tags("br").Item(0)
```
IHTMLButtonElement Object

IHTMLButtonElement | IHTMLFormElement

Represents the BUTTON element in an HTML document. The IHTMLButtonElement object provides limited access to properties and method related to the BR element. For access to all properties and methods, use the FPHTMLButtonElement object.
Using the IHTMLButtonElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents all the BUTTON elements in a document. Use the **Item** method to return an **IHTMLButtonElement** object that accesses a specific BUTTON element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first BUTTON element in the active document.

Dim objButton As IHTMLButtonElement

Set objButton = ActiveDocument.all.tags("button").Item(0)
IHTMLCommentElement Object

The IHTMLCommentElement object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**IHTMLControlElement Object**

**IHTMLControlElement**

Represents HTML elements that provide user interactivity when they are displayed in a browser. Corresponding objects for these elements have, for example, `tabIndex` and `accessKey` properties, indicating that they are keyboard accessible, and `onfocus` and `onblur` properties, indicating that they are capable of receiving input focus. Elements that support the **IHTMLControlElement** object include:

<table>
<thead>
<tr>
<th>BODY</th>
<th>IFRAME</th>
<th>SELECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUTTON</td>
<td>IMAGE</td>
<td>TABLE</td>
</tr>
<tr>
<td>CAPTION</td>
<td>INPUT</td>
<td>TD</td>
</tr>
<tr>
<td>EMBED</td>
<td>LEGEND</td>
<td>TEXTAREA</td>
</tr>
<tr>
<td>FIELDSET</td>
<td>MARQUEE</td>
<td>TH</td>
</tr>
<tr>
<td>FRAME</td>
<td>OBJECT</td>
<td></td>
</tr>
</tbody>
</table>
Using the IHTMLControlElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all of the specified elements in a document. Use the **Item** method to return an **IHTMLControlElement** object that accesses a specific element, referenced by ordinal number or by the value of the **id** attribute. The following example sets the BODY element in the active document to an **IHTMLControlElement** object.

```vbnet
Dim objControl As IHTMLControlElement
Set objControl = ActiveDocument.body
```
IHTMLDatabinding Object

IHTMLDatabinding

Represents elements that can be bound to data. Data binding, or client-side data binding, is a proprietary Microsoft Internet Explorer technology for which Microsoft FrontPage provides support through the IHTMLDatabinding object. The elements that provide databinding functionality and can use the IHTMLDatabinding object are the following:

- A
- img
- option
- button
- input
- select
- div
- label
- span
- frame
- marquee
- table
- iframe
- object
- textarea
Using the IHTMLDatabinding Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all of the specified elements in a document. Use the **Item** method to return an **IHTMLDatabinding** object that accesses a specific element, referenced by ordinal number or by the value of the **id** attribute. The following example sets the first IMG element in the active document to an **IHTMLDatabinding** object.

```vba
Dim objDataBound As IHTMLDatabinding
Set objDataBound = ActiveDocument.all.tags("img").Item(0)
```
IHTMLDDElement Object

IHTMLDDElement

Represents a DD element in an HTML document. The IHTMLDDElement object provides access to a limited number of properties and methods for the DD element. For access to all properties and methods, use the FPHTMLDDElement object.
Using the IHTMLDDElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents all the DD elements in a document. Use the **Item** method to return an **IHTMLDDElement** object that accesses a specific DD element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first DD element in the active document.

Dim objDescr As IHTMLDDElement

Set objDescr = ActiveDocument.all.tags("dd").Item(0)
IHTMLDialog Object

The IHTMLDialog object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IHTMLDivElement Object

IHTMLDivElement

Represents a DIV element in an HTML document. The IHTMLDivElement object provides access to a limited number of properties and methods that relate to the DIV element. For access to all properties and methods, use the FPHTMLDivElement object.
Using the FPHTMLDivElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents all the DIV elements in a document. Use the **Item** method to return an **IHTMLDivElement** object that accesses a specific DIV element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first DIV element in the active document.

```vbscript
Dim objDiv As IHTMLDivElement
Set objDiv = ActiveDocument.all.tags("div").Item(0)
```
IHTMLEDivPosition Object

The IHTMLEDivPosition object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IHTMLDListElement Object

IHTMLDListElement

Represents a DL element in an HTML document. The IHTMLDListElement object provides access to a limited number of properties and methods for the DL element. For access to all properties and methods, use the FPHTMLDListElement object.
Using the IHTMLDListElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents all the DL elements in a document. Use the **Item** method to return an **IHTMLDListElement** object that accesses a specific DL element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first DL element in the active document.

```vba
Dim objDef As IHTMLDListElement
Set objDef = ActiveDocument.all.tags("dl").Item(0)
```
IHTMLDocument2 Object

Multiple objects

- IHTMLDocument2

- Multiple objects

Represents the active HTML document. The IHTMLDocument2 object contains property information related to the HTML in a Web page such as background color, link color, and style sheet as well as methods that act on the data such as the CreateElement and ExecCommand methods.

The IHTMLDocument2 object provides access to a limited number of properties and methods that relate to a Web page. For access to all properties and methods, use the FPHTMLDocument object.
Using the IHTMLDocument2 object

Use the ActiveDocument property of the Application object or the Document property of the PageWindowEx object to return an IHTMLDocument2 object. The following example accesses the active document by using the ActivePageWindow property.

Dim objDoc As IHTMLDocument2
Set objDoc = ActivePageWindow.Document

The following example accesses the active document by using the ActiveDocument property.

Dim objDoc As IHTMLDocument2
Set objDoc = ActiveDocument

Note There is no way to access the contents of a Web page using Visual Basic for Applications (VBA) in Microsoft FrontPage without first opening the document and setting it as the active document using either the ActivePageWindow property, as shown in the preceding example, or the ActiveDocument property.
Remarks

The `PageWindowEx` object, in the FrontPage Web Object Model, contains both an `ActiveDocument` property and a `Document` property. Both properties return an `IHTMLDocument2` object. However, when a FRAMESET element is contained within a page, the `Document` property returns the page containing the FRAMESET element, and the `ActiveDocument` property returns the page referenced in the `SRC` attribute of the FRAME element located at the current insertion point position.
IHTMLDTElement Object

IHTMLDTElement

Represents a DT element in an HTML document. The IHTMLDTElement object provides access to a limited number of properties and methods for the DT element. For access to all properties and methods, use the FPHTMLDTElement object.
Using the IHTMLDTEElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents all the DT elements in a document. Use the **Item** method to return an **IHTMLDDElement** object that accesses a specific DT element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first DT element in the active document.

Dim objTerm As IHTMLDTElement

Set objTerm = ActiveDocument.all.tags("dt").Item(0)
**IHTMLElement Object**

Multiple objects

Represented any HTML element in an HTML document.

While most HTML elements have a corresponding object, use the **IHTMLElement** object to access elements for which there isn't a corresponding object. For example, the HEAD element doesn't have a corresponding object in the Microsoft FrontPage Visual Basic for Applications (VBA) Page Object Model. In this case, to access the elements and properties related to the HEAD element, create an **IHTMLElement** object to access the HEAD element in a document.

Also use the **IHTMLElement** object to loop through objects in an **IHTMLElementCollection** collection that is made of more than one type of element.
Using the IHTMLElement Object

Use the `tags` method to return an `IHTMLElementCollection` collection that represents a collection of a specific type of element in a document. Use the `Item` method to return an `IHTMLElement` object that accesses a specific element, referenced by ordinal number or by the value of the `id` attribute.

The following example accesses the HEAD element in the active document. While there is only one HEAD element in an HTML document, the `tags` method returns an `IHTMLElementCollection` collection, so the `Item` method accesses the first, which is also the only HEAD element in the document.

```vbnet
Dim objElement As IHTMLElement
Set objElement = ActiveDocument.all.tags("head").Item(0)
```

The following example uses the `IHTMLElement` object to loop through all the elements in the body of the active document and assign an `id` attribute based on the name of the element and the ordinal position of the element in the document.

```vbnet
Dim objElement As IHTMLElement
Dim intCount As Integer
For Each objElement In ActiveDocument.body.all
    intCount = intCount + 1
    objElement.Id = objElement.tagName & intCount
Next
```
IHTMLEmbedElement Object

IHTMLEmbedElement

Represents an EMBED element in an HTML document. The IHTMLEmbedElement object provides access to a limited number of properties and methods related to the EMBED element. For access to all properties and methods, use the FPHTMLEmbed object.
Using the IHTMLEmbedElement Object

Use the embeds or plugins property to return an IHTMLElementCollection collection that represents a collection of all the EMBED elements in a document. Use the Item method to return an IHTMLEmbedElement object that accesses a specific EMBED element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first EMBED element in the active document.

Dim objEmbed As IHTMLEmbedElement
Set objEmbed = ActiveDocument.embeds.Item(0)
IHTMLEventObj Object

Multiple objects \[\text{IHTMLEventObj}\]
\[\text{IHTMLElement}\]

Represents an user action that occurs because on a document, window, or element within a document.
Using the IHTMLEventObj object

Use the `event` property of the `FPHTMLWindow2` object to return an `IHTMLEventObj` object.

The following example creates an `IHTMLEventObj` object within the `onclick` event for the active document window which causes Microsoft FrontPage to display the number of times the mouse button has been clicked when a user clicks on the document.

```vbscript
Private WithEvents objDoc As FPHTMLDocument
Public objWindow As FPHTMLWindow2

Private Sub Class_Initialize()
    Set objWindow = ActiveDocument.parentWindow
    Set objDoc = ActiveDocument
End Sub

Private Function objDoc_onclick() As Boolean
    Dim objEvent As IHTMLEventObj
    Set objEvent = objWindow.event
    MsgBox objEvent.Button
End Function

Place the above code in a class module named "Class1." Then place the following code in a regular code module and run it to activate the class module and the resulting events.

```vbscript
Dim objFPEvent As Class1
Sub startevent()
    Set objFPEvent = New Class1
End Sub
```
IHTMLFieldSetElement Object

IHTMLFieldSetElement

Represents a FIELDSET element in an HTML document. The IHTMLFieldSetElement object provides access to a limited number of properties and methods related to the FIELDSET element. For access to additional properties and methods, use the FPHTMLFieldSetElement object.
Using the IHTMFieldSetElement Object

Use the \texttt{tags} method to return an \texttt{IHTMLElementCollection} collection that represents a collection of all the FIELDSET elements in a document. Use the \texttt{Item} method to return an \texttt{IHTMLFieldSetElement} object that accesses a specific FIELDSET element, referenced by ordinal number or by the value of the \texttt{id} attribute. The following example accesses the first FIELDSET element in the active document.

\begin{verbatim}
Dim objFields As IHTMLElementCollection
Dim objField As IHTMLFieldSetElement

Set objFields = ActiveDocument.all.tags("fieldset")
Set objField = objFields.Item(0)
\end{verbatim}
IHTMLFiltersCollection Collection

The IHTMLFiltersCollection collection is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IHTMLFontElement Object

IHTMLFontElement

Represents a FONT element in an HTML document. The IHTMLFontElement object provides access to a limited number of properties and methods related to the FONT element. For access to all properties and methods, use the FPHTMLFontElement object.
Using the IHTMLFontElement Object

Use the `tags` method to return an `IHTMLElementCollection` collection that represents a collection of all FONT elements in a document. Use the `Item` method to return an `IHTMLFontElement` object that accesses a specific FONT element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first FONT element in the active document.

```vba
Dim objFont As IHTMLFontElement
Set objFont = ActiveDocument.all.tags("font").Item(0)
```
Multiple objects  

Represents a FORM element in an HTML document. The **IHTMLFormElement** object provides access to a limited number of properties and methods related to the FORM element. For access to all properties and methods, use the **FPHTMLFormElement** object.
Using the IHTMLFormElement Object

Use the forms property to return an IHTMLElementCollection collection that represents a collection of all the FORM elements in a document. Use the Item method to return an IHTMLFormElement object that accesses a specific FORM element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first FORM element in the active document.

Dim objForm As IHTMLFormElement
Set objForm = ActiveDocument.forms.Item(0)
IHTMLFrameBase Object

The IHTMLFrameBase object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IHTMLFrameElement Object

IHTMLFrameElement

Represents a FRAME element in an HTML document. The IHTMLFrameElement object provides access to a limited number of properties and methods related to the FRAME element. For access to all properties and methods, use the FPHTMLFrameElement object.
Using the IHTMLFrameElement Object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all FRAME elements in a document. Use the Item method to return an IHTMLFrameElement object that accesses a specific FRAME element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first FRAME element in the active document.

Dim objFrame As IHTMLFrameElement
Set objFrame = ActiveDocument.all.tags("frame").Item(0)
IHTMLFrameSetElement Object

IHTMLFrameSetElement

Represents a FRAMESET element in an HTML document. The IHTMLFrameSetElement object provides limited access to properties and methods related to the FRAMESET element. For access to all properties and methods, use the FPHTMLFrameSetSite object.
Using the IHTMLFrameSetElement Object

Use the `tags` method to return an `IHTMLElementCollection` collection that represents the collection of FRAMESET elements in a document. Use the `Item` method to access an `IHTMLFrameSetElement` object that accesses a specific FRAMESET element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first FRAMESET element in the active document.

```vbscript
Dim objFrameSet As IHTMLFrameSetElement
Set objFrameSet = ActiveDocument.all.tags("frameset").Item(0)
```
IHTMLFrontPageBotElement Object

IHTMLFrontPageBotElement

Represents a component in Microsoft FrontPage. See also the FPHTMLFrontPageBotElement and IHTMLFrontPageBotElement2 objects.
Using the IHTMLFrontPageBotElement object

FrontPage provides the ability to get, remove, and set attributes for FrontPage components. The following example illustrate accessing the `getBotAttribute`, `removeBotAttribute`, and `setBotAttribute` methods.

Private Sub AccessBots()
    Dim objFPBot As IHTMLFrontPageBotElement
    Dim objBody As FPHTMLBody
    Dim strBot As String
    Dim objPage As PageWindow

    strBot = ""
    strBot = strBot & "<!-- webbot bot=""Search"" s-index=""All""-->
    strBot = strBot & " s-fields s-text=""Search for:"""
    strBot = strBot & " i-size=""20"" s-submit=""Start Search"""
    strBot = strBot & " s-clear=""Reset"" s-timestampformat=""%m/%d/"
    strBot = strBot & " tag=""BODY"" -->

    Set objBody = _ActivePageWindow.Document.body
    Set objPage = ActivePageWindow

    Call objBody.insertAdjacentHTML("BeforeEnd", _
        strBot)

    Set objFPBot = _
        objPage.Document.all.tags("webbot").Item(0)

    MsgBox objFPBot.getBotAttribute("s-submit")
    objFPBot.setBotAttribute "s-submit", "new item"
    MsgBox objFPBot.getBotAttribute("s-submit")
    objFPBot.removeBotAttribute "s-submit"
    MsgBox objFPBot.getBotAttribute("s-submit")
End Sub
IHTMLFrontPageBotElement2

Object

IHTMLFrontPageBotElement2

Represents a component in Microsoft FrontPage. See also the IHTMLFrontPageBotElement and IHTMLFrontPageBotElement2 objects.
Using the IHTMLFrontPageBotElement object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all the FrontPage components in a document. Use the **Item** method to access an **IHTMLFrontPageBotElement2** object that accesses a specific FrontPage component, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first FrontPage component in the active document.

Dim objWebbot As FPHTMLFrontPageBotElement

Set objWebbot = ActiveDocument.all.tags("webbot").Item(0)
IHTMLHeaderElement Object

IHTMLHeaderElement

Represents headings in an HTML document. Headings are represented using the H1 through H6 elements. The IHTMLHeaderElement object provides access to a limited number of properties and methods related to H elements. For access to all properties and members, use the FPHTMLHeaderElement object.
Using the IHTMLHeaderElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all the headings of a specified type in a document. For example, use `tags("h1")` to return a collection of all H1 elements in a document and use `tags("h2")` to return a collection of all H2 elements in a document.

Use the **Item** method to return an **IHTMLHeaderElement** object that accesses a specific H element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first H1 element in the active document.

```vbs
Dim objHeading As FPHTMLHeaderElement
Set objHeading = ActiveDocument.all.tags("h1").Item(0)
```
IHTMLHRElement Object

IHTMLHRElement

Represents an HR element in an HTML document. The IHTMLHRElement object provides access to a limited number of properties and methods related to HR elements. For access to all properties and methods, use the FPHTMLHRElement object.
Using the IHTMLHRElement Object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the HR elements in a document. Use the Item method to return an IHTMLHRElement object that accesses a specific HR element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first HR element in the active document.

Dim objLine As IHTMLHRElement

Set objLine = ActiveDocument.all.tags("hr").Item(0)
IHTMLIFrameElement Object

IHTMLIFrameElement

Represents an IFRAME element in an HTML document. The IHTMLIFrameElement object provides access to a limited number of properties and methods that relate to IFRAME elements. For access to all properties and methods, use the FPHTMLIFrame object.
Using the IHTMLIFrameElement Object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the IFRAME elements in a document. Use the Item method to return an IHTMLIFrameElement object that accesses a specific IFRAME element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first IFRAME element in the active document.

Dim objIFrame As IHTMLIFrameElement

Set objIFrame = ActiveDocument.all.tags("iframe").Item(0)
IHTMLImageElementFactory Object

The IHTMLImageElementFactory object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IHTMLImgElement Object

IHTMLImgElement

Represents an individual IMG element in an HTML document. The IHTMLImgElement object provides access to a limited number of properties and methods related to the IMG element. For access to all properties and methods, use the FPHTMLImg object.
Using the IHTMLImgElement Object

Use the `images` property to return an `IHTMLElementCollection` collection that represents a collection of all the IMG elements in a document. Use the `Item` method to return an `FPHTMLImg` object that accesses a specific IMG element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first IMG element in the active document.

```vba
Dim objImage As IHTMLImgElement
Set objImage = ActiveDocument.images.Item(0)
```
IHTMLInputButtonElement Object

IHTMLInputButtonElement \subset \text{IHTMLFormElement}

Represents an INPUT element of type "button" in an HTML document. The IHTMLInputButtonElement object provides access to a limited number of properties and methods related to the INPUT element. For access to additional properties and methods, use the FPHTMLInputButtonElement object.
Using the IHTMLInputButtonElement Object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the INPUT elements in a document. Use the Item method to return an IHTMLInputButtonElement object that accesses a specific INPUT element, referenced by ordinal number or by the value of the id attribute. Use the type property to determine the type of INPUT element.

Note The type property is not a member of the IHTMLElement object; however, it will return valid results for the value of the type attribute for an element, which in this case is "button".

The following example places the first INPUT element in the active document into an IHTMLElement object variable, then uses the type property to determine the type of INPUT element, and places the variable into an appropriate object variable, providing access to the properties and methods that relate to INPUT elements of the specified type.

```vbs
Dim objInput As IHTMLElement
Dim objButton As IHTMLInputButtonElement
Dim objFile As IHTMLInputFileElement
Dim objHidden As IHTMLInputHiddenElement
Dim objImage As IHTMLInputImage
Dim objText As IHTMLInputTextElement

Set objInput = ActiveDocument.all.tags("input").Item(0)
Select Case objInput.Type
    Case "button"
        Set objButton = objInput
    Case "file"
        Set objFile = objInput
    Case "hidden"
        Set objHidden = objInput
    Case "image"
        Set objImage = objInput
    Case "text"
        Set objText = objInput
End Select
```
IHTMLInputElement Object

IHTMLInputElement \langle IHTMLFormElement \rangle

Represents an INPUT element of type "file" in an HTML document. The IHTMLInputElement object provides access to a limited number of properties and methods related to the INPUT element. For access to additional properties and methods, use the FHTMLInputElement object.
Using the IHTMLInputElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all the INPUT elements in a document. Use the **Item** method to return an **IHTMLInputElement** object that accesses a specific INPUT element, referenced by ordinal number or by the value of the **id** attribute. Use the **type** property to determine the type of INPUT element.

**Note** The **type** property is not a member of the **IHTMLElement** object; however, it will return valid results for the value of the **type** attribute for an element, which in this case is "file".

The following example places the first INPUT element in the active document into an **IHTMLElement** object variable, then uses the **type** property to determine the type of INPUT element, and places the variable into an appropriate object variable, providing access to the properties and methods that relate to INPUT elements of the specified type.

```vbscript
Dim objInput As IHTMLElement
Dim objButton As IHTMLInputElement
Dim objFile As IHTMLInputElement
Dim objHidden As IHTMLInputElement
Dim objImage As IHTMLImageElement
Dim objText As IHTMLInputElement

Set objInput = ActiveDocument.all.tags("input").Item(0)

Select Case objInput.Type
  Case "button"
    Set objButton = objInput
  Case "file"
    Set objFile = objInput
  Case "hidden"
    Set objHidden = objInput
  Case "image"
    Set objImage = objInput
  Case "text"
    Set objText = objInput
End Select
```
IHTMLInputHiddenElement Object

IHTMLInputHiddenElement \<\ IHTMLFormElement

Represents an INPUT element of type "hidden" in an HTML document. The IHTMLInputHiddenElement object provides access to a limited number of properties and methods related to the INPUT element. For access to additional properties and methods, use the FPHTMLInputHiddenElement object.
Using the IHTMLInputElementHiddenElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all the INPUT elements in a document. Use the **Item** method to return an **IHTMLInputElementHiddenElement** object that accesses a specific INPUT element, referenced by ordinal number or by the value of the **id** attribute. Use the **type** property to determine the type of INPUT element.

**Note** The **type** property is not a member of the **IHTMLElement** object; however, it will return valid results for the value of the **type** attribute for an element, which in this case is "hidden".

The following example places the first INPUT element in the active document into an **IHTMLElement** object variable, then uses the **type** property to determine the type of INPUT element, and places the variable into an appropriate object variable, providing access to the properties and methods that relate to INPUT elements of the specified type.

```vbs
Dim objInput As IHTMLElement
Dim objButton As IHTMLInputButtonElement
Dim objFile As IHTMLInputFileElement
Dim objHidden As IHTMLInputHiddenElement
Dim objImage As IHTMLInputImage
Dim objText As IHTMLInputTextElement

Set objInput = ActiveDocument.all.tags("input").Item(0)

Select Case objInput.Type
    Case "button"
        Set objButton = objInput
    Case "file"
        Set objFile = objInput
    Case "hidden"
        Set objHidden = objInput
    Case "image"
        Set objImage = objInput
    Case "text"
        Set objText = objInput
End Select
```
IHTMLInputImage Object

IHTMLInputImage

Represents an INPUT element of type "image" in an HTML document. The IHTMLInputImage object provides access to a limited number of properties and methods related to the INPUT element. For access to additional properties and methods, use the FPHTMLInputImage object.
Using the IHTMLInputImage Object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the INPUT elements in a document. Use the Item method to return an IHTMLInputImage object that accesses a specific INPUT element, referenced by ordinal number or by the value of the id attribute. Use the type property to determine the type of INPUT element.

Note The type property is not a member of the IHTMLElement object; however, it will return valid results for the value of the type attribute for an element, which in this case is "image".

The following example places the first INPUT element in the active document into an IHTMLElement object variable, then uses the type property to determine the type of INPUT element, and places the variable into an appropriate object variable, providing access to the properties and methods that relate to INPUT elements of the specified type.

```vbnet
Dim objInput As IHTMLElement
Dim objButton As IHTMLInputButtonElement
Dim objFile As IHTMLInputFileElement
Dim objHidden As IHTMLInputHiddenElement
Dim objImage As IHTMLInputImage
Dim objText As IHTMLInputTextElement

Set objInput = ActiveDocument.all.tags("input").Item(0)

Select Case objInput.Type
    Case "button"
        Set objButton = objInput
    Case "file"
        Set objFile = objInput
    Case "hidden"
        Set objHidden = objInput
    Case "image"
        Set objImage = objInput
    Case "text"
        Set objText = objInput
End Select```

IHTMLInputTextElement Object

IHTMLInputTextElement  ↓  IHTMLFormElement

Represents an INPUT element of type "text" in an HTML document. The IHTMLInputTextElement object provides access to a limited number of properties and methods related to the INPUT element. For access to additional properties and methods, use the FPHTMLInputTextElement object.
Using the IHTMLInputTextElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all the INPUT elements in a document. Use the **Item** method to return an **IHTMLInputTextElement** object that accesses a specific INPUT element, referenced by ordinal number or by the value of the **id** attribute. Use the **type** property to determine the type of INPUT element.

**Note** The **type** property is not a member of the **IHTMLElement** object; however, it will return valid results for the value of the **type** attribute for an element, which in this case is "text".

The following example places the first INPUT element in the active document into an **IHTMLElement** object variable, then uses the **type** property to determine the type of INPUT element, and places the variable into an appropriate object variable, providing access to the properties and methods that relate to INPUT elements of the specified type.

```vbscript
Dim objInput As IHTMLElement
Dim objButton As IHTMLInputButtonElement
Dim objFile As IHTMLInputFileElement
Dim objHidden As IHTMLInputHiddenElement
Dim objImage As IHTMLInputImage
Dim objText As IHTMLInputTextElement

Set objInput = ActiveDocument.all.tags("input").Item(0)

Select Case objInput.Type
    Case "button"
        Set objButton = objInput
    Case "file"
        Set objFile = objInput
    Case "hidden"
        Set objHidden = objInput
    Case "image"
        Set objImage = objInput
    Case "text"
        Set objText = objInput
End Select
```
IHTMLIsIndexElement Object

IHTMLIsIndexElement

Represents an ISINDEX element in an HTML document. The IHTMLIsIndexElement object provides access to a limited number of properties and methods that relate to the ISINDEX element. For access to all properties and methods, use the FHTMLIsIndexElement object.
Using the IHTMLIsIndexElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all the ISINDEX elements in a document. Use the **Item** method to return an **IHTMLIsIndexElement** object that accesses a specific ISINDEX element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first ISINDEX element in the active document.

Dim objIndex As IHTMLIsIndexElement

Set objIndex = ActiveDocument.all.tags("isindex").Item(0)
IHTMLLabelElement Object

IHTMLLabelElement

Represents a LABEL element in an HTML document. The IHTMLLabelElement provides access to a limited number of properties and methods related to the LABEL element. For access to all properties and methods, use the FPHTMLLabelElement object.
Using the IHTMLLabelElement Object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the LABEL elements in a document. Use the Item method to return an IHTMLLabelElement object that accesses a specific LABEL element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first LABEL element in the active document.

Dim objLabel As IHTMLLabelElement
Set objLabel = ActiveDocument.all.tags("label").Item(0)
IHTMLLegendElement Object

IHTMLLegendElement

Represents a LEGEND element for a FIELDSET element in an HTML document. The FPHTMLLegendElement provides access to a limited number of properties and methods related to the LEGEND element. For access to all properties and methods, use the FPHTMLLegendElement object.
Using the IHTMLLegendElement Object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the LEGEND elements in a document. Use the Item method to return an IHTMLLegendElement object that accesses a specific LEGEND element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first LEGEND element in the active document.

Dim objLegend As IHTMLLegendElement
Set objLegend = ActiveDocument.all.tags("legend").Item(0)
IHTMLLIIEElement Object

IHTMLLIIEElement

Represents a LI element in an HTML document. The **IHTMLLIIEElement** object provides access to a limited number of properties and methods that relate to the LI element. For access to additional properties and methods, use the **FPHTMLLIIElement** object.
Using the IHTMLELIEElement Object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the LI elements in a document. Use the Item method to return an IHTMLELIEElement object that accesses a specific LI element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first LI element in the active document.

Dim objItem As IHTMLELIEElement

Set objItem = ActiveDocument.all.tags("li").Item(0)

The following example accesses the first LI element in the first ordered list in the active document.

Dim objList As IHTMLULEListElement
Dim objItem As IHTMLELIEElement

Set objList = ActiveDocument.all.tags("ul").Item(0)
Set objItem = objList.all.tags("li").Item(0)
IHTMLLinkElement Object

IHTMLLinkElement \[\text{IHTMLStyleSheet}\]

Represents a LINK element in an HTML document. The IHTMLLinkElement provides access to a limited number of properties and methods related to the LINK element. For access to all properties and methods, use the FPHTMLLinkElement object.
Using the IHTMLLinkElement object

Use the `tags` method to return an `IHTMLElementCollection` collection that represents a collection of all the LINK elements in a document. Use the `Item` method to return an `IHTMLLinkElement` object that accesses a specific LINK element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first LINK element in the active document.

Dim objLink As IHTMLLinkElement
Set objLink = ActiveDocument.all.tags("link").Item(0)

**Note** The `IHTMLLinkElement` object cannot be accessed using the `links` property. The `links` property applies to hyperlinks in a document not the LINK element.
IHTMLLocation Object

Multiple objects | IHTMLLocation

The IHTMLLocation object contains members that provide information about the current URL.
Using the IHTMLLocation object

Use the `location` property to return an `IHTMLLocation` object. Use the `href` and `pathname` properties of the `IHTMLLocation` object to return the path and filename of a specified document. Use the `reload` method to reload the page. The following example reloads the specified document.

**Note** When you use the `reload` method on a saved document that has been changed but not resaved, a message appears asking if the user wants to revert to the previously saved version.

```vba
Sub ReloadDocument()
    Dim objLocation As IHTMLLocation

    Set objLocation = ActiveDocument.Location

    objLocation.reload
End Sub
```
IHTMLMapElement Object

IHTMLMapElement \[\text{IHTMLAreasCollection}\]

Represents a MAP element in an HTML document. See also FPHTMLMapElement object.
Using the IHTMLMapElement object

Use the **Item** method to return a specific **IHTMLMapElement** object. The following example accesses the first MAP element in the active document.

```vbscript
Dim objMap As IHTMLMapElement
Set objMap = ActiveDocument.all.tags("map").Item(0)
```
IHTMLMarqueeElement Object

IHTMLMarqueeElement

Represents a MARQUEE element in an HTML document. The IHTMLMarqueeElement provides access to a limited number of properties and methods related to the MARQUEE element. For access to all properties and methods, use the FPHTMLMarqueeElement object.
Using the IHTMLMarqueeElement Object

Use the `tags` method to return an `IHTMLElementCollection` collection that represents a collection of all the MARQUEE elements in a document. Use the `Item` method to return an `IHTMLMarqueeElement` object that accesses a specific MARQUEE element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first MARQUEE element in the active document.

```vbscript
Dim objMarquee As IHTMLMarqueeElement
Set objMarquee = ActiveDocument.all.tags("marquee").Item(0)
```
IHTMLMetaElement Object

IHTMLMetaElement

Represents a META element in an HTML document. The IHTMLMetaElement provides access to a limited number of properties and methods related to the MARQUEE element. For access to all properties and methods, use the FPHTMLMetaElement object.
Using the IHTMLMetaElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all the META elements in a document. Use the **Item** method to return an **IHTMLMetaElement** object that accesses a specific META element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first META element in the active document.

```vbnet
Dim objMeta As IHTMLMetaElement
Set objMeta = ActiveDocument.all.tags("meta").Item(0)
```
IHTMLMimeTypesCollection Object

The IHTMLMimeTypesCollection object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IHTMLNextIdElement Object

The IHTMLNextIdElement object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IHTMLObjectElement Object

IHTMLObjectElement \| IHTMLFormElement

Represents a OBJECT element in an HTML document. The **IHTMLObjectElement** object provides access to a limited number of properties and methods related to the OBJECT element. For access to all properties and methods, use the **FPHTMLObjectElement** object.
Using the IHTMLObjectElement object

Use the the `tags` method to return an `IHTMLElementCollection` collection that represents a collection of all the OBJECT elements in a document. Use the `Item` method to return an `IHTMLObjectElement` object that accesses a specific OBJECT element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first OBJECT element in the active document.

```vba
Dim objObject As IHTMLObjectElement
Set objObject = ActiveDocument.all.tags("object").Item(0)
```
IHTMLOListElement Object

IHTMLOListElement

Represents an OL element in an HTML document. The IHTMLOListElement object provides access to a limited number of properties and methods related to the OL element. For access to additional properties and methods, use the FPHTMLOListElement object.
Using the IHTMLOListElement Object

Use the `tags` method to return an `IHTMLElementCollection` collection that represents a collection of all the OL elements in a document. Use the `Item` method to return an `IHTMLOListElement` object that accesses a specific OL element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first OL element in the active document.

Dim objList As IHTMLOListElement
Set objList = ActiveDocument.all.tags("ol").Item(0)
IHTMLOpsProfile Object

The IHTMLOpsProfile object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IHTMLOptionButtonElement Object

The IHTMLOptionButtonElement object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IHTMLOptionElement Object

Represents an OPTION element in an HTML document. The IHTMLOptionElement object provides access to a limited number of properties and methods related to the OPTION element. For access to all properties and methods, use the FPHTMLOptionElement object.
Using the IHTMLOptionElement Object

Use the **options** property to return an **IHTMLElementCollection** collection that represents a collection of all the OPTION elements in a SELECT element. Use the **Item** method to return an **IHTMLOptionElement** object that accesses a specific OPTION element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first OPTION element of the first SELECT element in the active document.

```vbscript
Dim objSelect As IHTMLSelectElement
Dim objOption As IHTMLOptionElement

Set objSelect = ActiveDocument.all.tags("select").Item(0)
Set objOption = objSelect.Options(0)
```
IHTMLOptionElementFactory Object

The IHTMLOptionElementFactory object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IHTMLParaElement Object

*IHTMLParaElement*

Represents a P element in an HTML document. The *IHTMLParaElement* object provides a limited number of properties and methods related to the P element. For access to all properties and methods, use the *FPHTMLParaElement* object.
Using the IHTMLParaElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all the P elements in a document. Use the **Item** method to return an **IHTMLParaElement** object that accesses a specific P element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first P element in the active document.

```vbnet
Dim objPara As IHTMLParaElement
Set objPara = ActiveDocument.all.tags("p").Item(0)
```
IHTMLPluginsCollection Object

The IHTMLPluginsCollection object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IHTMLRuleStyle Object

The IHTMLRuleStyle object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IHTMLScreen Object

The **IHTMLScreen** object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IHTMLScriptElement Object

IHTMLScriptElement

Represents a SCRIPT element in an HTML document. The IHTMLScriptElement object provides access to a limited number of properties and methods related to the SCRIPT element. For access to all properties and methods, use the FPHTMLScriptElement object.
Using the IHTMLScriptElement object

Use the scripts property to return an IHTMLElementCollection collection that represents a collection of all the SCRIPT elements in a document. Use the Item method to return an IHTMLScriptElement object that accesses a specific SCRIPT element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first SCRIPT element in the active document.

Dim objScript As IHTMLScriptElement
Set objScript = ActiveDocument.Scripts.Item(0)
IHTMLSelectElement Object

IHTMLSelectElement \rightarrow IHTMLFormElement

Represents a SELECT element in an HTML document. The IHTMLSelectElement object provides access to a limited number of properties and methods related to the SELECT element. For access to all properties and methods, use the FPHTMLSelectElement object.
Using the IHTMLSelectElement Object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the SELECT elements in a document. Use the Item method to return an IHTMLSelectElement object that accesses a specific SELECT element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first SELECT element in the active document.

Dim objSelect As IHTMLSelectElement

Set objSelect = ActiveDocument.all.tags("select").Item(0)
IHTMLSelectionObject Object

Multiple objects | IHTMLSelectionObject

Represents the text selected in the page window in Microsoft FrontPage.
Using the IHTMLSelectionObject Object

Use the `selection` property of the `FPHTMLDocument` or `IHTMLDocument2` object to return an `IHTMLSelectionObject` object. The following example accesses the selection in the active document.

Dim objSelection As IHTMLSelectionObject
Set objSelection = ActiveDocument.Selection

The `IHTMLSelectionObject` object contains methods, such as the `clear` method, that allows you to clear the contents of the selection if the type property for the selection returns "text". The following example clears the contents of the current selection.

If ActiveDocument.Selection.Type = "Text" Then _
   ActiveDocument.Selection.Clear

Use the `empty` method to clear the contents of a selection if the `type` property returns something other than "text".

If ActiveDocument.Selection.Type <> "Text" Then _
   ActiveDocument.Selection.empty

Use the `createRange` method to create an `IHTMLTxtRange` object to access additional properties and method that you can perform on the selected text.

Dim objRange As IHTMLTxtRange
Set objRange = ActiveDocument.Selection.createRange
**IHTMLSpanFlow Object**

The **IHTMLSpanFlow** object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IHTMLStyle Object

Multiple objects IHTMLStyle

Represents a style attribute for the cascading style sheet (CSS) for an HTML element. The IHTMLStyle object provides access to a limited number of properties and methods related to the style attribute. For access to all properties and methods, use the FPHTMLStyle object.
Using the IHTMLStyle object

Use the style property to return an IHTMLStyle object. The following example accesses the style attribute for the body of the active document.

Dim objStyle As IHTMLStyle
Set objStyle = ActiveDocument.body.Style
IHTMLStyleElement Object

IHTMLStyleElement  \| IHTMLStyleSheet

Represents a STYLE element in an HTML document. The IHTMLStyleElement object provides access to a limited number of properties and methods related to the STYLE element. For access to all properties and methods, use the FPHTMLStyleElement object.
Using the IHTMLStyleElement object

Use the the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all the STYLE elements in a document. Use the **Item** method to return an **IHTMLStyleElement** object that accesses a specific STYLE element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first STYLE element in the active document.

```vba
Dim objStyle As IHTMLStyleElement
Set objStyle = ActiveDocument.all.tags("style").Item(0)
```
**IHTMLStyleFontFace Object**

The **IHTMLStyleFontFace** object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IHTMLStyleSheet Object

Multiple objects

Represents a cascading style sheet reference in an HTML document. The IHTMLStyleSheet object provides access to a limited number of properties and methods related to cascading style sheets. For access to all properties and methods, use the FPHTMLStyleSheet object.
Using the IHTMLStyleSheet object

Use the **stylesheets** property to return an **FPHTMLStyleSheetCollection** collection that represents a collection of all the cascading style sheets referenced in a document. Use the **Item** method to return an **IHTMLStyleSheet** object that accesses a specific cascading style sheet, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first cascading style sheet in the active document.

```
Dim objStyleSheet As IHTMLStyleSheet

Set objStyleSheet = ActiveDocument.styleSheets.Item(0)
```
IHTMLStyleSheetRule Object

IHTMLStyleSheetRulesCollection  IHTMLStyleSheetRule
IHTMLRuleStyle

Represents an individual rule from a collection of cascading style sheet (CSS) rules.
Using the IHTMLStyleSheetRule object

Use the rules property to return an IHTMLStyleSheetRulesCollection collection that represents a collection of all the cascading style sheet rules in an attached CSS. Use the Item method to return an IHTMLStyleSheetRule object that accesses a specific CSS rule, referenced by ordinal number. The following example accesses the first rule in the collection of CSS rules for the first CSS attached to the active document.

```vbs
Dim objCSS As FPHTMLStyleSheet
Dim objRules As IHTMLStyleSheetRulesCollection
Dim objRule As IHTMLStyleSheetRule

Set objCSS = ActiveDocument.stylesheets(0)
Set objRules = objCSS.rules
Set objRule = objRules.Item(0)
```
IHTMLTable Object

IHTMLTable

Multiple objects

Represents a TABLE element in an HTML document. The IHTMLTable object provides access to a limited number of properties and methods related to the TABLE element. For access to all properties and methods, use the FPHTMLTable object.
Using the IHTMLTable object

Use the the `tags` method to return an `IHTMLElementCollection` collection that represents a collection of all the ELE elements in a document. Use the `Item` method to return an `IHTMLTable` object that accesses a specific TABLE element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first TABLE element in the active document.

```vbs```
Dim objTable As IHTMLTable
Set objTable = ActiveDocument.all.tags("table").Item(0)
```

To change the properties for the TFOOT, THEAD, TR, and TD elements, use the `IHTMLTableSection`, `IHTMLTableRow`, `IHTMLTableCol`, and `IHTMLTableCell` objects.
IHTMLTableCaption Object

Multiple objects

IHTMLTableCaption

Represents a CAPTION element for a TABLE element in an HTML document. The IHTMLTableCaption object provides access to a limited number of properties and methods related to the CAPTION element. For access to all properties and methods, use the FPHTMLTableCaption object.
Using the IHTMLTableCaption object

Use the `caption` property of the IHTMLTable object to return an IHTMLTableCaption object that represents the CAPTION element for that table. The following example accesses the first CAPTION element in the first TABLE element in the active document.

```vbscript
Dim objTable As IHTMLTable
Dim objCaption As IHTMLTableCaption

Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objCaption = objTable.Caption
```
IHTMLTableCell Object

IHTMLTableCell

Represents a TD element in an HTML document. The **IHTMLTableCell** object provides access to a limited number of properties and methods related to the TD element. For access to all properties and methods, use the **FPHTMLTableCell** object.
Using the IHTMLOTableCell object

Use the `cells` property of the `IHTMLOTableRow` object to return an `IHTMLElementCollection` collection that represents a collection of all the TD elements in a row in a table. Use the `Item` method to return an `IHTMLOTableCell` object that accesses a specific TD element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first TD element in the first row in the first table in the active document.

```vbscript
Dim objTable As IHTMLOTable
Dim objRow As IHTMLOTableRow
Dim objCell As IHTMLOTableCell

Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objRow = objTable.rows(0)
Set objCell = objRow.cells(0)
```
IHTMLTableCol Object

IHTMLTableCol

Represents a COL element in an HTML document. The IHTMLTableCol object provides access to a limited number of properties and methods related to the COL element. For access to all properties and methods, use the FPHTMLTableCol object.
Using the IHTMLTableCol object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all the COL elements in a TABLE element in a document. Use the **Item** method to return an **IHTMLTableCol** object that accesses a specific COL element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first COL element in the first table in the active document.

```vbscript
Dim objTable As IHTMLTable
Dim objCol As IHTMLTableCol

Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objCol = objTable.all.tags("col").Item(0)
```
IHTMLTableRow Object

IHTMLTableRow \rightarrow IHTMLElementCollection

Represents a TR element in an HTML document. The IHTMLTableRow object provides access to a limited number of properties and methods related to the TR element. For access to all properties and methods, use the FPHTMLTableRow object.
Using the IHTMLTableRow object

Use the rows property to return an IHTMLElementCollection collection that represents a collection of all the TR elements in a table. Use the Item method to return an IHTMLTableRow object that accesses a specific TR element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first TR element in the first table in the active document.

Dim objTable As IHTMLTable
Dim objRow As IHTMLTableRow

Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objRow = objTable.rows(0)
IHTMLTableRowMetrics Object

The IHTMLTableRowMetrics object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IHTMLTableSection Object

Multiple objects \[\text{IHTMLTableSection}\]
\[\text{IHTMLElementCollection}\]

Represents the THEAD, TFOOT, and TBODY elements inside of a TABLE element in an HTML document. The \text{IHTMLTableSection} object provides access to a limited number of properties and methods related to the THEAD, TFOOT, and TBODY elements. For access to all properties and methods, use the \text{FPHTMLTableSection} object.
Using the IHTMLTableSection object

Use the tHead property of the IHTMLTable object to return an IHTMLTableSection object that represents the THEAD element for a table. The following example accesses the THEAD element for the first table in the active document.

Dim objTable As IHTMLTable
Dim objTHead As IHTMLTableSection
Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objTHead = objTable.tHead

Use the createTHead method of the IHTMLTable object to add a THEAD element. Use the deleteTHead method of the FPHTMLTable object to remove a THEAD element.

Use the tFoot property of the IHTMLTable object to return an IHTMLTableSection object that represents the TFOOT element for a table. The following example accesses the TFOOT element for the first table in the active document.

Dim objTable As IHTMLTable
Dim objTHead As IHTMLTableSection
Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objTHead = objTable.tFoot

Use the createTFoot method of the IHTMLTable object to add a TFOOT element. Use the deleteTFoot method of the IHTMLTable object to remove a TFOOT element.
IHTMLTemplateRegionElement Object

IHTMLTemplateRegionElement

Represents a region in an HTML document that corresponds to a region in a Dynamic Web Template. The IHTMLTemplateRegionElement object provides access to a limited number of properties and methods related to Dynamic Web Template regions. For access to all properties and methods, use the FPHtmlTemplateRegionElement object.
Using the IHTMLTemplateRegionElement object

Use the templateRegions property to return an IHTMLElementCollection collection that represents a collection of all the Dynamic Web Template regions in a document. Use the Item method to return an individual IHTMLTemplateRegionElement object that accesses a specific Dynamic Web Template region, referenced by ordinal number or by the name of the region. The following example accesses the first Dynamic Web Template region in the active document and sets the id attribute equal to the value of the name of the region.

Dim objRegions As IHTMLElementCollection
Dim objRegion As IHTMLTemplateRegionElement

Set objRegions = ActiveDocument.templateRegions
Set objRegion = objRegions.Item(0)

objRegion.Id = objRegion.Name
IHTMLTextAreaElement Object

IHTMLTextAreaElement \raction{\rightarrow} IHTMLFormElement

Represents a TEXTAREA element in an HTML document. The IHTMLTextAreaElement object provides access to a limited number of properties and methods related to the TEXTAREA element. For access to all properties and methods, use the FPHTMLTextAreaElement object.
Using the IHTMLTextAreaElement object

Use the `tags` method to return an `IHTMLElementCollection` collection that represents a collection of all the TEXTAREA elements in a document. Use the `Item` method to return an `IHTMLTextAreaElement` object that accesses a specific TEXTAREA element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first TEXTAREA element in the active document.

```vba
Dim objTextArea As IHTMLTextAreaElement

Set objTextArea = ActiveDocument.all.tags("textarea").Item(0)
```
IHTMLTextContainer Object

The IHTMLTextContainer object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IHTMLTitleElement Object

IHTMLTitleElement

Represents a TITLE element in an HTML document. The IHTMLTitleElement object provides access to a limited number of properties and methods related to the TITLE element. For access to all properties and methods, use the FPHTMLTitleElement object.
Using the IHTMLTitleElement object

Use the tags method to return an IHTMLElementCollection collection that represents a collection of all the TITLE elements in a document. Use the Item method to return an IHTMLTitleElement object that accesses a specific TITLE element, referenced by ordinal number or by the value of the id attribute. The following example accesses the first TITLE element in the active document.

Dim objTitle As FPHTMLTitleElement
Set objTitle = ActiveDocument.all.tags("title").Item(0)

Note While there can only be one TITLE element in an HTML document, the tags method by default returns an IHTMLElementCollection. Therefore, you must use the Item method to return a single IHTMLTitleElement object.

Note You can set the value of the TITLE element by using the title property.
IHTMLTxtRange Object

IHTMLTxtRange

Represents a range or selection of text and elements in an HTML document.
Using the IHTMLTxtRange object

Use the `createTextRange` method of the `FPHTMLBody` object to return an `IHTMLTxtRange` object. The following example creates a range that includes the entire body of the active document, and then selects the entire document.

```vba
Dim objRange As IHTMLTxtRange
Set objRange = ActiveDocument.body.createTextRange
objRange.Select
```

Use the `createRange` method of the `IHTMLSelectionObject` object to return an `IHTMLTxtRange` object that represents the currently selected text and elements in the active document. The following example creates a range from the current selection in the active document.

```vba
Dim objRange As IHTMLTxtRange
Set objRange = ActiveDocument.selection.createRange
```
IHTMLULListElement Object

IHTMLULListElement

Represents a UL element in an HTML document. The IHTMLULListElement object provides access to a limited number of properties and methods that relate to the UL element. For access to additional properties and methods, use the FPHTMLULListElement object.
Using the IHTMLULListElement Object

Use the **tags** method to return an **IHTMLElementCollection** collection that represents a collection of all the UL elements in a document. Use the **Item** method to return an **IHTMLULListElement** object that accesses a specific UL element, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first UL element in the active document.

Dim objList As IHTMLULListElement

Set objList = ActiveDocument.all.tags("ul").Item(0)
IHTMLWebPartElement Object

IHTMLWebPartElement

Represents a Web part embedded in an HTML document. The IHTMLWebPartElement object provides access to a limited number of properties and methods related to the embedded Web parts. For access to all properties and methods, use the FPHTMLWebPartElement object.

This object is supported only by Web pages or sites that are based on Microsoft Windows SharePoint Services.
Using the IHTMLWebPartElement object
Use the webParts property of the IFPDocument object to return an
IHTMLElementCollection collection that represents a collection of all the
embedded Web parts within a document. Use the Item method to return an
IHTMLWebPartElement object that accesses a specific Web part, referenced
by ordinal number. The following example accesses the first Web part in the
active document.
Dim objPart As IHTMLWebPartElement
Set objPart = ActiveDocument.webParts.Item(0)

Use the InsertAdjacentHTML method, as shown in the following example, to
insert a Web Part into a document.
Public Sub InsertWebPart()
Dim
Dim
Dim
Dim

strPageDirective As String
strRegisterDirective As String
strWebPart As String
objWebPart As FPHTMLWebPartElement

strPageDirective = "<%@ Page Language=""C#"" Debug=""true""%>"
strRegisterDirective = "<%@ Register TagPrefix=""WebPartPages"" " & _
"Namespace=""Microsoft.SharePoint.WebPartPages"" " & _
"Assembly=""Microsoft.SharePoint, Version=11.0.0.0, " & _
"Culture=neutral, PublicKeyToken=71e9bce111e9429c""%>"
strRegisterDirective & vbCrLf & ActiveDocument.DocumentHTML
strWebPart = "<form runat=""server"">" & vbCrLf & _
"<WebPartPages:ImageWebPart webpart=""true"" " & _
"runat=""server"" id=""insertedwebpart"">" & vbCrLf & _
"</WebPartPages:ImageWebPart>" & vbCrLf & "</form>"
ActiveDocument.body.insertAdjacentHTML "afterBegin", strWebPart
Set objWebPart = ActiveDocument.webParts.Item("insertedwebpart")
End Sub


IHTMLWebPartZoneElement Object

IHTMLWebPartZoneElement \< IHTMLElementCollection

Represents a Web part zone in an HTML document. The IHTMLWebPartZoneElement object provides access to a limited number of properties and methods related to the Web Part zone. For access to all properties and methods, use the FPHHTMLWebPartZoneElement object.

This object is supported only by Web pages or sites that are based on Microsoft Windows SharePoint Services.
Using the IHTMLWebPartZoneElement object

Use the 

**webPartZones**

property to return an 

**IHTMLElementCollection**

collection that represents a collection of all the Web Part zones in a document. Use the 

**Item**

method to return an 

**IHTMLWebPartZoneElement**

object that accesses a specific Web Part zone, referenced by ordinal number or by the value of the 

**id**

attribute. The following example accesses the first Web Part zone in the active document.

```vbnet
Dim objZone As IHTMLWebPartZoneElement
Set objZone = ActiveDocument.webPartZones.Item(0)
```
IHTMLWindow2 Object

Multiple objects

Represents the window in which an HTML document or a frame within an HTML document is displayed. The IHTMLWindow2 object provides access to a limited number of properties and methods related to the window. For access to all properties and methods, use the FPHTMLWindow2 object.
Using the IHTMLWindow2 object

Use the parentWindow property of the FPHTMLDocument object to return an IHTMLWindow2 object. The following example accesses the window in which the active document is displayed.

Dim objWindow As IHTMLWindow2

Set objWindow = ActiveDocument.parentWindow
IHTMLXSLElement Object

IHTMLXSLElement

Represents an Extensible Stylesheet Language (XSL) element for an XSL Data View Web Part. The IHTMLXSLElement object provides access to a limited number of properties and methods related to the XSL element. For access to all properties and methods, use the FPHTMLXSLElement object.

This object is supported only by Web pages or sites that are based on Microsoft Windows SharePoint Services.
Using the IHTMLXSLElement Object

Use the **all** property for the **FPHTMLXSLWebPartElement** object to return an **IHTMLElementCollection** collection that represents a collection of all the elements for an XSL Data View Web Part. Then use the **tags** method to return an **IHTMLElementCollection** collection that represents all of the XSL elements of the specified type (for example, "xsl-template", "xsl-outside-if", "xsl-value-of", etc.) in a Data View Web Part. The following example returns the first xsl:template element in the first Web Part in the active document. This example assumes that the first Web Part in the active document is a Data View Web Part.

```vba
Dim objWebPart As FPHTMLXSLWebPartElement
Dim objXSLElement As IHTMLXSLElement

Set objWebPart = ActiveDocument.WebParts.Item(0)
Set objXSLElement = objWebPart.all.tags("xsl-template").Item(0)
```
IHTMLXSLWebPartElement Object

IHTMLXSLWebPartElement

Represents an Extensible Stylesheet Language (XSL) Data View Web Part. The IHTMLXSLWebPartElement object provides access to a limited number of properties and methods related to the XSL Data View Web Part. For access to all properties and methods, use the FPHTMLXSLWebPartElement object.

This object is supported only by Web pages or sites that are based on Microsoft Windows SharePoint Services.
Using the IHTMLXSLWebPartElement object

Use the **webParts** property to return an **IHTMLElementCollection** collection that represents a collection of all the Web Parts in a document. Use the **Item** method to return an **IHTMLXSLWebPartElement** object that accesses a specific XSL Data View Web Part, referenced by ordinal number or by the value of the **id** attribute. The following example accesses the first Web Part in the active document. This example assumes that the first Web Part is an XSL Data View Web Part.

Dim objXSLWebPart As IHTMLXSLWebPartElement

Set objXSLWebPart = ActiveDocument.webParts.Item(0)
IOmHistory Object

The **IOmHistory** object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IOmNavigator Object

The **IOmNavigator** object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
IServiceProvider Object

The `IServiceProvider` object is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**SearchInfo Object**

*SearchInfo*

Provides access to programmatic search and replace functionality to pages in Microsoft FrontPage.
Using the SearchInfo object

Use the `CreateSearchInfo` method to create a `SearchInfo` object. The following example creates a new `SearchInfo` object.

```vba
Dim objSearch As SearchInfo
Set objSearch = Application.CreateSearchInfo
```

Use the `Find` property to specify a character or string of characters for which to perform the search. Use the `Action` property to specify the type of search to perform. The following example sets the `Find` and `Action` properties for the `SearchInfo` object created in the previous example.

```vba
objSearch.Find = "p"
objSearch.Action = fpSearchFindTag
```

Use the `Find` method of the `FPHTMLDocument` object to perform the search. The following example performs the search defined above starting at the insertion point.

```vba
Dim blnFound As Boolean
Dim objRange As IHTMLTxtRange
Set objRange = Application.ActiveDocument.selection.createRange
blnFound = Application.ActiveDocument.Find(objSearch, Nothing, objRange)
```

The following example combines the above code samples. This example searches for the next occurrence of the `P` element. If the `P` element is found, it selects the `P` element and associated text.

```vba
Dim objSearch As SearchInfo
Dim blnFound As Boolean
Dim objRange As IHTMLTxtRange
Set objSearch = Application.CreateSearchInfo
objSearch.Find = "p"
objSearch.Action = fpSearchFindTag
Set objRange = Application.ActiveDocument.selection.createRange
blnFound = Application.ActiveDocument.Find(objSearch, Nothing, objRange)
If blnFound = True Then objRange.Select
```
abort Method

Stops execution of the specified **FPHTMLUndoTransaction** object. For example, if you're running a macro and you've previously created an **FPHTMLUndoTransaction** object for the macro, you can use the **abort** method to stop the macro when an error condition results.

```plaintext
expression.abort
```

*expression* An expression that returns an **FPHTMLUndoTransaction** object.
Remarks

Once an **FPHTMLUndoTransaction** object is created, Microsoft FrontPage places the name of the **FPHTMLUndoTransaction** object in the **Undo** command on the **Edit** menu.
Example

Use the `createUndoTransaction` method of the `IFPDocument` object to create an `FPHTMLUndoTransaction` object. The `String` argument passed in with the `createUndoTransaction` method is added to the `Undo` menu when the `Commit` method is called. The following example creates an `FPHTMLUndoTransaction` object illustrating the `abort` and `Commit` methods.

```vbnet
Private Sub CreateUndoTransaction()
    Dim objDoc As FPHTMLDocument
    Dim objTransaction As FPHTMLUndoTransaction

    On Error GoTo CreateUndoTransactionError
    Set objDoc = ActiveDocument
    Set objTransaction = objDoc._.CreateUndoTransaction("Last Macro")

    objDoc.body.insertAdjacentHTML "BeforeEnd", _
        "<b>Added by FP Programmability</b>"

    objTransaction.Commit

ExitCreateUndoTransaction:
    Exit Sub

CreateUndoTransactionError:
    objTransaction.abort
    GoTo ExitCreateUndoTransaction
End Sub
```
add Method

Returns an IHTMLAreaElement that represents an AREA element added to a document.

expression.add(element, before)

expression Required. An expression that returns an IHTMLAreasCollection collection.

element Required. Specifies an IHTMLElement object that represents the image MAP element to which to add the AREA element.

before Optional. Variant. Specifies the index position in the collection where the new element is placed. If no value is given, the method places the element at the end of the collection.
Remarks

Use the createElement method to create the new AREA element, and then use the Add method to add the AREA element to the specified MAP element.
Example

This example creates an AREA element, specifies the shape, size, hyperlink address for the element, and adds it to the specified document.

Dim objArea As FPHTMLAreaElement
Set objArea = ActiveDocument.createElement("area")

objArea.Shape = "rect"
objArea.coords = "50,50,50,50"
objArea.href = "http://www.microsoft.com/frontpage"
ActiveDocument.all.tags("map").Item(0).areas.Add objArea
addFilter Method

The **addFilter** method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
addImport Method

Adds a style sheet to the collection of imported style sheets for a specified style sheet. Returns a Long that represents the index number of the new stylesheet.

expression.addImport(bstrURL, lIndex)

expression  Required. An expression that returns one of the objects in the Applies To list.

bstrURL  Required String. The path and filename for the external cascading style sheet.

lIndex  Optional Long. Specifies the requested position for the style sheet in the collection. If omitted, the style sheet is added to the end of the collection.
Remarks

Use the `removeImport` method to remove an imported style sheet from the collection of imported style sheets for a STYLE element. Use the `imports` property to access the collection of imported style sheets for a STYLE element.
**Example**

The following example checks to see if there are any imported style sheets referenced in the first STYLE element in the active document. If there are no imported style sheets, a new imported style sheet reference is added to the STYLE element.

```vbnet
Dim objStyle As FPHTMLStyleSheet
Set objStyle = ActiveDocument.styleSheets.Item(0)

If objStyle.imports.Length = 0 Then
    objStyle.addImport("web.css")
End If
```
addReadRequest Method

The `addReadRequest` method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
addRequest Method

The `addRequest` method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
addRule Method

Creates a new style rule for a cascading style sheet, and returns a Long that represents the index of the new style rule in the rules collection.

expression.addRule(bstrSelector, bstrStyle, lIndex)

expression  Required. An expression that returns one of the objects in the Applies To list.

bstrSelector  Required String. Specifies the selector for the new rule. Single contextual selectors are valid. For example, "DIV P B" is a valid contextual selector.

bstrStyle  Required String. Specifies the style assignments for this style rule. This style takes the same form as an inline style specification. For example, "color:blue" is a valid style parameter.

lIndex  Optional Long. Specifies the location in the rules collection to which to add the new style rule. If an index is not provided, the rule is added to the end of the collection by default.
Remarks

The `addRule` method only adds cascading style sheet rules to STYLE elements in a page and does not modify external style sheets referenced.
**Example**

The following example adds a cascading style sheet rule that sets the color of the text in the P element.

```vbscript
Dim objCSS As FPHTMLStyleSheet
Set objCSS = ActiveDocument.styleSheets(1)
objCSS.addRule "p", "color: rgb(255,153,0)"
```
addScript Method

Adds a script to an element for a specified event.

expression.addScript(element, event, script, index)

expression  Required. An expression that returns one of the objects in the Applies To list.

element  Required IHTMLElement. Specifies the element that receives the action.

event  Required String. Specifies the name of the event that causes the script to run.

script  Required String. Specifies the script that runs when the event occurs.

index  Optional Variant.
Example

The following example inserts a script into the onmouseover event for the first paragraph in the active document:

Dim objPara As FPHTMLParaElement
Set objPara = ActiveDocument.all.tags("p").Item(0)
ActiveDocument.addScript objPara, "onmouseover", "alert('hello!');"
alert Method

Displays a dialog box containing an application-defined message.

`expression.alert(message)`

`expression`  Required. An expression that returns one of the objects in the Applies To list.

`message`  Optional String, specifies the message to display in the Alert dialog box. If no value is provided, the dialog box contains no message.
Example

The following example displays a message box to the user.

```javascript
ActiveDocument.parentWindow.alert "Hello!"
```
apply Method

Applies a style setting to a specified object.

`expression.apply`

`expression`  Required. An expression that returns an `IFPStyleState` object.
Example

The following example inserts formatted text into the specified document, creates an **IHTMLTxtRange** object that includes the entire document, and then removes all formatting associated with the range. Then it inserts similarly formatted text directly below the first text inserted, creates a second **IHTMLTxtRange** object that again includes the entire document, and removes only text formatting associated with the range. The first paragraph inserted has all formatting removed, including paragraph formatting; the second paragraph retains its paragraph formatting but all text formatting is removed.

Sub ClearFormatting()
    Dim objSS As IFPStyleState
    Dim objDoc As FPHTMLDocument
    Dim objPara As FPHTMLParaElement
    Dim objRng As IHTMLTxtRange

    Set objDoc = ActiveDocument
    objDoc.body.innerHTML = "<p align="center"><b><i><u>" & 
        "<span style="background-color: #0000FF">Clears " & _
        "all formatting; paragraph is no longer centered" & _
        "</span></u></i></b></p>" & vbCrLf

    Set objSS = objDoc.createStyleState

    'Create the first text range.
    Set objRng = objDoc.body.createTextRange

    With objSS
        .gather objRng

        'Clear all formatting, including paragraph formatting.
        .ClearAllFormatting
        .Apply
    End With

    'Create the second text range.
    objDoc.body.insertAdjacentHTML "beforeend", "<p align=" & _
        ""center"">" & ""& "<span style="background" & _
        "-color: #0000FF">" & "Clears text formatting; " & _
        "paragraph formatting remains</span></u></i></b></p>"

    Set objRng = objDoc.body.createTextRange
End Sub
With objSS
  .gather objRng

  'Clear only formatting that applies to text.
  .ClearTextFormatting
  .Apply
End With

Set objSS = Nothing
Set objDoc = Nothing
Set objRng = Nothing
End Sub
applyDynamicTemplate Method

Applies a Dynamic Web Template to document.

*expression*.applyDynamicTemplate(*urlMaster, state*)

*expression* Required. An expression that returns one of the objects in the Applies To list.

*urlMaster* Required *String*. The path and file name for the Dynamic Web Template.

*state* Required *DynamicTemplateState*. Specifies the region mapping for the Dynamic Web Template. Use the *SetHeadMapping* and *SetBodyMapping* methods of the *DynamicTemplateState* object to customize region mapping.
Remarks

To detach a Dynamic Web Template, use the `ApplyDynamicTemplate` method with the `urlMaster` parameter set to an empty string.
Example

The following example applies the specified Dynamic Web Template to the active document.

Dim objState As DynamicTemplateState
Set objState = Application.CreateDynamicTemplateState
ActiveDocument.ApplyDynamicTemplate "template.dwt", objState
applyToElement Method

Applies an IFPStyleState object to a specified element.

`expression.applyToElement(element)`

`expression` Required. An expression that returns one of the objects in the Applies To list.

`element` Required IHTMLElement object. The name of the element that you want to apply a given style property to.
Example

The following example gathers style properties from an element, as specified in the objHeading1 variable. It then creates an IFPStyleState object, which is stored in the objSS variable, modifies the style properties, and then applies the modified style properties to the element stored in the objHeading2 variable.

Sub ApplyStyleToElement()
    Dim objSS As IFPStyleState
    Dim objHeading1 As IHTMLElement
    Dim objHeading2 As IHTMLElement

    ActiveDocument.body.innerHTML = "<H1>This is line 1</H1>" _
         & vbCrLf & "<H1>This is line 2</H1>"

    Set objHeading1 = ActiveDocument.all.tags("h1").Item(0)
    Set objHeading2 = ActiveDocument.all.tags("h1").Item(1)

    Set objSS = ActiveDocument.createStyleState

    With objSS
        .GatherFromElement objHeading1
        .fontFamily = "Arial"
        .Color = vbRed
        .textAlign = "right"
        .setProperty "background-color", vbBlue
        .textDecorationBlink = True
        .textDecorationLineThrough = True
        .textDecorationOverline = True
        .textDecorationUnderline = True
        .textTransform = "uppercase"
        .applyToElement objHeading2
    End With

    Set objSS = Nothing
    Set objHeading1 = Nothing
    Set objHeading2 = Nothing
End Sub
applyToRange Method

Applies the style properties of an IFPStyleState object to an IHTMLTxtRange object.

expression.applyToRange(range)

definition

expression  Required. An expression that returns one of the objects in the Applies To list.

range  Required IHTMLTxtRange object. The text range to which to apply the style.
Example

The following example creates an `IFPStyleState` object and applies its style properties to a text range stored in the `objRng` variable.

```vba
Sub ApplyStyleToRange()
    Dim objSS As IFPStyleState
    Dim objDoc As FPHTMLDocument
    Dim objRng As IHTMLTxtRange

    Set objDoc = Application.ActiveDocument

    objDoc.body.innerHTML = "<h1><b>Heading 1</b></h1>"
    Set objSS = objDoc.createStyleState
    Set objRng = objDoc.body.createTextRange

    With objSS
        .gather objRng
        .fontFamily = "Arial"
        .Color = vbRed
        .textAlign = "right"
        .setProperty "background-color", vbBlue
        .textDecorationLineThrough = True
        .textDecorationOverline = True
        .textDecorationUnderline = True
        .textTransform = "uppercase"
        .applyToRange objRng
    End With

    Set objSS = Nothing
    Set objDoc = Nothing
    Set objRng = Nothing
End Sub
```
assign Method

The **assign** method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
back Method

The back method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
blur Method

The `blur` method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
clear Method

The clear method is not supported in Microsoft FrontPage and is therefore not available for programmatic access.
clearAllFormatting Method

Clears any formatting associated with a given text range or element.

`expression.ClearAllFormatting`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

When you compare the ClearAllFormatting method with the ClearTextFormatting method, you notice that while the ClearAllFormatting method clears all formatting in the specified IHTMLTxtRange object, including paragraph formatting, the ClearTextFormatting method clears only formatting that modifies the appearance of text in the range.

Use the Apply method to apply the changes to the text range or element.
Example

The following example inserts formatted text into the specified document, creates an IHTMLTxtRange object that includes the entire document, and then removes all formatting associated with the range. Then it inserts similarly formatted text directly below the first text inserted, creates a second IHTMLTxtRange object that again includes the entire document, and removes only text formatting associated with the range. The first paragraph inserted has all formatting removed, including paragraph formatting; the second paragraph retains its paragraph formatting but all text formatting is removed.

Sub ClearFormatting()
    Dim objSS As IFPStyleState
    Dim objDoc As FPHTMLDocument
    Dim objPara As FPHTMLParaElement
    Dim objRng As IHTMLTxtRange

    Set objDoc = ActiveDocument

    objDoc.body.innerHTML = "<p align="center">"& _
    "<span style="background-color: #0000FF">Clears " & _
    "all formatting; paragraph is no longer centered" & _
    "</span></u></i></b></p>" & vbCrLf

    Set objSS = objDoc.createStyleState

    'Create the first text range.
    Set objRng = objDoc.body.createTextRange

    With objSS
        .gather objRng

        'Clear all formatting, including paragraph formatting.
        .ClearAllFormatting
        .Apply
    End With

    'Create the second text range.
    objDoc.body.insertAdjacentHTML "beforeend", "<p align=" & _
    """center"">"& _
    "<span style="background" & _
    "-color: #0000FF">" & "Clears text formatting; " & _
    "paragraph formatting remains</span></u></i></b></p>"

    Set objRng = objDoc.body.createTextRange
clearInterval Method

The `clearInterval` method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
clearRequest Method

The clearRequest method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
clearTextFormatting Method

Clears any character formatting associated with an element or text range.

expression.ClearTextFormatting

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

When you compare the `ClearAllFormatting` method with the `ClearTextFormatting` method, you notice that while the `ClearAllFormatting` method clears all formatting in the specified `IHTMLTxtRange` object, including paragraph formatting, the `ClearTextFormatting` method clears only formatting that modifies the appearance of text in the range.

Use the `Apply` method to apply the changes to the text range or element.
Example

The following example inserts formatted text into the specified document, creates an **IHTMLTxtRange** object that includes the entire document, and then removes all formatting associated with the range. Then it inserts similarly formatted text directly below the first text inserted, creates a second **IHTMLTxtRange** object that again includes the entire document, and removes only text formatting associated with the range. The first paragraph inserted has all formatting removed, including paragraph formatting; the second paragraph retains paragraph formatting but all text formatting is removed.

```vbscript
Sub ClearFormatting()
    Dim objSS As IFPStyleState
    Dim objDoc As FPHTMLDocument
    Dim objPara As FPHTMLParaElement
    Dim objRng As IHTMLTxtRange

    Set objDoc = ActiveDocument
    objDoc.body.innerHTML = "<p align="center">""<b><i><u>"& _ 
        "<span style="background-color: #0000FF">"Clears " & _ 
        "all formatting; paragraph is no longer centered" & _ 
        "</span></u></i></b></p>" & vbCrLf
    Set objSS = objDoc.createStyleState

    'Create the first text range.
    Set objRng = objDoc.body.createTextRange
    With objSS
        .gather objRng
        'Clears all formatting including paragraph formatting
        .ClearAllFormatting
        .Apply
    End With

    'Create the second text range.
    objDoc.body.insertAdjacentHTML "beforeend", "<p align=" & _ 
        ""center"">""<b><i><u>"& _ 
        "<span style="background-color: #0000FF">"Clears text formatting; " & _ 
        "paragraph formatting remains</span></u></i></b></p>"
    Set objRng = objDoc.body.createTextRange
```

With objSS
  .gather objRng

  'Clear only formatting that applies to text.
  .ClearTextFormatting
  .Apply
End With

Set objSS = Nothing
Set objDoc = Nothing
Set objRng = Nothing
End Sub
clearTimeout Method

The `clearTimeout` method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**click Method**

The **click** method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
clone Method

Returns an IFPStyleState object that represents a complete copy of a specified IFPStyleState object.

expression.Clone

expression Required. An expression that returns an IFPStyleState object.
Example

The following example inserts HTML into the specified document and creates two `IHTMLElement` objects, one for each paragraph element inserted. Then it creates an `IFPStyleState` object called objSS1 and adds font formatting properties. It then uses the `Clone` method to create a copy of the `IFPStyleState` object that is stored in the objSS2 variable, and modifies the background color of the `IFPStyleState` object called objSS2. Once the `IFPStyleState` objects are created, they are applied to the element object variables objElement1 and objElement2.

```vbnet
Sub CloneStyleState()
    Dim objDoc As FPHTMLDocument
    Dim objSS1 As IFPStyleState
    Dim objSS2 As IFPStyleState
    Dim objElement1 As IHTMLElement
    Dim objElement2 As IHTMLElement

    Set objDoc = Application.ActiveDocument

    objDoc.body.innerHTML = '<p>Line One</p>' & vbCrLf & _
    '<p>Line Two</p>'

    Set objElement1 = objDoc.all.tags("p").Item(0)
    Set objElement2 = objDoc.all.tags("p").Item(1)

    Set objSS1 = objDoc.createStyleState
    objSS1.fontStyle = "italic"

    Set objSS2 = objSS1.Clone
    objSS2.backgroundColor = vbBlue

    objSS1.applyToElement objElement1
    objSS2.applyToElement objElement2

    Set objElement2 = Nothing
    Set objElement1 = Nothing
    Set objSS2 = Nothing
    Set objSS1 = Nothing
    Set objDoc = Nothing
End Sub
```
close Method

The close method is not supported in Microsoft FrontPage and is therefore not available for programmatic access.
**collapse Method**

Moves the insertion point to the beginning or end of a range.

\[ expression\.collapse(start) \]

*expression*  Required. An expression that returns one of the objects in the Applies To list.

*start*  Optional Boolean. **True** moves the insertion point to the beginning of the text range. **False** moves the insertion point to the end of the text range. Default value is **True**.
commit Method

Executes a series of operations that was added to the undo stack of an \texttt{FPHTMLUndoTransaction} object. For example, if you create an \texttt{FPHTMLUndoTransaction} object, and then run a macro, the \texttt{commit} method allows the macro to continue or, if the macro is finished, the \texttt{commit} method prevents the rollback of the effects of the macro. In other words, the series of operations the macro performed cannot be undone.

\textit{expression}.\texttt{commit}

\textit{expression} \ An expression that returns an \texttt{FPHTMLUndoTransaction} object.
Example

Use the `createUndoTransaction` method of the `IFPDocument` object to create an `FPHTMLUndoTransaction` object. The `String` argument passed in with the `createUndoTransaction` method is added to the `Undo` command on the `Edit` menu when the `commit` method is called. The following example creates an `FPHTMLUndoTransaction` object illustrating the `abort` and `commit` methods.

```vba
Private Sub CreateUndoTransaction()
    Dim objDoc As FPHTMLDocument
    Dim objTansaction As FPHTMLUndoTransaction

    On Error GoTo CreateUndoTransactionError

    Set objDoc = ActiveDocument
    Set objTansaction = objDoc._CreateUndoTransaction("Last Macro")

    objDoc.body.insertAdjacentHTML "BeforeEnd", _
    "<b>Added by FP Programmability</b>"

    objTansaction.commit

ExitCreateUndoTransaction:
    Exit Sub

CreateUndoTransactionError:
    objTansaction.abort
    GoTo ExitCreateUndoTransaction
End Sub
```
commitChanges Method

The commitChanges method is not supported in Microsoft FrontPage and is therefore not available for programmatic access.
**compareEndPoints Method**

Compares an end point of an IHTMLTxtRange object with an end point of another IHTMLTxtRange object. Returns a Long that represents the comparison of the specified points.

expression.compareEndPoints(how, SourceRange)

**expression**  Required. An expression that returns an IHTMLTxtRange object.

**how**  Required String. Specifies how the two IHTMLTxtRange objects are to be compared. May be one of four values.

The String value of the how parameter can be one of the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartToStart</td>
<td>Compares the start of the specified IHTMLTxtRange object with the start of the IHTMLTxtRange object specified in the SourceRange parameter.</td>
</tr>
<tr>
<td>StartToEnd</td>
<td>Compares the start of the specified IHTMLTxtRange object with the end of the IHTMLTxtRange object specified in the SourceRange parameter.</td>
</tr>
<tr>
<td>EndToStart</td>
<td>Compares the end of the specified IHTMLTxtRange object with the start of the IHTMLTxtRange object specified in the SourceRange parameter.</td>
</tr>
<tr>
<td>EndToEnd</td>
<td>Compares the end of the specified IHTMLTxtRange object with the end of the IHTMLTxtRange object specified in the SourceRange parameter.</td>
</tr>
</tbody>
</table>

**SourceRange**  Required IHTMLTxtRange. The range with which to compare the initial range specified.
## Remarks

The **Long** value returned for the **compareEndPoints** method can be one of the following values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>The end point of the object is further to the right than the end point of the <strong>IHTMLTxtRange</strong> object specified in the <strong>SourceRange</strong> parameter.</td>
</tr>
<tr>
<td>0</td>
<td>The end point of the object is at the same location as the end point of the <strong>IHTMLTxtRange</strong> object specified in the <strong>SourceRange</strong> parameter.</td>
</tr>
<tr>
<td>1</td>
<td>The end point of the object is to the left of the end point of the <strong>IHTMLTxtRange</strong> object specified in the <strong>SourceRange</strong> parameter.</td>
</tr>
</tbody>
</table>
Example

The following example compares the selected range with the document and displays a message stating whether the selected range is at the beginning of the document or the end of the document.

Dim objDoc As IHTMLTxtRange
Dim objRange As IHTMLTxtRange

Set objDoc = ActiveDocument.body.createTextRange
Set objRange = ActiveDocument.Selection.createRange

If objRange.compareEndPoints("endtoend", objDoc) = 0 Then
    MsgBox "The selected text is at the end of the page."
ElseIf objRange.compareEndPoints("starttostart", objDoc) = 0 Then
    MsgBox "The selected text is at the beginning of the page."
Else
    MsgBox "The selected text is in the middle of the page."
End If
confirm Method

Displays a confirmation dialog box that contains an optional message as well as OK and Cancel buttons and returns a **Boolean** that represents the user's response. **True** indicates that the user clicked OK. **False** indicates that the user clicked Cancel.

```
expression.confirm(message)
```

- **expression** Required. An expression that returns one of the objects in the Applies To list.
- **message** Optional **String**. Specifies the message to display in the confirmation dialog box. If no value is provided, the dialog box does not contain a message.
Example

The following example displays a message to the user and captures their response in a variable.

Dim blnResponse As Boolean

blnResponse = ActiveDocument.parentWindow_.confirm("Are you sure you want to do that?")
contains Method

Returns **True** if the **IHTMLElement** object specified in the *pChild* argument is contained within the parent object.

`expression.contains(pChild)`

*expression*   Required. An expression that returns one of the objects in the Applies To list.

*pChild*   Required **IHTMLElement** object.
Remarks

Use the `contains` method to quickly determine whether an element is contained within another element.
Example

The following example uses the `contains` method to determine if the active element is contained within the expense form and, if it is, inserts a paragraph. This example assumes that the active document has a FORM element named "expense."

```vba
Sub AddParagraphToForm()
    Dim objForm As FPHTMLFormElement
    Set objForm = ActiveDocument.forms.Item("expense")
    If objForm.contains(ActiveDocument.activeElement) Then
        ActiveDocument.activeElement.insertAdjacentHTML "beforeend", "<p>This is a paragraph inside of a form.</p>
    End If
End Sub
```
create Method

The create method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
createCaption Method

Returns an IHTMLTableCaption object that represents a CAPTION element added to a TABLE element.

expression.createCaption

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

Use the `deleteCaption` method to remove a caption from a TABLE element.
Example

The following example creates a new `CAPTION` element in the first `TABLE` element in the active document.

```
Dim objTable As FPHTMLTable
Dim objCaption As FPHTMLTableCaption

Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objCaption = objTable.createCaption
```

createControlRange Method

The `createControlRange` method is not supported in Microsoft FrontPage and is therefore not available for programmatic access.
createElement Method

Creates a new IHTMLElement object that represents an HTML element based on the specified argument.

expression.createElement(eTag)

**expression**  Required. An expression that returns a FPHTMLDocument object.

**eTag**  Required. A String that represents the new HTML tag type.
Remarks

While you can use the `createElement` method to create any number of elements, you need to use the `InsertAdjacentHTML` method or the `InnerHTML` property to add the elements to a Web page, with the exception of the `AREA` element for which you would use the `Add` method.
Example

The following example creates a new P element (paragraph), specifies text for the new paragraph, and then inserts the new paragraph at the end of the active document.

Dim objPara As FPHTMLParaElement

Set objPara = ActiveDocument.CreateElement("P")
objPara.innerHTML = "{NewPara}.
ActiveDocument.body.innerHTML = ActiveDocument.body _
                   .innerHTML & objPara.outerHTML
createRange Method

Returns an IHTMLTxtRange object that represents a selection of text in a document.

expression.createRange

expression  Required. An expression that returns an IHTMLSelectionObject object.
Example

The following example creates a range from the currently selected text in the active document.

Dim objRange As IHTMLTxtRange
Set objRange = ActiveDocument.selection.createRange
createStyleLength Method

Returns an IFPStyleLength object that represents measurement properties for a style.

expression.createStyleLength

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example creates a new **IFPStyleLength** object.

```vbscript
Dim objState As IFPStyleState
Dim objLength As IFPStyleLength

Set objState = Application.ActiveDocument.createStyleState
Set objLength = objState.createStyleLength
```
createStyleSheet Method

The `createStyleSheet` method is not supported in Microsoft FrontPage and is therefore not available for programmatic access.
createStyleState Method

Returns an IFPStyleState object that represents the style properties associated with a specified text range.

expression.createStyleState

expression  Required. An expression that returns one of the objects in the Applies to list.
Remarks

Use the `gather` method to prepare an `IFPStyleState` object to use. Once you've applied the style attributes for the `IFPStyleState` object, use the `Apply` method to make the specified changes in the document.
Example

The following example specifies the font name and weight for text in the active document.

**Note** This example creates a FONT element with the **face** attribute specified and a B element around all text in the active document. Note that each section of text is treated separately. For example, opening and closing FONT and B tags are placed around the text in each paragraph in a document and around the text in each table cell. Therefore, if you have the paragraph <p>text</p> it will become <p><font face="Verdana"><b>text</b></font></p>. Also, if a paragraph contains no text, which includes paragraphs that contain nonprinting symbols such as &nbsp; but no other printable symbols, no elements are applied.

```vba
Dim objStyleState As IFPStyleState
Dim objRange As IHTMLTxtRange

Set objRange = Application.ActiveDocument.body.createTextRange
Set objStyleState = Application.ActiveDocument.createStyleState

With objStyleState
    .gather objRange
    .fontWeight = 700
    .fontFamily = "Verdana"
    .Apply
End With
```
createTextRange Method

Returns an IHTMLOLEditObject object that represents a range of text and elements in a document.

expression.createTextRange

expression  Required. An expression that returns one of the objects in the Applies To list.
**Example**

The following example searches the active document for the specified text, and then selects the text if it is found.

```vbscript
Dim objRange As IHTMLTxtRange
Dim blnFound As Boolean

Set objRange = ActiveDocument.body.createTextRange
blnFound = objRange.findText("search text")
If blnFound = True Then objRange.Select
```
createTFoot Method

Creates a TFOOT element inside of a TABLE element.

expression.createTFoot

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

Use the `deleteTFoot` method to remove a TFOOT element from a table.
Example

The following example creates a TFOOT element in the first table in the active document.

Dim objTable As FPHTMLTable
Dim objTHead As FPHTMLTableSection

Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objTHead = objTable.createTFooter
createTHead Method

Creates a THEAD element inside of a TABLE element.

expression.createTHead

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

Use the `deleteTHead` method to remove a THEAD element from a table.
Example

The following example creates a THEAD element in the first table in the active document.

Dim objTable As FPHTMLTable
Dim objTHead As FPHTMLTableSection

Set objTable = ActiveForm.all.tags("table").Item(0)
Set objTHead = objTable.createTHead
createUndoTransaction Method

Creates a new instance of an **FPHTMLUndoTransaction** object for the specified document.

`expression.createUndoTransaction(title)`

*expression* An expression that returns an **FPHTMLDocument**, **IFPDocument**, or **IHTMLDocument** object.

*title* Required **String**. The string that represents the title of the **FPHTMLUndoTransaction** object and appears on the **Undo** portion of the **Edit** menu.
Remarks

An **FPHTMLUndoTransaction** object allows you to track every action that occurs (after the undo transaction stack is created). You can then use the programming elements provided in the Page object model at run-time to track the actions of a macro.
Example

Use the `createUndoTransaction` method of the `FPHTMLDocument`, `IFPDocument`, and `IHTMLDocument` objects to create an `FPHTMLUndoTransaction` object. The `String` argument passed in with the `createUndoTransaction` method is added to the `Undo` command on the `Edit` menu when the `Commit` method is called. The following example creates an `FPHTMLUndoTransaction` object illustrating the `abort` and `Commit` methods.

```vba
Private Sub CreateUndoTransaction()
    Dim objDoc As FPHTMLDocument
    Dim objTransaction As FPHTMLUndoTransaction

    On Error GoTo CreateUndoTransactionError
    Set objDoc = ActiveDocument
    Set objTransaction = objDoc .CreateUndoTransaction("Last Macro")

    objDoc.body.insertAdjacentHTML "BeforeEnd", _
    "<b>Added by FP Programmability</b>"

    objTransaction.Commit

ExitCreateUndoTransaction:
    Exit Sub

CreateUndoTransactionError:
    objTransaction.abort
    GoTo ExitCreateUndoTransaction
End Sub
```
currentDataNode Method

Returns an IXMLDOMNode object (which is part of the Microsoft XML, version 2.0 object model interface) that represents the Extensible Markup Language (XML) node for the specified text range inside an XML Data View Web Part.

expression.currentDataNode(Range)

expression  Required. An expression that returns one of the objects in the Applies To list.

Range  Required IHTMLTxtRange. Specifies a range or selection of text and elements inside of an XML Data View Web Part.
**dataViewNodeSetIterator Method**

Returns an **IHTMLXSLElement** object that represents the first node set iterator Extensible Stylesheet Language (XSL) construct (e.g. xsl:apply-templates or xsl:for-each) in the specified range.

*expression*.`dataViewNodeSetIterator(Range)`

*expression* Required. An expression that returns one of the objects in the Applies To list.

*Range* Required **IHTMLTxtRange**. Specifies a range or selection of text and elements inside of a Data View Web Part.
**dataViewTemplateBodyExpansion**

Method

Returns an **IHTMLXSLElement** object that represents the first parent template or Extensible Stylesheet Language (XSL) xsl:for-each construct of a Data View Web Part.

`expression.dataViewTemplateBodyExpansion(Range)`

*expression*  Required. An expression that returns one of the objects in the Applies To list.

**Range**  Required **IHTMLTxtRange**. Specifies a range or selection of text and elements inside of a Data View Web Part.
deleteCaption Method

Removes a CAPTION element in a TABLE element.

expression.deleteCaption

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

Use the `createCaption` method to add a caption to a TABLE element.
Example

The following example deletes the caption in the first table in the active document.

Dim objTable As FPHTMLTable

Set objTable = ActiveDocument.all.tags("table").Item(0)

objTable.deleteCaption
**deleteCell Method**

Deletes a cell (TD element) from a row in a table.

```
expression.deleteCell(index)
```

*expression*  Required. An expression that returns one of the objects in the Applies To list.

*index*  Optional `Long`. Specifies the cell to be deleted from the table row. If omitted, deletes the last cell in the row.
DeleteMemberByDispID Method

The **DeleteMemberByDispID** method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
DeleteMemberByName Method

The **DeleteMemberByName** method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**deleteTFoot Method**

Removes a TFOOT element from a TABLE element.

`expression.deleteTFoot`

*expression* Required. An expression that returns one of the objects in the Applies To list.
Remarks

Use the `createTFoot` method to add a TFOOT element to a table.
Example

The following example removes the TFOOT element, if one exists, in the first table in the active document.

Dim objTable As FPHtmlTable
Set objTable = ActiveDocument.all.tags("table").Item(0)
objTable.deleteTFoot
deleteTHead Method

Removes a THEAD element from a TABLE element.

expression.deleteTHead

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

Use the `createTHead` method to add a THEAD element to a table.
**Example**

The following example removes the THEAD element, if one exists, in the first table in the active document.

```vba
Dim objTable As FPHTMLTable
Set objTable = ActiveDocument.all.tags("table").Item(0)
objTable.deleteTHEad
```
doReadRequest Method

The `doReadRequest` method is not supported in Microsoft FrontPage and is therefore not available for programmatic access.
doRequest Method

The doRequest method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
doWriteRequest Method

The doWriteRequest method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**duplicate Method**

Returns an **IHTMLTxtRange** that represents a duplicate of another range.

*expression.duplicate*

*expression*  Required. An expression that returns an **IHTMLTxtRange** object.
Example

The following example creates a duplicate of the current selection in the active document.

Dim objRange As IHTMLTxtRange
Dim objDuplicate As IHTMLTxtRange

Set objRange = ActiveDocument.selection.createRange
Set objDuplicate = objRange.Duplicate
**dynamicTemplate Property**

Returns a **String** that represents the path and file name of a Dynamic Web Template. An empty **String** indicates that no Dynamic Web Template is attached. Setting the **dynamicTemplate** property to an empty string detaches an attached Dynamic Web Template from the page.

**expression.dynamicTemplate**

**expression**  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example updates the Dynamic Web Template if the specified Dynamic Web Template is attached to the active document. If it isn't already attached, the code attaches it.

Dim objState As DynamicTemplateState

Set objState = Application.CreateDynamicTemplateState

If ActiveDocument.DynamicTemplate = "template.dwt" Then
    ActiveDocument.UpdateDynamicTemplate objState
End If
elementFromPoint Method

Returns the IHTMLElement object that corresponds to the specified page coordinates.

expression.elementFromPoint(x, y)

expression Required. An expression that returns one of the objects in the Applies To list.

x Required. A Long that represents a specified point on the horizontal axis from the left edge of the page.

y Required. A Long that represents a specified point on the vertical axis from the top edge of the page.
Remarks

The coordinates correspond to page coordinates (0,0 is the top-left corner of the page). For the `elementFromPoint` method to exhibit the expected behavior, the object or element located at position (x, y) must support and respond to mouse events.
Example

The following example sets the active element using the `elementFromPoint` method.

Dim objElement As IHTMLElement

Set objElement = ActiveDocument.elementFromPoint(
    ActiveDocument.activeElement.offsetTop,
    ActiveDocument.activeElement.offsetWidth)
empty Method

Deselects the current selection.

`expression.empty`

`expression` Required. An expression that returns an `IHTMLSelectionObject` object.
**Example**

The following example creates a selection object from the currently selected text in the active document and then deselects the text.

```vba
Dim objSel As IHTMLSelectionObject
Set objSel = ActiveDocument.selection
objSel.empty
```
execCommand Method

Executes a specified action on the current document, current selection, or the specified range, and then returns a Boolean that indicates whether the command completed successfully.

expression.execCommand(cmdID, showUI, value)

expression  Required. An expression that returns one of the objects in the Applies To list.

cmdID  Required. A String that represents the command to execute.

The cmdID parameter can be one of the following String values:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BackColor</td>
<td>Sets the background color of the current selection. The color name of the background color. For available color Table.</td>
</tr>
<tr>
<td>Copy</td>
<td>Copies the current selection to the Clipboard.</td>
</tr>
<tr>
<td>CreateBookmark</td>
<td>Creates a bookmark anchor around the selected text.</td>
</tr>
</tbody>
</table>

Note  You must use a color name and not a Web-safe RGB color value, such as #663399.

ActiveDocument.execCommand "backcolor", False, "yellow"

ActiveDocument.execCommand "copy"

ActiveDocument.execCommand "copy"
ActiveDocument.execCommand "createbookmark", False, "newbookmark"

**CreateLink**

Inserts a hyperlink on the current selection or displays a dialog box enabling the user to specify a URL to insert as a hyperlink on the current selection. The URL of the hyperlink must be specified.

The following code creates a hyperlink to the Microsoft website.

ActiveDocument.execCommand "createlink", False, "http://www.microsoft.com"

**Cut**

Cuts the current selection and copies it to the Clipboard.

The following code cuts the selected text and places a copy in the Clipboard.

ActiveDocument.execCommand "cut"

**Delete**

Deletes the current selection without storing a copy of it.

The following code deletes the selected text.

ActiveDocument.execCommand "delete"

**FontSize**

Sets the font size for the current selection. The **value** parameter must specify the size of the font to use.

The following code specifies a font size for the selected text, which is 115% of the size of the font setting for the parent element.

ActiveDocument.execCommand "fontsize", False, "115%"

**FontName**

Sets the font of the selected text. The **value** parameter must specify the name of the font to use.

The following code specifies the Stencil font for the selected text.

ActiveDocument.execCommand "fontname", False, "stencil"

**Note** If the font name specified in the **value** parameter is not available on the user's machine, the selected text will display in the browser in the default font.

Sets the color of the selected text. The **value** parameter must specify the color name of the font to use.

The following code sets the color of the selected text to red.

ActiveDocument.execCommand "fontcolor", False, "red"
the text color. For available color names, see the HTML Color Table.

The following code sets the color of the selected text to red:

```javascript
ActiveDocument.execCommand "forecolor", False, "red"
```

**Note** You must use a color name and not a Web-safe RGB color value, such as #663399.

Indents the selected text by inserting a BLOCKQUOTE element and any surrounding text that belongs to the same parent element.

The following code indents the selected text and related text:

```javascript
ActiveDocument.execCommand "indent"
```

Inserts a BUTTON element at the insertion point. If text is selected, the new element overwrites the selected text. Use the `value` parameter to specify a new element.

The following code inserts a BUTTON element with an "newbutton" at the insertion point.

```javascript
ActiveDocument.execCommand "insertbutton", False, "newbutton"
```

Inserts a FIELDSET element at the insertion point. If text is selected, the new element overwrites the selected text. Use the `value` parameter to specify a new element.

The following code inserts a FIELDSET element with an id attribute value of "newfield" at the insertion point.

```javascript
ActiveDocument.execCommand "insertfieldset", False, "newfield"
```

Inserts an HR element at the insertion point. If text is selected, the new element overwrites the selected text.

The following code inserts an HR element at the insertion point:

```javascript
ActiveDocument.execCommand "inserthorizontalrule"
```
overwrites the selected text. Use the `value` parameter to in the inline frame.

**InsertIFrame**

The following code inserts an IFRAME element at the attribute of "form.htm".

```javascript
ActiveDocument.execCommand "insertiframe", False, "form.htm"
```

Inserts an IMG element at the insertion point. If text is overwrites the selected text. Use the `value` parameter to

**InsertImage**

The following code inserts the chelan.jpg image file int point.

```javascript
ActiveDocument.execCommand "insertimage", False, "chelan.jpg"
```

Inserts an INPUT element of type "button" at the insert new element overwrites the selected text. Use the `value` attribute for the new element.

**InsertInputButton**

The following code inserts a new input button with an `i` at the insertion point.

```javascript
ActiveDocument.execCommand "insertinputbutton"
```

Inserts an INPUT element of type "checkbox" at the ins the new element overwrites the selected text. Use the `value` attribute for the new element.

**InsertInputCheckbox**

The following code inserts a new check box with an `id "newcheckbox" at the insertion point.

```javascript
ActiveDocument.execCommand "insertinputcheckbox"
```

Inserts an INPUT element of type "fileupload" at the in the new element overwrites the selected text. Use the `value` attribute for the new element. Overwrites a file upload c

**InsertInputFileUpload**

The following code inserts an INPUT element of type "

```javascript
ActiveDocument.execCommand "insertinputfileupload"
```
InsertInputHidden

Inserts an INPUT element of type "hidden" at the insertion point. The new element overwrites the selected text. Use the value attribute for the new element.

The following code inserts a hidden INPUT element at the insertion point:

ActiveDocument.execCommand "insertinputhidden"

InsertInputPassword

Inserts an INPUT element of type "password" at the insertion point. The new element overwrites the selected text. Use the value attribute for the new element.

The following code inserts a text box for entering passwords at the insertion point:

ActiveDocument.execCommand "insertinputpassword"

InsertInputReset

Inserts an INPUT element of type "reset" at the insertion point. The new element overwrites the selected text. Use the value attribute for the new element.

The following code inserts a Reset button at the insertion point:

ActiveDocument.execCommand "insertinputreset"

InsertInputSubmit

The following code inserts a Submit button at the insertion point:

ActiveDocument.execCommand "insertinputsubmit"

InsertInputText

Inserts an INPUT element of type "submit" at the insertion point. The new element overwrites the selected text. Use the value parameter to specify a new element.

The following code inserts a one-line text box at the insertion point:

ActiveDocument.execCommand "insertinputtext"

Inserts a MARQUEE element at the insertion point. If text is selected, the new element overwrites the selected text.

The following code inserts a MARQUEE element at the insertion point:

ActiveDocument.execCommand "insertinputhidden"
InsertMarquee

The following code inserts a new MARQUEE element

```javascript
ActiveDocument.execCommand "insertmarquee"
```

If text is selected, toggles the text selection between items in paragraphs. If no text is selected, inserts an OL element.

InsertOrderedList

The following inserts an ordered list at the insertion point.

```javascript
ActiveDocument.execCommand "insertorderedlist"
```

InsertParagraph

The following inserts a new paragraph at the insertion point.

```javascript
ActiveDocument.execCommand "insertparagraph", True, "newpara"
```

InsertSelectDropdown

The following inserts a new drop-down list at the insertion point.

```javascript
ActiveDocument.execCommand "insertselectdropdown", True, "newdropdownlist"
```

InsertSelectListbox

The following code inserts a list box at the insertion point.

```javascript
ActiveDocument.execCommand "insertselectlistbox", True, "newlistbox"
```

InsertTextArea

The following code inserts a multiple-line text box at the insertion point.

```javascript
ActiveDocument.execCommand "inserttextarea", True, "newtextarea"
```
ActiveDocument.execCommand "inserttextarea", True, "newtextarea"

If text is selected, toggles the text selection between items in paragraphs. If no text is selected, inserts a UL element.

InsertUnorderedList

The following inserts a bulleted list at the insertion point or a bulleted list.

ActiveDocument.execCommand "insertunorderedlist"

Toggles the current selection between italic and nonitalic.

Italic

The following toggles italics for the selected text.

ActiveDocument.execCommand "italic"

Centers the selected text and any surrounding text that belongs to the same parent element. If the affected text alignment is already centered, then this command returns the text to the default alignment.

JustifyCenter

The following code centers the selected text and any related text.

ActiveDocument.execCommand "justifycenter"

Left aligns the selected text and any surrounding text that belongs to the same parent element. If the affected text alignment is already left aligned, then this command returns the text alignment to the default alignment.

JustifyLeft

The following code left aligns the selected text and any related text.

ActiveDocument.execCommand "justifyleft"

Right aligns the selected text and any surrounding text that belongs to the same parent element. If the affected text alignment is already right aligned, then this command returns the text alignment to the default alignment.

JustifyRight

The following code right aligns the selected text and any related text.

ActiveDocument.execCommand "justifyright"

Decreases by one increment the indentation of the selected text. The BLOCKQUOTE element from around the selected text...
Outdent

belongs to the same parent element. If there are no surrounding elements, this command does nothing.

Decreases by one the indentation of the selected text and any related text.

`ActiveDocument.execCommand "outdent"`

Paste

Overwrites the contents of the Clipboard on the current position or overwrites any selected text.

The following code pastes the contents of the Clipboard into the document at the cursor position.

`ActiveDocument.execCommand "paste"`

RemoveFormat

Removes the font and character formatting from the current selection.

The following code removes all formatting from the selected text.

`ActiveDocument.execCommand "removeformat"`

SelectAll

Selects the entire document.

The following code selects all contents in the active document.

`ActiveDocument.execCommand "selectall"`

UnBookmark

Removes any bookmark from the current selection. If the selected text is not affected by a bookmark anchor, the bookmark anchor will close before and reopen after the selected text. If the selected text is all the text affected by a bookmark anchor, the entire bookmark is removed from the document. If multiple bookmarks are included in the selection, then all bookmarks are removed from the selection.

The following code removes all bookmarks from the selected text.

`ActiveDocument.execCommand "unbookmark"`

Underline

Toggles the current selection between underlined and not underlined.

The following code toggles underlining for the selected text.

`ActiveDocument.execCommand "underline"`
Unlink

Removes any hyperlinks from the current selection. If the selected text is all the text affected by a hyperlink, the entire hyperlink is removed from the document. If multiple hyperlinks are affected by a hyperlink, the hyperlink will close before and reopen after the selected text. It the selected text is all the text affected by a hyperlink, the entire hyperlink is removed from the document. If multiple hyperlinks are included in the selection, then all hyperlinks are removed from the selection.

The following removes all hyperlinks from the selected text:

ActiveDocument.execCommand "unlink"

Unselect

Clears the current selection.

The following code unselects the text in the active document and places the cursor at the beginning of the document:

ActiveDocument.execCommand "unselect"

**showUI** Optional. *True* displays the dialog box associated with the command, if there is a corresponding dialog box. The default value is *False*.

**value** Optional. A *Variant* that specifies any additional arguments associated with the *cmdID* parameter. For example, if the *cmdID* parameter is specifies the BackColor command, then the *value* parameter contains the color name for the background color of the selection.
Remarks

If the command specified in the `CmdID` parameter sets a value, then the `value` parameter is required to complete the command.

If the `value` parameter contains a string that represents the value of the `id` attribute for an element and there already exists an element with the same `id` attribute value, the `execCommand` method will increment the `id` attribute string. For example, if you specify "newelement" for the `value` parameter and there is already an element with the `id` attribute of 'newelement", the new element that the `execCommand` parameter adds will be "newelement0". If there are several elements with the same `id` attribute, for example, "newelement", "newelement0", and "newelement1", then the newly added element will increment up to the next available number, which in this case would be "newelement2".

Many of the commands used by the `execCommand` method require that the user select text in the page window in Microsoft FrontPage. If no text is selected and the command requires selected text, the `execCommand` method will fail and display an error.
execCommandShowHelp Method

The execCommandShowHelp method is not supported in Microsoft FrontPage and is therefore not available for programmatic access.
execScript Method

The execScript method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
Show All
expand Method

Expands the range so that partial units are completely contained and returns a Boolean that represents whether the method was successful.

expression.expand(Unit)

description

expression  Required. An expression that returns one of the objects in the Applies To list.

Unit  Required String. Specifies the number of units to move in the range. Can be one of the following String values.

- character Expands a character.
- word Expands a word. A word is a collection of characters terminated by a space or another white-space character, such as a tab.
- sentence Expands a sentence. A sentence is a collection of words terminated by an ending punctuation character, such as a period.
- textedit Expands to enclose the entire range.
Find Method

Executes a search and returns a **Boolean** that represents whether the search was successful.

`expression.Find(info, limits, startRange)`

*expression* Required. An expression that returns one of the objects in the Applies To list.

*info* Required **SearchInfo**. The **SearchInfo** object that contains the search information.

*limits* Optional **Variant**. Specifies an **IHTMLTxtRange** object that represents the range of text on which to perform the search. If omitted, search is performed on the entire document from the specified starting point.

*startRange* Optional **Variant**. Specifies an **IHTMLTxtRange** object that represents the position in the document from which to start the search. If omitted, search starts at the beginning of the document.
Remarks

If the **Action** property is set to `fpSearchFindText` or `fpSearchReplace` text, the **Find** property defines the text for which to search; if the **Action** property is set to `fpSearchFindTag`, then the **Find** property defines the name of the HTML tag for which to search.
**Example**

The following example selects the next occurrence of the P element if one is found in the active document.

```vba
Dim objSearch As SearchInfo
Dim blnFound As Boolean
Dim objRange As IHTMLTxtRange

Set objSearch = Application.CreateSearchInfo
objSearch.Find = "p"
objSearch.Action = fpSearchFindTag

Set objRange = Application.ActiveDocument.selection.createRange
blnFound = Application.ActiveDocument.Find(objSearch, Nothing, objRange)
If blnFound = True Then objRange.Select
```
**findByID Method**

Returns an `FPLayoutRegion` object that represents the cell formatting table specified by using the ival parameter.

`expression.findByID(ival)`

*expression*  Required. An expression that returns an `FPLayoutTablesAndCells` collection.

*ival*  Required `Long`. Specifies the value of the `mscellformattingtableid` attribute as it appears in the post-comment tag, for example, `<-- MSCellFormattingTableID="ID" -->`. 
Example

The following example returns the specified layout cell.

Dim objLayout As FPLayoutRegion

Set objLayout = ActiveDocument.layoutTablesAndCells.findById(1)
**findText Method**

Searches for text in the document, and positions the start and end points of the range to encompass the search string and returns a **Boolean** that represents whether the search was successful.

*expression*.findText(*String*, *Count*, *flags*)

*expression*  Required. An expression that returns one of the objects in the Applies To list.

*String*  Required *String*. Specifies the text to find.

*Count*  Optional *Long*. Specifies the number of characters to search from the starting point of the range. A positive integer indicates a forward search; a negative integer indicates a backward search.

*flags*  Optional *Long*. Specifies one or more of the following *flags* to indicate the type of search:

2 Matches whole words only.
4 Matches case.
Example

The following example searches the active document for the specified text, and then selects the text if it is found.

```
Dim objRange As IHTMLTxtRange
Dim blnFound As Boolean

Set objRange = ActiveDocument.body.createTextRange
blnFound = objRange.findText("search text")
If blnFound = True Then objRange.Select
```
focus Method

The focus method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
forward Method

The **forward** method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
gather Method

Gathers the style properties from the specified IHTMLTxtRange object into the specified IFPStyleState object.

`expression.gather(range)`

**expression** Required. An expression that returns an IFPStyleState object.

**range** Required IHTMLTxtRange object. The text range to be associated with the IFPStyleState object.
Example

The following example inserts a formatted paragraph into the current document, then creates an \texttt{IFPStyleState} object and an \texttt{IHTMLTxtRange} object and uses the \texttt{gather} method of the \texttt{IFPStyleState} object (stored in the \texttt{objSS} variable) to associate the style properties in the \texttt{IHTMLTxtRange} object (stored in the \texttt{objRng} variable). The example then clears existing formatting in the \texttt{objSS} variable and applies a background color.

```vba
Sub GatherAndClearStyleProperties()
    Dim objSS As IFPStyleState
    Dim objDoc As FPHTMLDocument
    Dim objRng As IHTMLTxtRange

    Set objDoc = ActiveDocument

    objDoc.body.innerHTML = "<p><b><i><u>Heading 1</u></i></b></p>
    Set objSS = objDoc.createStyleState
    Set objRng = objDoc.body.createTextRange

    With objSS
        .gather objRng
        .ClearAllFormatting
        .Apply
        .backgroundColor = vbBlue
        .Apply
    End With

    Set objSS = Nothing
    Set objDoc = Nothing
    Set objRng = Nothing
End Sub
```
gatherFromElement Method

Gathers the style properties from the specified IHTMLElement object into the specified IFPStyleState object.

expression.GatherFromElement(element)

expression  Required. An expression that returns one of the objects in the Applies To list.

element  Required IHTMLElement object. The element associated with a given style.
Example

The following example inserts two heading elements and gathers the style properties from the first heading element stored in the objHeading1 variable. It then creates an IFPStyleState object, which is stored in the objSS variable, modifies the style properties, and then applies the modified style properties to the element stored in the objHeading2 variable.

Sub ApplyStyleToElement()
    Dim objSS As IFPStyleState
    Dim objHeading1 As IHTMLElement
    Dim objHeading2 As IHTMLElement

    ActiveDocument.body.innerHTML = "<H1>This is line 1</H1>" _ & vbCrLf & "<H1>This is line 2</H1>"

    Set objHeading1 = ActiveDocument.all.tags("h1").Item(0)
    Set objHeading2 = ActiveDocument.all.tags("h1").Item(1)

    Set objSS = ActiveDocument.createStyleState

    With objSS
        .GatherFromElement objHeading1
        .fontFamily = "Arial"
        .Color = vbRed
        .textAlign = "right"
        .setProperty "background-color", vbBlue
        .textTransform = "uppercase"
        .applyToElement objHeading2
    End With

    Set objSS = Nothing
    Set objHeading1 = Nothing
    Set objHeading2 = Nothing
End Sub
getAttributes Method

As it applies to the `FPLayoutRegion` object

Returns a `Variant` that represents the value of the specified attribute.

`expression.getAttribute(attr)`

- `expression` Required. An expression that returns an `FPLayoutRegion` object.
- `attr` Required `FpLayoutCellAttributes`. Specifies the attribute.

`FpLayoutCellAttributes` can be one of the following `FpLayoutCellAttributes` constants.

- `border_bottom_color`
- `border_bottom_size`
- `border_foot_color`
- `border_foot_size`
- `border_head_color`
- `border_head_size`
- `border_left_color`
- `border_left_size`
- `border_right_color`
- `border_right_size`
- `border_top_color`
- `border_top_size`
- `cellFomatingID`
- `content_color`
- `content_html`
- `content_padding`
- `content_type`
- `content_valign`
As it applies to all other objects in the Applies To list

Returns a **Variant** that represents the value of the specified attribute.

```plaintext
expression.getAttribute(strAttributeName, lFlags)
```

**expression**    Required. An expression that returns one of the objects in the Applies To list.

**strAttributeName**    Required **String**. The name of the attribute.

**lFlags**    Optional **Variant**. The value of the object.
**Example**

As it applies to the **FPHTML.Body** object

The following example sets the **position** attribute for the body of the active document.

Dim strAttribute As String

strAttribute = ActiveDocument.body_.Style.getAttribute("position")
**getBookmark Method**

Returns a **String** that represents the start and end positions of an **IHTMLTxtRange** object.

`expression.getBookmark`

*expression* Required. An expression that returns one of the objects in the Applies To list.
Example

The following example inserts two paragraphs into the active document and then modifies the contents of each paragraph.

Sub MoveToBookmarkMethod2()
    Dim objRange As IHTMLTxtRange
    Dim strBookmark As String

    ActiveDocument.body.innerHTML = "<p>Line one</p><p>Line two</p>"

    Set objRange = ActiveDocument.body.createTextRange

    objRange.moveToElementText ActiveDocument.body.all.tags("p").Item(0)
    strBookmark = objRange.getBookmark

    With objRange
        .moveToElementText ActiveDocument.body.all.tags("p").Item(1)
        .Text = "Changed Line 2"
        .moveToBookmark strBookmark
        .Text = "Changed line 1"
        .Select
    End With

    Set objRange = Nothing
End Sub
getBorderColor Method

Returns a String that represents the border color for a layout cell or cell formatting table.

`expression.getBorderColor(pszType)`

- `expression` Required. An expression that returns an `FPLayoutRegion` object.
- `pszType` Required String. Specifies which border. Can be one of the following String values: "all", "top", "bottom", "left", "right", "header" or "footer".
Remarks

Use the `setBorderColor` method to set the border color for a layout cell or cell formatting table.
**getBorderSize Method**

Returns an **Integer** that represents the thickness of the border for a layout cell or cell formatting table.

`expression.getBorderSize(pszType)`

*expression*  Required. An expression that returns an **FPLayoutRegion** object.

*pszType*  Required **String**. Specifies which border. Can be one of the following **String** values: "all", "top", "bottom", "left", "right", "header" and "footer".
Remarks

Use the `setBorderSize` method to set the thickness of the border for a layout cell or cell formatting table.
getBotAttribute Method

Returns a **Variant** that represents the specified **strAttributeName** attribute for a specified object, which represents a Microsoft FrontPage component inserted into a Web page.

*expression*.getBotAttribute(*strAttributeName*)

*expression* An expression that returns one of the objects in the Applies To list.

*strAttributeName* Required **String**. The string that represents the name of the attribute.
Example

This example inserts the FrontPage Search component, and then uses the `getBotAttribute` method to return the inserted object.

```vbscript
Private Sub AccessSearchComponent()
    Dim objComponent As FPHTMLFrontPageBotElement
    Dim objBody As FPHTMLBody
    Dim strHTML As String
    Dim objWindow As PageWindow

    strHTML = "<!--webbot bot=""Search"" s-index=""All"" & _
        " s-fields s-text=""Search for:"" & _
        " i-size=""20"" s-submit=""Start Search"" & _
        " s-clear=""Reset"" s-timestampformat=""%m/%d/%y"" & _
        " tag=""BODY"" -->"

    Set objBody = ActivePageWindow.Document.body
    Set objWindow = ActivePageWindow

    objBody.insertAdjacentHTML "BeforeEnd", strHTML
    Set objComponent = objWindow.Document.all.tags("webbot").Item(0)

    MsgBox objComponent.getBotAttribute("s-submit")
    objComponent.setBotAttribute "s-submit", "new item"

    MsgBox objComponent.getBotAttribute("s-submit")
    objComponent.removeBotAttribute "s-submit"

    MsgBox objComponent.getBotAttribute("s-submit")
End Sub
```
**getContentTypeColor** Method

Returns a **String** that represents the background color of a layout cell or cell formatting table.

*expression*.**getContentTypeColor**

*expression*  Required. An expression that returns an **FPLayoutRegion** object.
Remarks

Use the `setContentColor` method to set the background color of a layout cell or cell formatting table.
**getContentPaneHtml Method**

Returns a **String** that represents the HTML text inside a layout cell or the content layout cell of a cell formatting table.

**expression**.getContentPaneHtml

**expression**  Required. An expression that returns an **FPLayoutRegion** object.
Remarks

Use the `setContentHtml` method to set the HTML text inside a layout cell or the content layout cell of a cell formatting table.
**getContentViewPadding Method**

Returns an **Integer** that represents the cell padding for a layout cell or the content layout cell of a cell formatting table.

*expression.getContentViewPadding*

*expression*  Required. An expression that returns an **FPLayoutRegion** object.
Remarks

Use the `setContentPadding` method to set the cell padding for a layout cell or the content layout cell of a cell formatting table.
**getContentType Method**

Returns a `String` that represents the type of content that is inside a layout cell or cell formatting table. This `String` is enclosed in quotes in the post-comment tag commonly used in the layout templates; for example, `<!-- MSCellType="stringvalue" -->`.

`expression.getContentType`

`expression` Required. An expression that returns an `FPLayoutRegion` object.
Remarks

Use the `setContentType` method to set the type of content that is inside a layout cell or cell formatting table.
**getContentViewAlign Method**

Returns a `String` that represents the vertical alignment for text inside a layout cell or cell formatting table.

`expression.getContentViewAlign`

`expression` Required. An expression that returns an `FPLayoutRegion` object.
Remarks

Use the `setContentVAlign` method to set the vertical alignment for text inside a layout cell or cell formatting table.
getCorner Method

Returns a String that represents the type of corner border for a specified corner in a layout cell or cell formatting table.

\textit{expression.getCorner} (pszType)

\textit{expression} Required. An expression that returns an \texttt{FPLayoutRegion} object.

\textit{pszType} Required String. Specifies which corner. Can be one of the following String values: "top-left", "top-right", "bottom-left", "bottom-right", or "all".
Remarks

Use the `setCorner` method to set the type of corner border for a specified corner or for all corners in a layout cell or cell formatting table.
getCornerBorderColor Method

Returns a `String` that represents the border color of a specified corner border for a layout cell or cell formatting table.

\[ \text{expression}.\text{getCornerBorderColor}(\text{pszType}) \]

**expression**  Required. An expression that returns an `FPLayoutRegion` object.

**pszType**  Required `String`. Specifies which corner. Can be one of the following `String` values: "top-left", "top-right", "bottom-left", "bottom-right", or "all".
Remarks

Use the `setCornerBorderColor` method to set the color of a corner border for a layout cell or cell formatting table.
getCornerColor Method

Returns a String that represents the color of a specified corner in a layout cell or cell formatting table.

expression.getCornerColor(pszType)

expression Required. An expression that returns an FPLayoutRegion object.

pszType Required String. Specifies which corner. Can be one of the following String values: "top-left", "top-right", "bottom-left", "bottom-right", or "all".
Remarks

Use the `setCornerColor` method to set the color of the corner border in a layout cell or cell formatting table.
getCornerHeight Method

Returns an **Integer** that represents the height of a specified corner in a layout cell or cell formatting table.

```plaintext
expression.getCornerHeight(pszType)
```

**expression** Required. An expression that returns an **FPLayoutRegion** object.

**pszType** Required **String**. Specifies which corner. Can be one of the following **String** values: "top-left", "top-right", "bottom-left", "bottom-right", or "all".
Remarks

Use the `setCornerHeight` method to set the height of a corner in a layout cell or cell formatting table.
**getCornerImageUrl Method**

Returns a **String** that represents the path and file name for an image to display for a specified corner in a layout cell or cell formatting table.

\[ expression.getCornerImageUrl(pszType) \]

**expression**  Required. An expression that returns an **FPLayoutRegion** object.

**pszType**  Required **String**. Specifies which corner. Can be one of the following **String** values: "top-left", "top-right", "bottom-left", "bottom-right", or "all".
Remarks

Use the `setCornerImageUrl` method to set the path and file name for an image to display for a specified corner in a layout cell or cell formatting table.
**getCornerWidth Method**

Returns an **Integer** that represents the width of a specified corner in a layout cell or cell formatting table.

*expression*.getCornerWidth(*pszType*)

*expression*  Required. An expression that returns an **FPLayoutRegion** object.

*pszType*  Required **String**. Specifies which corner. Can be one of the following **String** values: "top-left", "top-right", "bottom-left", "bottom-right", or "all".
Remarks

Use the `setCornerWidth` method to set the width of a corner in a layout cell or cell formatting table.
GetDispID Method

The **GetDispID** method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
getFooterColor Method

Returns a String that represents the color of the footer for a layout cell or cell formatting table.

expression.getFooterColor

expression  Required. An expression that returns an FPLayoutRegion object.
Remarks

Use the `setFooterColor` method to set the color of the footer for a layout cell or cell formatting table.
getFooterHtml Method

Returns a String that represents the HTML text inside a footer for a layout cell or cell formatting table.

$expression.getFooterHtml$

$expression$  Required. An expression that returns an FPLayoutRegion object.
Remarks

Use the `setFooterHtml` method to set the HTML text inside a footer for a layout cell or cell formatting table.
getFooterPadding Method

Returns an **Integer** that represents the cell padding in a footer for a layout cell or cell formatting table.

*expression*.getFooterPadding

*expression*  Required. An expression that returns an **FPLayoutRegion** object.
Remarks

Use the `setFooterPadding` method to set the cell padding in a footer for a layout cell or cell formatting table.
getFooterSize Method

Returns an **Integer** that represents the height of the footer for a layout cell or cell formatting table. A zero value indicates no footer.

`expression.getFooterSize`

`expression` Required. An expression that returns an **FPLayoutRegion** object.
Remarks

Use the `setFooterSize` method to set the height of the footer for a layout cell or cell formatting table.
getFooterVAlign Method

Returns a **String** that represents the vertical alignment for text inside a footer for a layout cell or cell formatting table.

*expression*.getFooterVAlign

*expression* Required. An expression that returns an **FPLayoutRegion** object.
Remarks

Use the `setFooterVAlign` method to set the vertical alignment for text inside a footer for a layout cell or cell formatting table.
**getHeaderColor Method**

Returns a **String** that represents the color of the header for a layout cell or cell formatting table.

*expression.getHeaderColor*

*expression*  Required. An expression that returns an [FPLayoutRegion](#) object.
Remarks

Use the `setHeaderColor` method to set the color of the header for a layout cell or cell formatting table.
**getHeaderHtml Method**

Returns a **String** that represents the HTML text inside a header for a layout cell or cell formatting table.

`expression.getHeaderHtml`

`expression`  Required. An expression that returns an **FPLayoutRegion** object.
Remarks

Use the `setHeaderHtml` method to set the HTML text inside a header for a layout cell or cell formatting table.
getHeaderPadding Method

Returns an **Integer** that represents the cell padding in a header for a layout cell or cell formatting table.

`expression.getHeaderPadding`

`expression`  Required. An expression that returns an `FPLayoutRegion` object.
Remarks

Use the `setHeaderPadding` method to set the cell padding in a header for a layout cell or cell formatting table.
getHeaderSize Method

Returns an **Integer** that represents the height of the header for a layout cell or cell formatting table.

`expression.getHeaderSize`

`expression` Required. An expression that returns an **FPLayoutRegion** object.
Remarks

Use the `setHeaderSize` method to set the height of the header for a layout cell or cell formatting table.
**getHeaderVAlign Method**

Returns a String that represents the vertical alignment for text inside a header for a layout cell or cell formatting table.

`expression.getHeaderVAlign`

`expression`  Required. An expression that returns an **FPLLayoutRegion** object.
Remarks

Use the `setHeaderVAlign` method to set the vertical alignment for text inside a header for a layout cell or cell formatting table.
**getHeight Method**

Returns an **Integer** that represents the height of a layout cell or cell formatting table.

*expression*.**getHeight**

**expression**  Required. An expression that returns an **FPLayoutRegion** object.
Remarks

Use the `setHeight` method to set the height of a layout cell or cell formatting table.
getMargin Method

Returns an **Integer** that represents the margin settings for a layout cell or cell formatting table.

```
expression.getMargin(pszType)
```

*expression* Required. An expression that returns an **FPLayoutRegion** object.

*pszType* Required **String**. Specifies which margin. Can be one of the following **String** values: "top", "bottom", "left", or "right".
Remarks

Use the `setMargin` method to set the margin settings for a layout cell or cell formatting table.
GetMemberName Method

The GetMemberName method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
GetMemberProperties Method

The GetMemberProperties method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
GetNamespaceParent Method

The GetNamespaceParent method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
GetNextDispID Method

The **GetNextDispID** method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**getPartProperty Method**

Returns a **Variant** that represents the value of a property for a Web Part or an Extensible Stylesheet Language (XSL) Data View Web Part.

```expression
getPartProperty(strPropName, strPropNamespace)
```

*expression*  Required. An expression that returns one of the objects in the Applies To list.

*strPropName*  Required **String**, Specifies the name of the property.

*strPropNamespace*  Optional **String**, Specifies the namespace for the property.
getProperty Method

Returns a **Variant** that represents the property value associated with the style property.

\[\text{expression}.\text{getProperty(\text{strAttributeName})}\]

**expression** Required. An expression that returns one of the objects in the Applies To list.

**strAttributeName** Required **String**. The name of the style attribute whose value is to be returned. You can get a list of style attributes from the [CSS Reference](https://msdn.microsoft.com) on the MSDN Web site.
Example

The following example returns the value associated with a given style property.

Sub GetSSProperties()
    Dim objSS As IFPStyleState
    Dim objDoc As FPHTMLDocument
    Dim objRng As IHTMLTxtRange

    Set objDoc = Application.ActiveDocument
    objDoc.body.innerHTML = "<p>Hello World</p>

    Set objRng = objdoc.body.createTextRange
    Set objSS = objdoc.createStyleState

    With objSS
        .gather objRng
        .Color = vbRed  ' vbRed = 255
        .backgroundColor = vbBlue  ' vbBlue = 16711680
        .Apply
    End With

    Set objRng = objDoc.body.createTextRange
    Set objSS = objDoc.createStyleState
    objSS.gather objRng

    MsgBox "Body background color value is " & objSS _
    .getProperty("background-color") & "." & vbCrLf & _
    "Body text color value is " & objSS.getProperty("color") & "

    Set objRng = Nothing
    Set objSS = Nothing
    Set objDoc = Nothing
End Sub
getScript Method

Returns a **String** that represents the script for an event for a specified element.

*expression.getScript(element, event, index)*

**expression**  Required. An expression that returns one of the objects in the Applies To list.

**element**  Required **IHTMLElement**. Specifies the element that contains the script.

**event**  Required **String**. Specifies the name of the event that causes the script to run.

**index**  Required **Long**. Specifies the script item if an event specifies more than one script to run when the event occurs. Multiple scripts are one-based rather than zero-based so that the **index** for the first script (in the case of multiple scripts) or the only script (in the case of a single script) for an event is "1".
Remarks

Use the `addScript` method to add a script to an element.
Example

The following example copies the script from the **onmouseover** event in the first paragraph of the active document and adds it to the **onclick** event for the body of the active document.

```vbscript
Dim objBody As FPHTMLBody
Dim objPara As FPHTMLParaElement
Dim strScript As String

Set objBody = ActiveDocument.body
Set objPara = ActiveDocument.all.tags("p").Item(0)

strScript = ActiveDocument.getScript(objPara, "onmouseover", 1)
ActiveDocument.addScript objBody, "onclick", "alert('hello!');"
```
**getShadowColor Method**

Returns a String that represents the color of a drop shadow border for a layout cell or cell formatting table.

*expression*.getShadowColor

*expression*  Required. An expression that returns an **FPLayoutRegion** object.
Remarks

Use the `setShadowColor` method to set the color of a drop shadow border for a layout cell or cell formatting table.
**getShadowPosition Method**

Returns a **String** that represents the position of the drop shadow border for a layout cell or cell formatting table.

*expression*.getShadowPosition

*expression* Required. An expression that returns an `FPLayoutRegion` object.
Remarks

Use the `setShadowPosition` method to set the position of the drop shadow border for a layout cell or cell formatting table.
getShadowSize Method

Returns an **Integer** that represents the thickness of the drop shadow border for a layout cell or cell formatting table.

*expression*.getShadowSize

*expression*  Required. An expression that returns an **FPLayoutRegion** object.
Remarks

Use the `setShadowSize` method to set the thickness of the drop shadow border for a layout cell or cell formatting table.
**getShadowSmooth Method**

Returns an **Integer** that represents how smooth or pixelated to render a drop shadow border for a layout cell or cell formatting table.

*expression.getShadowSmooth*

*expression* Required. An expression that returns an **FPLayoutRegion** object.
Remarks

Use the `setShadowSmooth` method to set how smooth or pixelated to render a drop shadow border for a layout cell or cell formatting table.
getWidth Method

Returns an **Integer** that represents the width of a layout cell or cell formatting table.

`expression.getWidth`

`expression` Required. An expression that returns an **FPLayoutRegion** object.
Remarks

Use the `setWidth` method to set the width of a layout cell or cell formatting table.
go Method

The go method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
inRange Method

Returns True if the IHTMLTxtRange object specified in the range argument is contained in the original range.

expression.inRange(range)

expression  Required. An expression that returns an IHTMLTxtRange object.

range  Required IHTMLTxtRange object.
Example

The following example takes an IHTMLTxtRange object, and then returns True if the current selection is contained within the specified IHTMLTxtRange or returns False if the current selection isn't contained within the specified IHTMLTxtRange.

Function IsInRange(objRange As IHTMLTxtRange) As Boolean
    Dim objSelection As IHTMLTxtRange
    Set objSelection = ActiveDocument.Selection.createRange
    If objRange.inRange(objSelection) = False Then
        IsInRange = False
    Else
        IsInRange = True
    End If
End Function

Use the following example to call the preceding function.

Sub CallIsInRange()
    Dim objRange As IHTMLTxtRange
    Set objRange = ActiveDocument.body.createTextRange
    MsgBox IsInRange(objRange)
End Sub
**insertAdjacentHTML Method**

Inserts HTML text into a document at the specified location.

```javascript
expression.insertAdjacentHTML(where, html)
```

*expression* Required. An expression that returns one of the objects in the `Applies To list`.

*where* Required. *String* that specifies where to insert the HTML text. *Can be one of the following values*:

- **beforeBegin** Inserts the specified HTML text immediately before the opening element of the object.
- **afterBegin** Inserts the specified HTML text immediately after the opening element of the object but before all other content in the object.
- **beforeEnd** Inserts the specified HTML text immediately before the end of the object but after all other content in the object.
- **afterEnd** Inserts the specified HTML text immediately after the end of the object.

The following image illustrates where text or HTML is inserted when you use each of the above *String* values.

```html
<p>This is another paragraph.</p>
```

*html* Required. *String* that specifies the HTML text to insert. The string can be a combination of text and HTML tags. This must be valid, well-formed HTML.
Remarks

Use the `insertAdjacentText` method to insert text without providing the accompanying HTML tags. In addition, you can use the `outerHTML` and `outerText` properties to insert HTML and text outside the specified element, and the `innerHTML` and `innerText` properties to insert HTML and text inside the specified element. The following image illustrates where each of these properties applies. Note that the `innerHTML` and `outerHTML` properties require well-formed HTML.
Example

The following example inserts the date when the specified document was last modified into the specified document and formats it with the specified font name, font size, and color.

Sub AddModifiedDateToDocument(objDoc As FPHTMLDocument, strFont As String, strSize As String, strColor As String)
    Dim objFont As FPHTMLFontElement

    objDoc.body.insertAdjacentHTML where:="beforeEnd", _
    HTML:="<p><font id="modifiedon"></font></p>"

    Set objFont = objDoc.body.all.tags("font").Item("modifiedon")

    objFont.insertAdjacentText where:="beforeEnd", Text:="Last modified on: ",
    & objDoc.fileModifiedDate

    With objFont
        .face = strFont
        .Size = strSize
        .Color = strColor
    End With
End Sub

Use the following example to call the preceding example.

Sub CallAddModifiedDateToDocument()
    Call AddModifiedDateToDocument(ActiveDocument, "Arial", "2", "Blue")
End Sub
insertAdjacentText Method

Inserts text into a document at the specified location.

\[ expression.insertAdjacentText(where, text) \]

**expression** Required. An expression that returns one of the objects in the Applies To list.

**where** Required **String**. String that specifies where to insert the text. Can be one of the following values:

- `beforeBegin` Inserts the specified text immediately before the opening element of the object.
- `afterBegin` Inserts the specified text immediately after the opening element of the object but before all other content in the object.
- `beforeEnd` Inserts the specified text immediately before the end of the object but after all other content in the object.
- `afterEnd` Inserts the specified text immediately after the end of the object.

The following image illustrates where text or HTML is inserted when you use each of the above **String** values.

\[
\begin{align*}
&\text{beforeBegin} & \text{afterBegin} & \text{beforeEnd} \\
&amp;\text{This is another paragraph.}\n\end{align*}
\]

**text** Required **String**. String that specifies the text to insert.
Remarks

Use the `insertAdjacentHTML` method to insert text that contains HTML. In addition, you can use the `outerHTML` and `outerText` properties to insert HTML and text outside the specified element, and the `innerHTML` and `innerText` properties to insert HTML and text inside the specified element. The following image illustrates where each of these properties applies. Note that the `innerHTML` and `outerHTML` properties require well-formed HTML.
Example

The following example inserts the date when the specified document was last modified into the specified document and formats it with the specified font name, font size, and color.

Sub AddModifiedDateToDocument(objDoc As FPHTMLDocument, strFont As String, strSize As String, strColor As String)
    Dim objFont As FPHTMLFontElement
    objDoc.body.insertAdjacentHTML where:="beforeEnd", _
    HTML:="<p><font id="filename">" _
Set objFont = objDoc.body.all.tags("font").Item("filename")
objFont.insertAdjacentText "beforeEnd", "Last modified on: " _
& objDoc.fileModifiedDate
    With objFont
        .face = strFont
        .Size = strSize
        .Color = strColor
    End With
End Sub

Use the following example to call the preceding example.

Sub CallAddModifiedDateToDocument()
    Call AddModifiedDateToDocument(ActiveDocument, "Arial", "2", "Blue"
End Sub
**insertCell Method**

Returns an [FPHTMLTableCell](#) object or [IHTMLTableCell](#) object that represents a cell added to a table.

`expression.insertCell(index)`

**expression**  Required. An expression that returns one of the objects in the [Applies To](#) list.

**index**  Optional [Long](#). The column number in the row at which to insert the new cell.
Remarks

When you use the `insertCell` method to create a new cell in a table, it inserts the cell into the specified row but does not insert an entire column. All cells in the row to the right of the new cell are pushed to the right to make room for the new cell.
Example

The following example inserts a new cell in the specified table at the specified row and column. Note that the code subtracts one from both the row number (intRow) and the position of the new cell (intCell). This is because all indexing in Microsoft FrontPage Visual Basic for Applications is zero based.

Function InsertNewCell(intRow As Integer, intCell As Integer, _
objTable As FPHTMLTable) As FPHTMLTableCell
    Dim objCell As FPHTMLTableCell
    Set objCell = objTable.rows(intRow - 1).insertCell(intCell - 1)
    Set InsertNewCell = objCell
End Function

Use the following example to call the preceding function.

Sub CallInsertNewCell()
    Dim objTable As FPHTMLTable
    Dim objCell As FPHTMLTableCell
    Dim objTable As FPHTMLTable
    Dim objCell As FPHTMLTableCell
    Set objTable = ActiveDocument.all.tags("table").Item(0)
    If Not (objTable Is Nothing) Then
        Set objCell = InsertNewCell(1, 3, objTable)
        objCell.innerText = "Hello, World!"
    Else
        MsgBox "You don't have a table in your document." & vbCrLf & "Add a table and run this code again."
    End If
End Sub
**InsertInteractiveButton Method**

Inserts an interactive button into a document at the specified position.

```plaintext
expression.InsertInteractiveButton(Range, styles, title, width, height, URL, target)
```

**expression** Required. An expression that returns one of the objects in the Applies To list.

**Range** Required **IHTMLTxtRange**. Specifies the position where to insert the interactive button. If the range specified includes text or elements, the button will replace these.

**styles** Required **String**. Specifies the style that represents the appearance of the interactive button.

The **style** parameter can be one or more of the following semi-colon delimited **String** values:

<table>
<thead>
<tr>
<th>PREFIX</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>fp-align-rect:</td>
<td>Specifies the pixel coordinates for the original text alignment base image. Image scales appropriately for the preview. Coordinates are left, top, right, bottom or blank (default is the entire image).</td>
</tr>
<tr>
<td>fp-align-rect1:</td>
<td>Specifies the pixel coordinates for the hover text alignment based image. Coordinates are left, top, right, bottom or blank (default is</td>
</tr>
<tr>
<td>fp-align-rect2:</td>
<td>Specifies the pixel coordinates for pressed text alignment based image. Coordinates are left, top, right, bottom or blank (default is</td>
</tr>
<tr>
<td>fp-bbgcolor:</td>
<td>Specifies the default background color for the button. Any Web safe or named color value that is allowed for Cascading Style Sheets (default is white). Note for transparent buttons.</td>
</tr>
<tr>
<td>specifies the internal name for a preset button. Preset button style: following:</td>
<td>Border Bottom 1    Brick Column 6    Embossed Capsule 7</td>
</tr>
</tbody>
</table>
Specifies the font name for the button text.

`fp-font-color-hover:` Specifies the font color for text for the hover state. Any Web safe or named color value that is allowed for Cascading Style Sheets (blank defaults to "Automatic").

`fp-font-color-normal:` Specifies the font color for the button text when in the normal state. Any Web safe or named color value that is allowed for Cascading Style Sheets (blank defaults to black).

`fp-font-color-pressed:` Specifies the font color for button text when in the pressed state. Any Web safe or named color value that is allowed for Cascading Style Sheets or blank (blank equates to "Automatic").

`fp-font-size:` Specifies is the font size, measured in points, for the button text.

`fp-font-style:` Specifies the font style characteristics for the button text. Can be Regular, Italic, Bold, and Bold Italic.

`fp-img-hover:` Specifies whether FrontPage creates a hover image for a checkbox button (1 means true).

`fp-img-pressed:` Specifies whether FrontPage creates a pressed image for a checkbox button (1 means true).

`fp-justify-h:` Specifies the horizontal alignment for the button text. Can be left, right, or blank (default is center).

`fp-justify-v:` Specifies the vertical alignment for button text. Can be top, center, or bottom (default is center).

`fp-preload:` Specifies whether the images for a checkbox button are preloaded on the page. Can be 0 or 1 (1 means true).

`fp-transparent:` Specifies whether to use a solid or a transparent background for a radio button. Can be 0 or 1 (1 means transparent).

`fp-direction:` Specifies whether the text is rendered as left-to-right or right-to-left.

**title** Required **String.** Specifies the text displayed on the button.
**width**  Required **Long**. Specifies the width of the button.

**height**  Required **Long**. Specifies the height of the button.

**URL**  Optional **Variant**. Specifies the URL that is opened when a user clicks on the button in a browser.

**target**  Optional **Variant**. Specifies the target window in which to display the linked page. Correspond to the settings for the **target** property.
Example

The following example inserts an interactive button into the active document at the insertion point.

Dim objRange As IHTMLTxtRange

Set objRange = ActiveDocument.selection.createRange
insertLayoutCell Method

Inserts a layout cell into a layout table at the insertion point.

\[\text{expression}.\text{insertLayoutCell}(\text{iWidth, iHeight, szType, szLocation})\]

\text{expression}  Required. An expression that returns a \text{FPLayoutTablesAndCells} collection.

\text{iWidth}  Required \text{Integer}. Specifies the width of the cell.

\text{iHeight}  Required \text{Integer}. Specifies the height of the cell.

\text{szType}  Required \text{String}. Specifies the type of cell. Can be one of the following \text{String} values: "horizontal", "vertical", or "inline".

\text{szLocation}  Required \text{String}. Specifies where to insert the cell in relation to the insertion point. Can be one of the following \text{String} values: "before", "after", or "wrap".
Example

The following example inserts a layout cell into a cell formatting table. This example assumes that the cell formatting table exists.

ActiveDocument.layoutTablesAndCells_.
   .insertLayoutCell 200, 200, "inline", "wrap"
insertLayoutTable Method

Inserts a cell formatting table into the active document at the insertion point.

expression.insertLayoutTable(iWidth, iHeight)

expression Required. An expression that returns an FPLayoutTablesAndCells collection.

iWidth Required Integer. Specifies the width of the table.

iHeight Required Integer. Specifies the height of the table.
Example

The following example inserts a cell formatting table into the active document at the insertion point.

`ActiveDocument.layoutTablesAndCells _
.insertLayoutTable 100, 100`
insertRow Method

Returns an FPHTMLTableRow object that represents a new row added to a TABLE element.

Note  The new row doesn't include cells. Use the insertCell method to insert cells in a row.

expression.insertRow(index)

expression  Required. An expression that returns one of the objects in the Applies To list.

index  Optional Long. Specifies the index number of the new row. Row numbering starts at zero. If omitted, new row is added to the end of a table.
Remarks

Use the `deleteRow` method to removed a row from a TABLE element.
isEqual Method

Returns True if the IHTMLTxtRange object specified in the range argument is the same as the original IHTMLTxtRange object.

`expression.isEqual(range)`

expression Required. An expression that returns an IHTMLTxtRange object.

range Required IHTMLTxtRange object.
Example

The following example takes an **IHTMLTxtRange** object, checks it against the current selection, and returns **True** if the two ranges are the same.

```vba
Function RangeIsEqual(objRange As IHTMLTxtRange) As Boolean
    Dim objSelection As IHTMLTxtRange
    Set objSelection = ActiveDocument.Selection.createRange
    If objRange.IsEqual(objSelection) = False Then
        RangeIsEqual = False
    Else
        RangeIsEqual = True
    End If
End Function
```

Use the following example to call the preceding function.

```vba
Sub CallRangeIsEqual()
    Dim objRange As IHTMLTxtRange
    Set objRange = ActiveDocument.body.createTextRange
    MsgBox RangeIsEqual(objRange)
End Sub
```
**item Method**

- item method as it applies to the `IHTMLStyleSheetRulesCollection` and `FPLayoutTablesAndCells` collections.

Returns an object that represents a member in one of the above collections.

```
expression.item(index)
```

- **expression**  Required. An expression that returns one of the above collections.

- **index**  Required `Long`. Specifies the ordinal position of an object in the collection.

- item method as it applies to the `FPHTMLFormElement`, `FPHTMLSelectElement`, `IHTMLAreasCollection`, `IHTMLElementCollection`, `IHTMLFormElement`, and `IHTMLSelectElement` objects.

Returns an object that represents a member in one of the above objects.

```
expression.item(name, index)
```

- **expression**  Required. An expression that returns one of the above objects.

- **name**  Optional `Variant`. Specifies the name of the object.

- **index**  Optional `Variant`. Specifies the position of the object in the corresponding collection.
Remarks

When you use a string variable as a value to the name argument, you must use the Microsoft Visual Basic CVAR function to convert the string to a strongly typed Variant.

Returns an object that represents a member of one of the remaining objects in the Applies To list.

expression.item(pvarIndex)

expression  Required. An expression that returns one of the above objects.

pvarIndex  Required Variant. Specifies the position of the object in the corresponding collection.
Remarks

When you use a string variable as a value to the `pvarIndex` argument, you must use the Visual Basic `CVAR` function to convert the string to a strongly typed `Variant`. 
Examples
As it applies to the IHTMLElementCollection object.
The following example function takes an FPHTMLDocument object and
checks whether the images in the document have alt text assigned to them, and
then returns a Boolean.
Function AllImagesHaveAltText(ByRef objDoc As FPHTMLDocument) As Boolean
Dim objImages As IHTMLElementCollection
Dim objImg As IHTMLElement
Dim intCount As Integer
Dim blnAlt As Boolean
Set objImages = objDoc.images
If objImages.Length > 0 Then
For intCount = 0 To objImages.Length - 1
Set objImg = objImages.Item(intCount)
If objImg.alt = "" Then
blnAlt = False
Exit For
Else
blnAlt = True
End If
Next
Else
blnAlt = True
End If
AllImagesHaveAltText = blnAlt
End Function

Use the following example to call the preceding function.
Sub CallAllImagesHaveAltTtext()
MsgBox AllImagesHaveAltText(objDoc:=ActiveDocument)
End Sub

As it applies to the FPHTMLFormElement object.
The following example creates a new form in the specified document and returns
the FPHTMLFormElement object that represents the new form.


Function CreateNewForm(ByRef objDoc As FPHTMLDocument, ByVal strFormName As String) As FPHTMLFormElement

    Dim objForm As IHTMLFormElement

    objDoc.body.insertAdjacentHTML "beforeend", _
    "<form id="" & strFormName & ""></form>"

    Set objForm = objDoc.body.all.tags("form") _
    .Item(CVar(strFormName))

    Set CreateNewForm = objForm
End Function

Use the following example to call the preceding function.

Sub CallCreateNewForm()
    Dim objForm As FPHTMLFormElement

    Set objForm = CreateNewForm(objDoc:=ActiveDocument, strFormName:="newform")
    objForm.insertAdjacentHTML "beforeend", "<input size=""20"">"
End Sub
javaEnabled Method

The javaEnabled method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
layoutRegion Method

Returns an **FPLayoutRegion** object that represents a layout cell in which a specified element is located.

*expression*.layoutRegion(*pHTMLElm*)

**expression**  Required. An expression that returns an **FPLayoutTablesAndCells** collection.

**pHTMLElm**  Required **IHTMLElement**. Specifies the HTML element that points to a layout table, layout cell, or cell formatting table.
Move Method (Page Object Model)

Collapses an **IHTMLTxtRange** to the start of the range, and then moves the insertion point from the starting point in the range by the specified number of units. Returns a **Long** that represents the number of units moved.

*expression*.**Move(Unit, Count)**

*expression* Required. An expression that returns an **IHTMLTxtRange** object.

**Unit** Required **String**. Specifies the type of unit. [Can be one of the following values]:

- character Moves one or more characters.
- word Moves one or more words. A word is a collection of characters terminated by a space or some other white-space character, such as a tab. Punctuation is also included in the word count, so a period at the end of a sentence or a comma in the middle of a sentence would increase the word count by one.
- sentence Moves one or more sentences. A sentence is a collection of words terminated by a punctuation character, such as a period.
- textedit Moves to the start or end of the original range.

**Count** Optional **Long**. Specifies the number of units to move. This number can be positive (moves the insertion point to the right of the starting point in the range) or negative (moves the insertion point to the left of the starting point in the range). The default is 1.
Example

The following example takes an **IHTMLTxtRange** object, a custom fpMoveUnit enumerated type (included in the code) that represents the string value of the *Unit* argument, and an **Integer** that represents the number of units to move, and then returns an **IHTMLTxtRange** object that represents the range after the *Move* method is called.

**Note** Place the following custom enumerated type in the General Declarations section of the code module.

```vba
Public Enum fpMoveUnit
    fpMoveCharacter
    fpMoveWord
    fpMoveSentence
    fpMoveTextEdit
End Enum
```

```vba
Function MoveTextRange(objRange As IHTMLTxtRange, eUnit As fpMoveUnit, intCount As Integer) As IHTMLTxtRange
    Dim strMoveUnit As String

    Select Case eUnit
        Case fpMoveCharacter
            strMoveUnit = "character"
        Case fpMoveWord
            strMoveUnit = "word"
        Case fpMoveSentence
            strMoveUnit = "sentence"
        Case fpMoveTextEdit
            strMoveUnit = "textedit"
    End Select

    If strMoveUnit = "textedit" Then
        objRange.Move strMoveUnit
    Else
        objRange.Move strMoveUnit, intCount
    End If

    Set MoveTextRange = objRange
End Function
```

Use the following example to call the preceding function.
Sub CallMoveTextRange()
    Dim objRange As IHTMLTxtRange

    Set objRange = ActiveDocument.body.createTextRange
    Set objRange = MoveTextRange(objRange, fpMoveWord, 3)

    objRange.Text = "<b>Hello, World!</b> "
End Sub
moveBy Method

The moveBy method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
moveEnd Method

Moves the ending position of an IHTMLTxtRange object. Returns a Long that represents the number of units moved.

expression.moveEnd(Unit, Count)

expression Required. An expression that returns an IHTMLTxtRange object.

Unit Required String. Specifies the type of unit. Can be one of the following values:

- character Moves one or more characters.
  - Moves one or more words. A word is a collection of characters terminated by a space or some other white-space character, such as a tab. Punctuation is also included in the word count, so a period at the end of a sentence or a comma in the middle of a sentence would increase the word count by one.
- sentence Moves one or more sentences. A sentence is a collection of words terminated by a punctuation character, such as a period.
- textedit Moves to the start or end of the original range.

Count Optional Long. Specifies the number of units to move. This number can be positive (moves range to the right) or negative (moves range to the left). The default is 1.
Example

The following example inserts a paragraph into the active document, and then moves the start and end points of the range to change the third and fourth words to a different word.

Sub MoveStartAndEndSelectionPoints()
    Dim objRange As IHTMLTxtRange
    Dim intStart As Integer
    Dim intEnd As Integer

    ActiveDocument.body.innerText = "I enjoy writing programs " & _
    "with FrontPage VBA."
    Set objRange = ActiveDocument.body.createTextRange

    objRange.moveStart "word", 2
    objRange.moveEnd "word", -4

    objRange.Text = "programming "
End Sub
moveStart Method

Moves the starting position of an IHTMLOutRange object. Returns a Long that represents the number of units moved.

expression.moveStart(Unit, Count)

expression Required. An expression that returns an IHTMLOutRange object.

Unit Required String. Specifies the type of unit. Can be one of the following values:

- character Moves one or more characters.
- word Moves one or more words. A word is a collection of characters terminated by a space or some other white-space character, such as a tab. Punctuation is also included in the word count, so a period at the end of a sentence or a comma in the middle of a sentence would increase the word count by one.
- sentence Moves one or more sentences. A sentence is a collection of words terminated by a punctuation character, such as a period.
- textedit Moves to the start or end of the original range.

Count Optional Long. Specifies the number of units to move. This can be positive (moves the start of the range to the right) or negative (moves the start of the range to the left). The default is 1.
Example

The following example inserts a paragraph into the active document, and then moves the start and end points of the range to change the third and fourth words to a different word.

Sub MoveStartAndEndSelectionPoints()
    Dim objRange As IHTMLTxtRange
    Dim intStart As Integer
    Dim intEnd As Integer

    ActiveDocument.body.innerText = "I enjoy writing programs " & _
    "with FrontPage VBA."

    Set objRange = ActiveDocument.body.createTextRange

    objRange.moveStart "word", 2
    objRange.moveEnd "word", -4

    objRange.Text = "programming "
End Sub
moveTo Method

The `moveTo` method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
moveToBookmark Method

Returns True if the specified IHTMLTxtRange object has moved to the String specified in the Bookmark argument.

expression.moveToBookmark(Bookmark)

expression Required. An expression that returns an IHTMLTxtRange object.

Bookmark Required String.
Remarks

Use the `getBookmark` method to return the String for the `Bookmark` argument.
Example

The following example inserts two paragraphs into the active document and then modifies the contents of each paragraph.

Sub MoveToBookmarkMethod()
    Dim objRange As IHTMLTxtRange
    Dim strBookmark As String

    ActiveDocument.body.innerHTML = "<p>Line One</p><p>Line two</p>"
    Set objRange = ActiveDocument.body.createTextRange
    objRange.moveToElementText ActiveDocument.body.all.tags("p").Item(0)
    strBookmark = objRange.getBookmark

    With objRange
        .moveToElementText ActiveDocument.body.all.tags("p").Item(1)
        .Text = "Change Line 2"
        .moveToBookmark strBookmark
        .Text = "Change Line 1"
        .Select
    End With
End Sub
moveToElementText Method

Moves the specified IHTMLTxtRange object to the HTML element specified in the element argument.

expression.moveToElementText(element)

element Required. An expression that returns an IHTMLTxtRange object.

element Required IHTMLElement object. The HTML element to which to move the specified IHTMLTxtRange object.
Example

The following example inserts two paragraphs into the active document, and then moves the **IHTMLTxtRange** object to modify the contents of each paragraph.

Sub MoveToBookmarkMethod()
    Dim objRange As IHTMLTxtRange
    Dim strBookmark As String

    ActiveDocument.body.innerHTML =ençaLine One</p><p>Line two�

    Set objRange = ActiveDocument.body.createTextRange

    objRange.moveToElementText ActiveDocument.body.all.tags("p").Item(0)
    strBookmark = objRange.getBookmark

    With objRange
        .moveToElementText ActiveDocument.body.all.tags("p").Item(1)
        .Text = "Changed Line 2"
        .moveToBookmark strBookmark
        .Text = "Changed Line 1"
        .Select
    End With
End Sub
moveToPoint Method

Moves the start and end positions of an IHTMLTxtRange object to the specified point.

expression.moveToPoint(x, y)

expression Required. An expression that returns an IHTMLTxtRange object.

x Required Long. The horizontal position from the left edge of the application window.

y Required Long. The vertical position from the top edge of the application window.
Remarks

The \( x \) and \( y \) parameters for the \texttt{moveToPoint} method are coordinates from the upper-left corner of the browser window. This includes the toolbars and \texttt{Views} bar in the Microsoft FrontPage application.
Example

This example inserts text into the active document, and then moves and selects the range and replaces it with new text.

Sub UseMoveToPointMethod()
    Dim objRange As IHTMLTxtRange

    ActiveDocument.body.innerHTML = "<p>This is a test.</p>
    Set objRange = ActiveDocument.Selection.createRange

    With objRange
        .moveToPoint 74, 28
        .expand "word"
        .moveStart "word", -1
        .Select
        .Text = "FrontPage"
    End With
End Sub
navigate Method

The `navigate` method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
nextPage Method

The nextPage method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
OnInsert Method

The **OnInsert** method is a member of the FrontPageHostedControl interfaces. For more information about how to use these when creating an ActiveX control to use in Microsoft FrontPage, see the FrontPage Software Development Kit (SDK) on the Microsoft Developer Network (MSDN) Web site.
OnLayout Method

The **OnLayout** method is a member of the FrontPageHostedControl interfaces. For more information about how to use these when creating an ActiveX control to use in Microsoft FrontPage, see the FrontPage Software Development Kit (SDK) on the Microsoft Developer Network (MSDN) Web site.
OnRemove Method

The OnRemove method is a member of the FrontPageHostedControl interfaces. For more information about how to use these when creating an ActiveX control to use in Microsoft FrontPage, see the FrontPage Software Development Kit (SDK) on the Microsoft Developer Network (MSDN) Web site.
OnRestore Method

The **OnRestore** method is a member of the FrontPageHostedControl interfaces. For more information about how to use these when creating an ActiveX control to use in Microsoft FrontPage, see the FrontPage Software Development Kit (SDK) on the Microsoft Developer Network (MSDN) Web site.
open Method

The **open** method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
optimizeHTML Method

Optimizes the Hypertext Markup Language (HTML) in a Web page.

`expression.optimizeHTML`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Remarks

Use the OptimizeHTMLFlags property to specify how HTML is to be optimized in Web pages.
Example

The following example optimizes the HTML in the active document.

`ActiveDocument.optimizeHTML`
**parentElement Method**

Returns an `IHTMLElement` object that represents the element that is one level up in the HTML element hierarchy.

`expression.parentElement`

`expression` Required. An expression that returns an `IHTMLTxtRange` object.
Example

The following example returns the parent element of the currently selected text.

```vba
Dim objRange As IHTMLTxtRange
Dim objElement As IHTML Element

Set objRange = ActiveDocument.selection.createRange
Set objElement = objRange.parentElement
```
parentNodeSetIterator Method

Returns an **IHTMLXSLElement** object that represents the first Extensible Markup Language (XML) node for the parent node set iterator construct (e.g. xsl:apply-templates or xsl:for-each) of the specified XSL element.

**expression.parentNodeSetIterator**

**expression**    Required. An expression that returns one of the objects in the Applies To list.
**parentTemplateBodyExpansion Method**

Returns an **IHTMLXSLElement** object that represents the first parent template or Extensible Stylesheet Language (XSL) xsl:for-each construct for the specified XSL element.

`expression.parentTemplateBodyExpansion`

*expression* Required. An expression that returns one of the objects in the Applies To list.
**parentWebPart Method**

Returns an **IHTMLWebPartElement** object that represents the parent Web Part for the specified Extensible Stylesheet Language (XSL) element in an XSL Data View Web Part.

*expression.parentWebPart*

**expression**  Required. An expression that returns one of the objects in the Applies To list.
parseCodeChanges Method

Updates the page when it is displayed in Code view.

expression.parseCodeChanges

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

In previous versions of Microsoft FrontPage, it was impossible to run Microsoft Visual Basic for Applications (VBA) code that modified the HTML in a page — for example, adding elements or changing attributes — while viewing the HTML in FrontPage. Doing so displayed a "Permission Denied" error.

With Microsoft Office FrontPage 2003, you can now make changes to the HTML in a page by using VBA code while a page is displayed in the FrontPage Code view. However, the FrontPage Page Object model becomes unavailable when edits are made in Code view. The `parseCodeChanges` method allows you to reparse the page without having to change the view and eliminates the "Permission Denied" error.
Example

The following example refreshes the active document.

ActiveDocument.parseCodeChanges
**pasteHTML Method**

Pastes the specified HTML into a range within a document. If anything is selected, the selection is replaced with the new HTML and text.

`expression.pasteHTML(html)`

*expression* Required. An expression that returns one of the objects in the Applies To list.

*html* Required *String*. The HTML to paste. May include text as well as HTML.
Example

The following example pastes a paragraph into the current selection, replacing any text and HTML that may be selected.

Dim objRange As IHTMLTxtRange
Set objRange = ActiveDocument.selection.createRange
objRange.pasteHTML "<p>New paragraph.</p>"
previousPage Method

The `previousPage` method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
prompt Method

The prompt method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
propertyInConflict Method

Returns a Boolean that indicates if a specified style setting is different from another style setting for the same element. True indicates that the style setting is in conflict.

expression.propertyInConflict(strPropertyName)

expression Required. An expression that returns one of the objects in the Applies To list.

strPropertyName Required String. The name of a style attribute. You can get a list of style attributes from the CSS Reference on the Microsoft Developer Network (MSDN) Web site.
Example

The following example uses two styles that contain background-color properties: one in a STYLE element in the HEAD element and one in a FONT element in the BODY element. This example will display a message saying that the two style properties are in conflict.

Note If the background-color property for both styles equalled the same color, the example would display a message saying that the two style properties are not in conflict.

Sub DisplayConflict()
    Dim objSS As IFPStyleState
    Dim objDoc As FPHTMLDocument
    Dim objHead As IHTMLElement
    Dim objRng As IHTMLTxtRange
    Dim objPara As FPHTMLParaElement

    Set objDoc = Application.ActiveDocument
    Set objHead = objDoc.all.tags("head").Item(0)

    'Add a STYLE element to the HEAD element.
    objHead.insertAdjacentHTML "beforeend", "<style>" & _
        "<!-- .bgStyle { color: #800080; background-color: #FF0000 } --></style>"

    'Add a paragraph to the body of the document.
    objDoc.body.innerHTML = "<p class="bgStyle">Example" & _
        "<font style="background-color: #0000FF">Paragraph</font></p>"

    'Create a style state, a paragraph, and a text range with which to work
    Set objSS = objDoc.createStyleState
    Set objPara = objDoc.body.all.tags("p").Item(0)
    Set objRng = objDoc.body.createTextRange

    'Move the text range to the paragraph element
    'and gather the paragraph element into the style state object. You need to select the text
    'range, and then use the gather method before the style state object will be usable.
    objRng.moveToElementText objPara
    objRng.Select
    objSS.gather objRng
With objSS
    If .propertyInConflict("background-color") Then
        MsgBox "The property is in conflict."
    Else
        MsgBox "The property is not in conflict."
    End If
End With
End Sub
# propertySettingReason Method

Returns an **Integer** that specifies the numeric value of the reason for the property setting. See [property setting reasons](#) for more information on the numeric values.

The property setting reasons and their numeric values are as follows:

<table>
<thead>
<tr>
<th>Value</th>
<th>Reason</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ssrNoReason</td>
<td>Specifies that there is no reason for the style property.</td>
</tr>
<tr>
<td>1</td>
<td>ssrInConflict</td>
<td>Specifies that the style property conflicts with another style property in the same range. For example, if you have <code>&lt;p&gt;&lt;span style=&quot;font-size: small&quot;&gt;foo&lt;/span&gt;&lt;font size=3&gt;bar&lt;/font&gt;&lt;/p&gt;</code> and you select the entire paragraph, the properties will be in conflict.</td>
</tr>
<tr>
<td>2</td>
<td>ssrUninheritedDefault</td>
<td>Specifies that the style property is a default HTML setting. For example, if an element does not have a <code>border</code> attribute or a border style and no border property has been set for the <code>style</code> attribute.</td>
</tr>
<tr>
<td>3</td>
<td>ssrCloneProps</td>
<td>Specifies that a style property has been copied using the <strong>clone</strong> method.</td>
</tr>
<tr>
<td>4</td>
<td>ssrStoredStyles</td>
<td>Specifies that the style property is being stored. For example, if you select a table with empty cells and apply formatting, such as BOLD, rather than putting in a <strong>B</strong> element at that point,</td>
</tr>
</tbody>
</table>
Microsoft FrontPage stores the value until the user enters text. Specifies that the style setting is inherited from the BODY element.

- **ssrInheritFromBody**

Specifies that the style properties are inherited from the BODY element.

- **ssrBgPropOfSpanningParent**

Specifies that the style properties are inherited from surrounding SPAN elements.

- **ssrBgPropOfTableRowSection**

Specifies that the style properties are inherited from table row sections.

- **ssrLegendTextAlign**

Specifies that the style properties are for text alignment in a LEGEND element.

- **ssrBodyColorInTable**

Specifies that the body color is inherited peculiarly in a table.

- **ssrHeaderNestedInSupOrSub**

Specifies that a header inside a SUP or SUB element has a special font size.

- **ssrHTMLSemanticStructuralTagDefault**

Specifies that the style property is for a semantic structural element. For example, an H1 element is a semantic structural element. In addition to forcing a text paragraph, it carries the semantic meaning of a “header”. Different semantic structural tags have different default text properties.

- **ssrHTMLSemanticSpanningTagDefault**

Specifies that the style property is for a semantic spanning element. For example, the EM element is a semantic spanning element; in addition to applying italics (a text property), it is supposed to specify that a region of text has emphasis.
<table>
<thead>
<tr>
<th>Line</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>ssrHTMLSpanningFormattingTag</td>
<td>Specifies that the style property is for a formatting element, such as the FONT, B, or I element.</td>
</tr>
<tr>
<td>14</td>
<td>ssrHTMLFontTag</td>
<td>Specifies that the style property is for a FONT element.</td>
</tr>
<tr>
<td>15</td>
<td>ssrHTMLBaseFontTag</td>
<td>Specifies that the style property is for a BASEFONT element.</td>
</tr>
<tr>
<td>16</td>
<td>ssrCSSStyleSheetRule</td>
<td>Specifies that the style property is set by using a cascading style sheet rule.</td>
</tr>
<tr>
<td>17</td>
<td>ssrCSSStyleSheetRuleBangImportant</td>
<td>Specifies that the style property is set by using a cascading style sheet rule with the !IMPORTANT modifier.</td>
</tr>
<tr>
<td>18</td>
<td>ssrCSSStyleAttribute</td>
<td>Specifies that the style property is set by using a cascading style sheet attribute.</td>
</tr>
<tr>
<td>19</td>
<td>ssrCSSStyleAttributeBangImportant</td>
<td>Specifies that the style property is set by using a cascading style sheet attribute with the !IMPORTANT modifier.</td>
</tr>
<tr>
<td>20</td>
<td>ssrHTMLAttr</td>
<td>Specifies that the style property is set by using an HTML attribute.</td>
</tr>
<tr>
<td>21</td>
<td>ssrHTMLBorderAttr</td>
<td>Specifies that the style property is set by using the <code>border</code> attribute.</td>
</tr>
<tr>
<td>22</td>
<td>ssrHTMLAlignAttr</td>
<td>Specifies that the style property is set by using the <code>align</code> attribute.</td>
</tr>
<tr>
<td>23</td>
<td>ssrHTMLLHSpaceAttr</td>
<td>Specifies that the style property is set by using the <code>hspace</code> attribute.</td>
</tr>
<tr>
<td>24</td>
<td>ssrHTMLVSpaceAttr</td>
<td>Specifies that the style property is set by using the <code>vspace</code> attribute.</td>
</tr>
<tr>
<td>Code</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ssrBODYAttr</td>
<td>Specifies that the style property is set by using an attribute in the BODY element.</td>
<td></td>
</tr>
<tr>
<td>ssrDTC</td>
<td>Specifies that the style property is set by using a design-time control.</td>
<td></td>
</tr>
<tr>
<td>ssrInternalVisibilityControl</td>
<td>Specifies a style property for which FrontPage is controlling the visibility for internal purposes.</td>
<td></td>
</tr>
<tr>
<td>ssrTheme</td>
<td>Specifies that the style property is set by using a FrontPage 98 style theme.</td>
<td></td>
</tr>
<tr>
<td>Note</td>
<td>Newer FrontPage themes are CSS-based.</td>
<td></td>
</tr>
<tr>
<td>ssrFunkyBotBusiness</td>
<td>Specifies that the style property is set by using a FrontPage component.</td>
<td></td>
</tr>
<tr>
<td>ssrListImageIndent</td>
<td>Specifies that the style property is for an image that is being used as a bullet in a list.</td>
<td></td>
</tr>
<tr>
<td>ssrCellHTMLPropInheritance</td>
<td>Specifies that the style property for a table cell is inherited from the parent table.</td>
<td></td>
</tr>
<tr>
<td>ssrCellFloating</td>
<td>Specifies that the padding property for a table cell is inherited from the parent table.</td>
<td></td>
</tr>
<tr>
<td>ssrCellPaddingFromTable</td>
<td>Specifies that the border property for a table cell is inherited from the parent table.</td>
<td></td>
</tr>
<tr>
<td>ssrCellBorderFromTable</td>
<td>Specifies that the alignment property is being affected by the float property.</td>
<td></td>
</tr>
<tr>
<td>ssrFloatImpliesAlign</td>
<td>Specifies that the border</td>
<td></td>
</tr>
</tbody>
</table>
36 ssrDoubleBorderMustBe3High  property is a certain thickness because the border style is set to double.

Specifies that the float property setting is required because the position property is set to something other than static or auto.

37 ssrPositionedCantFloat

Specifies that the z order property is required because the positioned property is set to static or auto.

38 ssrStaticObjectsNoZOrder

Specifies that the style property is for the height of a list item.

39 ssrListItemHeight

Specifies that the style property is for an object contained inside an Escher shape.

40 ssrEscherChild

Specifies that the style property is for the vertical alignment of an element or object.

41 ssrVerticalAlignSetsTextVAlign

Specifies that the style property is for an HR element for which the margin is being used as padding.

42 ssrHRMarginIsPadding

Specifies that the Input Method Editor is shown in FrontPage by manipulating formatting properties on the text associated with a SPAN element.

43 ssrIME

Specifies that the style property is for an INPUT element with a lang attribute setting that comes from the locale identifier (LCID) for the user's computer. This affects the font in which page content is displayed.

44 ssrSetLangForINPUTTag
45 ssrFloatMarginIsNonZero for floating text always has a margin of 3.
Specifies that the style property is for the margin attribute of a BODY element that cannot be set to less than zero.

46 ssrBodyMarginNotLessThanZero Specifies that the style property is for the formatting of a SUP or SUB element.

47 ssrSupOrSubFormattingTag Specifies that the paragraph alignment is set by using a CENTER element.

48 ssrHTMLCentering Specifies that the structural alignment for the style property is based on the inherited text alignment.

49 ssrTextAlignSetsStructAlign Specifies that the background properties have been reset because the specified object or element is manually positioned.

50 ssrPositionResetsBGProps Specifies that FrontPage calculates the top and bottom margin properties when the page layout is complete.

51 ssrComputedMargin

expression.propertySettingReason(strPropertyName)

describes the reason for setting the property value.

expression Required. An expression that returns one of the objects in the Applies To list.

strPropertyName Required. A String that represents a style attribute. You can get a list of style attributes from the CSS Reference on the Microsoft Developer Network (MSDN) Web site.
Remarks

Use the `propertySettingReason` method if you need to implement a formatting command that requires various style properties or where you need to know the current style properties for a range of elements. For example, if you have a text range that contains `<p><span style="font-size: small">foo</span><font size=3>bar</font></p>` and you want to set the `font-size` style attribute, you may want to check to see if the `propertySettingReason` method returns a value of 1 (`ssrInConflict`), which would indicate that there are conflicting property settings within the range. In that case, you may not want to apply yet another font size style setting to the range.
Example

The following example inserts text into the active document, and then if the `background-color` attribute isn't specified, it applies a background color to the body of the document.

```vbscript
Dim objStyleState As IFPStyleState
Dim objRange As IHTMLTxtRange

Set objRange = ActiveDocument.Selection.createRange
Set objStyleState = ActiveDocument.createStyleState

With objStyleState
    If .propertySettingReason("background-color") = 0 Then
        .gather objRange
        .setProperty "background-color", vbBlue
        .Apply
    End If
End With
```
queryCommandEnabled Method

Returns a **Boolean** that indicates if the specified command can be executed. Whether or not a command can be executed is based on the current state of the document. For more information on the state of the current document, see the document's `readyState` property.

`expression.queryCommandEnabled(cmdID)`

- **expression** Required. An expression that returns one of the objects in the Applies To list.

- **cmdID** Required. A **String** that represents the command identifier. For a list of available commands see the `execCommand` method.
Example

The following example prompts the user to enter a command identifier. The `queryCommandEnabled` method is executed using the user input and a message is displayed to the user depending on the result of the method.

Sub QueryCommand()
'Determines if a command can be executed
'based on the document state.

    Dim objApp As FrontPage.Application
    Dim objDoc As DispFPHTMLDocument
    Dim strUser As String

    Set objApp = FrontPage.Application
    Set objDoc = objApp.ActiveDocument
    'Prompt user to enter command name.
    strUser = InputBox("Enter a command identifier to be executed.")
    'Attept to run the associated command
    If objDoc.queryCommandEnabled(cmdID:=strUser) = True Then
        'If yes - display message.
        MsgBox "The command " & strUser & " can be executed."
    Else
        'If no - display message.
        MsgBox "The command " & strUser & " cannot be executed."
    End If

End Sub
**queryCommandIndeterm Method**

Returns a **Boolean** that determines if the specified command will return an indeterminate state. An indeterminate state means that the command could not return a binary result with the specified parameter. For example, the **Bold** command will return indeterminate if the current selection contains both bold and non-bold text. If **True**, the command will return an indeterminate result. If **False**, the command will return a binary result.

```
expression.queryCommandIndeterm(cmdID)
```

- **expression** Required. An expression that returns one of the objects in the Applies To list.
- **cmdID** Required. A **String** that represents the specified command identifier. For a list of available commands see the **execCommand** method.
Example

The following example prompts the user to enter a command identifier to determine if the specified command will return an indeterminate state. A message is displayed to the user based on the result of the method.

Sub QueryCommand()
'Determines if a command will return an
'indeterminate state

    Dim objApp As FrontPage.Application
    Dim objDoc As DispFPHTMLDocument
    Dim strUser As String

    Set objApp = FrontPage.Application
    Set objDoc = objApp.ActiveDocument
    'Prompt user to enter command name.
    strUser = InputBox("Enter a command identifier to be executed.")
    'Run the associated command
    If objDoc.queryCommandIndeterm(cmdID:=strUser) = True Then
        'If yes - display message.
        MsgBox "The command " & strUser & _
            " will return an indeterminate state."
    Else
        'If no - display message.
        MsgBox "The command " & strUser & _
            " will not return an indeterminate state."
    End If

End Sub
queryCommandState Method

Returns a **Boolean** that indicates the state of the specified command. If **True**, the specified command has already been carried out on the current selection. If **False**, the specified command has not yet been carried out on the current selection.

```
expression.queryCommandState(cmdID)
```

- **expression**  Required. An expression that returns one of the objects in the Applies To list.

- **cmdID**    Required. A **String** that specifies the command identifier. For a list of available commands see the **execCommand** method.
Remarks

This method returns **Null** if the state of the current selection could not be determined.
Example

The following example prompts the user for a command identifier and displays a message based on the result of the method.

Sub QueryCommand()
' Determines whether a command has been carried out

    Dim objApp As FrontPage.Application
    Dim objDoc As DispFPHTMLDocument
    Dim strUser As String

    Set objApp = FrontPage.Application
    Set objDoc = objApp.ActiveDocument
    ' Prompt user to enter command name.
    strUser = InputBox("Enter a command identifier to be executed.")
    ' Run the associated command, checks command status and displays
    If objDoc.queryCommandState(cmdID:=strUser) = True Then
        MsgBox "The command " & strUser & " has already been carried out."
    Else
        MsgBox "The command " & strUser & " has not yet been carried out."
    End If

End Sub
queryCommandSupported Method

Returns a Boolean that indicates if the specified command is supported by the current selection.

expression.queryCommandSupported(cmdID)

expression Required. An expression that returns one of the objects in the Applies To list.

cmdID Required. A String that represents the specified command identifier. For a list of available commands see the execCommand method.
Example

The following example prompts the user to enter a command identifier. A message is displayed to the user depending on the result of the command.

Sub QueryCommand()
    'Determines whether a command is supported
    'by the current selection

    Dim objApp As FrontPage.Application
    Dim objDoc As DispFPHTMLDocument
    Dim strUser As String

    Set objApp = FrontPage.Application
    Set objDoc = objApp.ActiveDocument
    'Prompt user to enter command name.
    strUser = InputBox("Enter a command identifier to be executed.")
    'Run the associated command
    If objDoc.queryCommandSupported(cmdID:=strUser) = True Then
        'If yes - display message.
        MsgBox "The command " & strUser & ":" & " is supported by the current selection."
    Else
        'If no - display message.
        MsgBox "The command " & strUser & " is not supported by the current selection."
    End If

End Sub
queryCommandText Method

Returns a String that represents a text value associated with the specified command.

\[ expression.queryCommandText(cmdID) \]

**expression**  Required. An expression that returns one of the objects in the Applies To list.

**cmdID**  Required. A String that represents the command identifier. For a list of available commands see the execCommand method.
**Example**

The following example displays the text value associated with a given command.

```vba
Sub QueryCommand()
    'Determines the value of a specified command
    Dim objApp As FrontPage.Application
    Dim objDoc As DispFPHTMLDocument
    Dim strUser As String
    Dim strValue As String

    Set objApp = FrontPage.Application
    Set objDoc = objApp.ActiveDocument
    'Prompt user to enter command name.
    strUser = InputBox("Enter a command identifier to be executed.")
    'Run the associated command.
    strValue = objDoc.queryCommandText(strUser)
    MsgBox "The text value associated with the specified command is:"
End Sub
```
queryCommandValue Method

Returns a **String** that indicates the value of the specified command.

`expression.queryCommandValue(cmdID)`

*expression*  Required. An expression that returns one of the objects in the Applies To list.

*cmdID*  Required. A **String** that represents the command identifier. For a list of available commands see the `execCommand` method.
**Example**

The following example displays the current value of the specified command.

```vbnet
Sub QueryCommand()
    'Determines the value of a specified command
    Dim objApp As FrontPage.Application
    Dim objDoc As DispFPHTMLDocument
    Dim strUser As String
    Dim strValue As String

    Set objApp = FrontPage.Application
    Set objDoc = objApp.ActiveDocument

    'Prompt user to enter command name.
    strUser = InputBox("Enter a command identifier to be executed.")

    'Run the associated command.
    strValue = objDoc.queryCommandValue(strUser)
    MsgBox "The value of the command is: " & strValue
End Sub
```
reapplyScript Method

Verifies that the functions associated with script calls on the selected element are within a SCRIPT element in the HEAD element of the document and inserts them if they are missing.

expression.reapplyScript(element)

expression Required. An expression that returns one of the objects in the Applies To list.

element Optional Variant. Specifies an element within the document. If omitted, the reapplyScript method checks all elements within the document.
Refresh Method (Page Object Model)

The **Refresh** method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
regenerateDataView Method

Executes the Extensible Stylesheet Language Transform (XSLT) on the data in an XSL Data View Web Part.

expression.regenerateDataView

expression Required. An expression that returns one of the objects in the Applies To list.
**reload Method**

Reloads the specified document.

`expression.reload(flag)`

**expression**  Required. An expression that returns one of the objects in the Applies To list.

**flag**  Optional **Boolean**. **False** reloads the page from the browser cache (default). **True** reloads the page from the server.
Remarks

If the **reload** method is run on a page to which changes have been made, a message is displayed asking if the user wants to save changes made to the document.
Example

The following example takes an `FPHTMLDocument` object and reloads the document from the browser cache.

```vba
Sub ReloadDocument(objDoc as FPHTMLDocument)
    objDoc.Location.reload
End Sub
```

Use the following example to call the preceding subroutine.

```vba
Sub CallReloadDocument()
    Call ReloadDocument(ActiveDocument)
End Sub
```
RemoteInvokeEx Method

The **RemoteInvokeEx** method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
RemoteQueryService Method

The RemoteQueryService method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
Remove Method (Page Object Model)

Removes an object from a collection.

expression.Remove(index)

expression  Required. An expression that returns one of the objects in the Applies To list.

index  Optional Long. specifies the ordinal position of the object in the collection.
Example

The following example removes the first AREA element from the first MAP element in the active document. This example assumes there is at least one MAP element that contains at least one AREA element in the active document.

Dim objMap As FPHTMLMapElement
Dim objAreas As IHTMLAreasCollection

Set objMap = ActiveDocument.all.tags("map").Item(0)
Set objAreas = objMap.areas

'Remove the first area
objAreas.Remove 0
removeAttribute Method

Returns a **Boolean** that represents whether the attribute was successfully removed. **True** indicates that the attribute was removed. **False** indicates that the attribute was not removed.

`expression.removeAttribute(strAttributeName, lFlags)`

*expression* Required. An expression that returns one of the objects in the Applies To list.

*strAttributeName* Required **String**.

*lFlags* Optional **Integer**. Specifies whether to use a case-sensitive search to locate the attribute. 1 (True) indicates that the uppercase and lowercase letters in the specified *sAttributeName* parameter must exactly match those in the attribute name. 0 (False) indicates that the attribute name match to the *sAttributeName* parameter is not case sensitive. Default value is 1.
Remarks

When no corresponding property exists for an attribute, use the `getAttribute` method to get the value of an attribute. When no corresponding property exists for an attribute or when the property that accesses an attribute is read-only, use the `setAttribute` method to set the value of an attribute.
Example

The following example specifies a source file for the first IMG element in the active document and removes the `height` and `width` attributes since the new image may have a different height and width.

Set objImage = ActiveDocument.images.Item(0)

With objImage
    .src = "graphics/chelan.jpg"
    .removeAttribute "height", False
    .removeAttribute "width", False
End With
removeBotAttribute Method

Removes the attribute specified by the \textit{strAttributeName} attribute. Returns a \textbf{Boolean} that represents whether the attribute was removed. \textbf{True} indicates the attribute was successfully removed.

\textit{expression}.\textit{removeBotAttribute}(\textit{strAttributeName})

\textit{expression}  Required. An expression that returns one of the objects in the Applies To list.

\textit{strAttributeName}  Required \textbf{String}. The string that represents the name of the attribute.
Example

This example uses the **removeBotAttribute** method to remove a search bot.

Private Sub AccessBots()
    Dim objFPBot As FPHTMLFrontPageBotElement
    Dim objBody As FPHTMLBody
    Dim strBot As String
    Dim objPage As PageWindow

    strBot = ""
    strBot = strBot & "<!— webbot bot=""Search"" s-index=""All""""
    strBot = strBot & " s-fields s-text=""Search for:""
    strBot = strBot & " i-size=""20"" s-submit=""Start Search"
    strBot = strBot & " s-clear=""Reset"" s-timestampformat=""%m/%d/
    strBot = strBot & " tag=""BODY"" -->

    Set objBody = ActivePageWindow.Document.body
    Set objPage = ActivePageWindow

    Call objBody.insertAdjacentHTML("BeforeEnd", _
        strBot)

    Set objFPBot = _
        objPage.Document.all.tags("webbot").Item(0)

    MsgBox objFPBot.getBotAttribute("s-submit")

    objFPBot.setBotAttribute "s-submit", "new item"
    MsgBox objFPBot.getBotAttribute("s-submit")

    objFPBot.removeBotAttribute "s-submit"
    MsgBox objFPBot.getBotAttribute("s-submit")
End Sub
removeFilter Method

The `removeFilter` method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**removeImport Method**

Removes the specified imported style sheet reference from a STYLE element.

`expression.removeImport(lIndex)`

- **expression**: Required. An expression that returns one of the objects in the Applies To list.

- **lIndex**: Required Long. The ordinal position of the imported style sheet to remove from the collection of imported style sheets in the STYLE element.
Remarks

Use the `addImport` method to add an imported style sheet to the collection of imported style sheets for a STYLE element. Use the `imports` property to access the collection of imported style sheets for a STYLE element.
Example

The following example removes the first imported style sheet from the first STYLE element in the active document if there is at least one imported style sheet referenced.

Dim objStyle As FPHTMLStyleSheet
Set objStyle = ActiveDocument.styleSheets.Item(0)

If objStyle.imports.Length <> 0 Then
    objStyle.removeImport (0)
End If
**removePartProperty Method**

Returns a *Boolean* that represents the successful removal of the specified property for a Web Part. *False* indicates that the property was not removed.

```csharp
expression.removePartProperty(strPropName, strPropNamespace)
```

- **expression** Required. An expression that returns one of the objects in the Applies To list.
- **strPropName** Required *String*. Specifies the name of the Web Part property.
- **strPropNamespace** Optional *String*. Specifies the namespace for the Web Part property.
removeRule Method

Removes the cascading style sheet rule at the specified position.

expression.removeRule(lIndex)

expression  Required. An expression that returns one of the objects in the Applies To list.

lIndex  Required Long. Specifies a zero-based number that indicates the ordinal position of the selector.
Remarks

The `removeRule` method removes all cascading style sheet rules for a selector. For example, if there are multiple style sheet attributes that affect the BODY element, all style attributes are removed.

The `removeRule` method only removes cascading style sheet rules from STYLE elements in a page and does not modify external style sheets referenced.
Example

The following example removes the first cascading style sheet rule in the first cascading style sheet in the active document.

Dim objCSS As FPHTMLStyleSheet
Set objCSS = ActiveDocument.styleSheets(0)
objCSS.removeRule 0
removeScript Method

Removes the specified script from a given element.

*expression*.removeScript(*element, event, index*)

*expression*  Required. An expression that returns one of the objects in the Applies To list.

*element*  Required *IHTMLElement*. Specifies the element that contains the script.

*event*  Required *String*. Specifies the name of the event that causes the script to run.

*index*  Optional *Variant*. Specifies the script item if an event specifies more than one script to run when the event occurs. Multiple scripts are one-based rather than zero-based so that the *index* for the first script for an event is "1".
Example

The following example removes the script from the `onclick` event in the BODY element of the active document.

```vba
Dim objBody As FHTMLBody
ActiveDocument.removeScript objBody, "onclick"
```
replace Method

The `replace` method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
reset Method

The reset method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
resizeBy Method

The *resizeBy* method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
resizeTo Method

The `resizeTo` method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
save Method

Saves a document.

\textit{expression.save(showUI)}

\textit{expression}  Required. An expression that returns one of the objects in the Applies To list.

\textit{showUI}  Optional \textbf{Boolean}. Specifies whether to display the Save As dialog box.
Example

The following example saves the active document.

ActiveDocument.Save False
scroll Method

The scroll method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
scrollBy Method

The `scrollBy` method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
scrollIntoView Method

Scrolls the page so the an element is positioned at the top of the page window.

expression.scrollIntoView(varargStart)

expression  Required. An expression that returns one of the objects in the Applies To list.

varargStart  Optional Variant. Specifies the element to scroll into view.
**Example**

The following example scrolls the first table in the active document so that it is positioned at the top of the page window.

Dim objTable As FPHTMLTable

Set objTable = ActiveDocument.all.tags("table").Item(0)
objTable.scrollIntoView
scrollTo Method

The `scrollTo` method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
select Method

Selects the text in the specified IHTMLTxtRange object.

expression.select

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example takes an FPHTMLDocument object, creates an IHTMLTxtRange object that represents the text in the specified document, selects the range, and then returns an IHTMLTxtRange object that represents the selection.

Function SelectDocument(ByRef objDoc As FPHTMLDocument) As IHTMLTxtRange
    Dim objRange As IHTMLTxtRange
    Set objRange = objDoc.body.createTextRange
    objRange.Select
    Set SelectDocument = objRange
End Function

Use the following example to call the preceding function.

Sub CallSelectDocument()
    Call SelectDocument(objDoc:=ActiveDocument)
End Sub
setAttribute Method

As it applies to the **FPLayoutRegion** object.

Sets the specified attribute to the given value.

```
expression.setAttribute(attr, val)
```

- **expression**   Required. An expression that returns an **FPLayoutRegion** object.
- **attr**   Required **FpLayoutCellAttributes**. Specifies the attribute.

**FpLayoutCellAttributes** can be one of the following **FpLayoutCellAttributes** constants.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>border_bottom_color</code></td>
<td>For a list of possible values, see the <strong>setBorderColor</strong> method.</td>
</tr>
<tr>
<td><code>border_bottom_size</code></td>
<td>For a list of possible values, see the <strong>setBorderSize</strong> method.</td>
</tr>
<tr>
<td><code>border_foot_color</code></td>
<td>For a list of possible values, see the <strong>setBorderColor</strong> method.</td>
</tr>
<tr>
<td><code>border_foot_size</code></td>
<td>For a list of possible values, see the <strong>setBorderSize</strong> method.</td>
</tr>
<tr>
<td><code>border_head_color</code></td>
<td>For a list of possible values, see the <strong>setBorderColor</strong> method.</td>
</tr>
<tr>
<td><code>border_head_size</code></td>
<td>For a list of possible values, see the <strong>setBorderSize</strong> method.</td>
</tr>
<tr>
<td><code>border_left_color</code></td>
<td>For a list of possible values, see the <strong>setBorderColor</strong> method.</td>
</tr>
<tr>
<td><code>border_left_size</code></td>
<td>For a list of possible values, see the <strong>setBorderSize</strong> method.</td>
</tr>
<tr>
<td><code>border_right_color</code></td>
<td>For a list of possible values, see the <strong>setBorderColor</strong> method.</td>
</tr>
<tr>
<td><code>border_right_size</code></td>
<td>For a list of possible values, see the <strong>setBorderSize</strong> method.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>border_top_color</code></td>
<td>For a list of possible values, see the <code>setBorderColor</code> method.</td>
</tr>
<tr>
<td><code>border_top_size</code></td>
<td>For a list of possible values, see the <code>setBorderSize</code> method.</td>
</tr>
<tr>
<td><code>cellFormattingID</code></td>
<td>For a list of possible values, see the <code>cellFormattingID</code> property.</td>
</tr>
<tr>
<td><code>content_color</code></td>
<td>For a list of possible values, see the <code>setContentColor</code> method.</td>
</tr>
<tr>
<td><code>content_html</code></td>
<td>For a list of possible values, see the <code>setContentHtml</code> method.</td>
</tr>
<tr>
<td><code>content_padding</code></td>
<td>For a list of possible values, see the <code>setContentPadding</code> method.</td>
</tr>
<tr>
<td><code>content_type</code></td>
<td>For a list of possible values, see the <code>setContentType</code> method.</td>
</tr>
<tr>
<td><code>content_valign</code></td>
<td>For a list of possible values, see the <code>setContentVAlign</code> method.</td>
</tr>
<tr>
<td><code>corner_bottom_left_border_color</code></td>
<td>For a list of possible values, see the <code>setCornerBorderColor</code> method.</td>
</tr>
<tr>
<td><code>corner_bottom_left_color</code></td>
<td>For a list of possible values, see the <code>setCornerColor</code> method.</td>
</tr>
<tr>
<td><code>corner_bottom_left_height</code></td>
<td>For a list of possible values, see the <code>setCornerHeight</code> method.</td>
</tr>
<tr>
<td><code>corner_bottom_left_imageurl</code></td>
<td>For a list of possible values, see the <code>setCornerImageUrl</code> method.</td>
</tr>
<tr>
<td><code>corner_bottom_left_type</code></td>
<td>For a list of possible values, see the <code>setCorner</code> method.</td>
</tr>
<tr>
<td><code>corner_bottom_left_width</code></td>
<td>For a list of possible values, see the <code>setCornerWidth</code> method.</td>
</tr>
<tr>
<td><code>corner_bottom_right_border_color</code></td>
<td>For a list of possible values, see the <code>setCornerBorderColor</code> method.</td>
</tr>
<tr>
<td><code>corner_bottom_right_color</code></td>
<td>For a list of possible values, see the <code>setCornerColor</code> method.</td>
</tr>
<tr>
<td><code>corner_bottom_right_height</code></td>
<td>For a list of possible values, see the <code>setCornerHeight</code> method.</td>
</tr>
<tr>
<td><code>corner_bottom_right_imageurl</code></td>
<td>For a list of possible values, see the <code>setCornerImageUrl</code> method.</td>
</tr>
</tbody>
</table>
corner_bottom_right_type
For a list of possible values, see the setCorner method.

corner_bottom_right_width
For a list of possible values, see the setCornerWidth method.

corner_top_left_border_color
For a list of possible values, see the setCornerBorderColor method.

corner_top_left_color
For a list of possible values, see the setCornerColor method.

corner_top_left_height
For a list of possible values, see the setCornerHeight method.

corner_top_left_imageurl
For a list of possible values, see the setCornerImageUrl method.

corner_top_left_type
For a list of possible values, see the setCorner method.

corner_top_left_width
For a list of possible values, see the setCornerWidth method.

corner_top_right_border_color
For a list of possible values, see the setCornerBorderColor method.

corner_top_right_color
For a list of possible values, see the setCornerColor method.

corner_top_right_height
For a list of possible values, see the setCornerHeight method.

corner_top_right_imageurl
For a list of possible values, see the setCornerImageUrl method.

corner_top_right_type
For a list of possible values, see the setCorner method.

corner_top_right_width
For a list of possible values, see the setCornerWidth method.

footer_color
For a list of possible values, see the setFooterColor method.

footer_html
For a list of possible values, see the setFooterHtml method.

footer_padding
For a list of possible values, see the setFooterPadding method.

footer_size
For a list of possible values, see the
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>footer_valign</td>
<td>For a list of possible values, see the <code>setFooterVAlign</code> method.</td>
</tr>
<tr>
<td>header_color</td>
<td>For a list of possible values, see the <code>setHeaderColor</code> method.</td>
</tr>
<tr>
<td>header_padding</td>
<td>For a list of possible values, see the <code>setHeaderPadding</code> method.</td>
</tr>
<tr>
<td>header_size</td>
<td>For a list of possible values, see the <code>setHeaderSize</code> method.</td>
</tr>
<tr>
<td>header_html</td>
<td>For a list of possible values, see the <code>setHeaderHtml</code> method.</td>
</tr>
<tr>
<td>header_valign</td>
<td>For a list of possible values, see the <code>setHeaderVAlign</code> method.</td>
</tr>
<tr>
<td>height</td>
<td>For a list of possible values, see the <code>setHeight</code> method.</td>
</tr>
<tr>
<td>is_cell</td>
<td>For a list of possible values, see the <code>iscell</code> property.</td>
</tr>
<tr>
<td>is_table</td>
<td>For a list of possible values, see the <code>isTable</code> property.</td>
</tr>
<tr>
<td>margin_bottom</td>
<td>For a list of possible values, see the <code>setMargin</code> method.</td>
</tr>
<tr>
<td>margin_left</td>
<td>For a list of possible values, see the <code>setMargin</code> method.</td>
</tr>
<tr>
<td>margin_right</td>
<td>For a list of possible values, see the <code>setMargin</code> method.</td>
</tr>
<tr>
<td>margin_top</td>
<td>For a list of possible values, see the <code>setMargin</code> method.</td>
</tr>
<tr>
<td>parent</td>
<td>For a list of possible values, see the <code>parent</code> property.</td>
</tr>
<tr>
<td>shadow_color</td>
<td>For a list of possible values, see the <code>setShadowColor</code> method.</td>
</tr>
<tr>
<td>shadow_position</td>
<td>For a list of possible values, see the <code>setShadowPosition</code> method.</td>
</tr>
<tr>
<td>shadow_size</td>
<td>For a list of possible values, see the <code>setShadowSize</code> method.</td>
</tr>
</tbody>
</table>
shadow_smooth

For a list of possible values, see the `setShadowSmooth` method.

width

For a list of possible values, see the `setWidth` method.

**val** Required **Variant**. Specifies the attribute value.

As it applies to all other objects in the Applies To list.

Sets the attribute for an element.

```
expression.setAttribute(strAttributeName, lFlags)
```

**expression** Required. An expression that returns one of the objects in the Applies To list.

**strAttributeName** Required **String**. The name of the attribute.

**AttributeValue** Required **Variant**. The value of the attribute.

**lFlags** Optional **Long**. The value of the object.
Remarks

Use the `setAttribute` method to programmatically set a read-only property that has a corresponding attribute or an attribute for which there is no property in the Microsoft FrontPage object model.
Example

As it applies to **FPHTMLStyle** object.

The following example sets the style attribute's position property for the BODY element of the active document. The **position** property is read only, so the **setAttribute** method is the only way to set the **position** property programmatically.

```javascript
If ActiveDocument.body.Style.Position = "" Then
    ActiveDocument.body.Style.setAttribute "position", "absolute"
End If
```
SetBodyMapping Method

Specifies the mapping that is applied to regions that are in the body section of a Dynamic Web Template.

expression.SetBodyMapping(bzOldMaster, bzNewMaster, bzOldRegion, bzNewRegion)

expression  Required. An expression that returns one of the objects in the Applies To list.

bzOldMaster  Required String. The relative path and filename of the old Dynamic Web Template file. Use an empty String if there is currently no attached to the document.

bzNewMaster  Required String. The relative path and filename of the new Dynamic Web Template file.

bzOldRegion  Required String. The region name in the old Dynamic Web Template file from which to map. Use an empty String to indicate the entire BODY element.

bzNewRegion  Required String. The region name in the new Dynamic Web Template file to which to map the old region.
Example

The following example maps the regions from the body and head sections of the document, and then attaches the Dynamic Web Template file.

Dim objState As DynamicTemplateState
Set objState = Application.CreateDynamicTemplateState

objState.SetBodyMapping "", "template.dwt", "", "A1"
objState.SetHeadMapping "", "template.dwt", "", "metadata"

ActiveDocument.ApplyDynamicTemplate "template.dwt", objState
setBorderColor Method

Sets the border color for a layout cell or cell formatting table.

.expression.setBorderColor(pszType, pszVal)

expression Required. An expression that returns an FPLayoutRegion object.

pszType Required String. Specifies which border. Can be one of the following String values: "all", "top", "bottom", "left", "right", "header" and "footer".

pszVal Required String. Specifies the color value. Can be any Web safe color value (for example, "#FFFFFF") or a named color value (for example, "red").
Remarks

Use the `getBorderColor` method to get the value of the color of a border for a layout cell or cell formatting table.
setBorderSize Method

Sets the thickness of the border for a layout cell or cell formatting table.

expression.setBorderSize(pszType, ival)

expression Required. An expression that returns an FPLayoutRegion object.

pszType Required String. Specifies which border. Can be one of the following String values: "all", "top", "bottom", "left", "right", "header" and "footer".

ival Required Integer. Specifies the thickness of the border. Corresponds to the border attribute.
Remarks

Use the `getBorderSize` method to return the thickness of the border for a layout cell or cell formatting table.
setBotAttribute Method

Sets an attribute for a Microsoft FrontPage component.

\textit{expression}.\texttt{setBotAttribute}(\texttt{strAttributeName}, \texttt{AttributeValue})

\textit{expression}  Required. An expression that returns one of the objects in the Applies To list.

\texttt{strAttributeName}  Required \texttt{String}. The string that represents the name of the attribute.

\texttt{AttributeValue}  Required \texttt{Variant}. The value of the attribute specified in the \texttt{strAttributeName} argument.
Example

This example uses the setBotAttribute method to set a search bot.

Private Sub AccessBots()
    Dim objFPBot As FPHTMLFrontPageBotElement
    Dim objBody As FPHTMLBody
    Dim strBot As String
    Dim objPage As PageWindow
    strBot = ""
    strBot = strBot & "<!— webbot bot=""Search"" s-index=""All""”"
    strBot = strBot & " s-fields s-text=""Search for:"""
    strBot = strBot & " i-size=""20"" s-submit=""Start Search"""
    strBot = strBot & " s-clear=""Reset"" s-timestampformat=""%m/%d/
    strBot = strBot & " tag=""BODY"" -->
    Set objBody = ActivePageWindow.Document.body
    Set objPage = ActivePageWindow
    Call objBody.insertAdjacentHTML("BeforeEnd", _
        strBot)
    Set objFPBot = _
        objPage.Document.all.tags("webbot").Item(0)
    MsgBox objFPBot.getBotAttribute("s-submit")
    objFPBot.setBotAttribute "s-submit", "new item"
    MsgBox objFPBot.getBotAttribute("s-submit")
    objFPBot.removeBotAttribute "s-submit"
    MsgBox objFPBot.getBotAttribute("s-submit")
End Sub
setContentColor Method

Sets the background color of a layout cell or cell formatting table.

`expression.setContentColor(pszVal)`

`expression` Required. An expression that returns an `FPLayoutRegion` object.

`pszVal` Required String. Specifies the color value. Can be any Web safe color value (for example, "#FFFFFF") or a named color value (for example, "red").
Remarks

Use the `getContentColor` method to get the background color of a layout cell or cell formatting table.
**setContentHtml Method**

Sets the HTML text inside of a layout cell or the content layout cell of a cell formatting table.

`expression.setContentHtml(pszVal)`

- **expression** Required. An expression that returns an `FPLayoutRegion` object.
- **pszVal** Required `String`. Specifies the HTML code and text.
**Remarks**

Use the ` getContentHtml ` method to return the HTML text inside of a layout cell or the content layout cell of a cell formatting table.
setContentPadding Method

Sets the cell padding for a layout cell or the content layout cell of a cell formatting table.

expression.setContentPadding(ival)

expression Required. An expression that returns an FPLayoutRegion object.

ival Required Integer. Specifies the size of the padding. Corresponds to the cellpadding attribute of the layout cell or the content layout cell from a cell formatting table.
Remarks

Use the `getContentPadding` method to return the cell padding for a layout cell or the content layout cell of a cell formatting table.
**setContentType Method**

Sets the type of content that is inside of a layout cell or cell formatting table.

```java
expression.setContentType(pszVal)
```

*expression*  Required. An expression that returns an [FPLayoutRegion](#) object.

*pszVal*  Required [String](#). Specifies the type of content. May be one of the following [String](#) values: "NavBody", "ContentHead2", "ContentBody", or another value. This [String](#) is enclosed in quotes in the post-comment tag commonly used in the layout templates; for example, ``` <!-- MSCellType="stringvalue" --> ```.
Remarks

This value is an in-memory value only and is not placed in the HTML in the document.

Use the `getContentType` method to return the type of content that is inside of a layout cell or cell formatting table.
setContentVAlign Method

Sets the vertical alignment for text inside of a layout cell or cell formatting table.

(expression).setContentVAlign(pszVal)

expression   Required. An expression that returns an FPLayoutRegion object.

pszVal   Required String. Specifies the vertical alignment. Can be one of the following String values: "top", "middle", "bottom".
Remarks

Use the `getContentVAlign` method to return the vertical alignment for text inside of a layout cell or cell formatting table.
setCorner Method

Sets the type of corner border for a specified corner in a layout cell or cell formatting table.

expression.setCorner(pszType, pszVal)

expression Required. An expression that returns an FPLayoutRegion object.

pszType Required String. Specifies which corner. Can be one of the following String values: "top-left", "top-right", "bottom-left", "bottom-right", and "all".

pszVal Required String. Specifies the type of corner border. Can be one of the following String values: "none", "round", or "custom".
Remarks

Use the `getCorner` method to return the type of corner border for a specified corner in a layout cell or cell formatting table.
setCornerBorderColor Method

Sets the color of a corner border for a layout cell or cell formatting table.

expression.setCornerBorderColor(pszType, pszVal)

expression  Required. An expression that returns an FPLLayoutRegion object.

pszType  Required String. Specifies which corner. Can be one of the following String values: "top-left", "top-right", "bottom-left", "bottom-right", and "all".

pszVal  Required String. Specifies the color value. Can be any Web safe color value (for example, "#FFFFFF") or a named color value (for example, "red").
Remarks

Use the `getCornerBorderColor` method to return the color of a corner border for a layout cell or cell formatting table.
setCornerColor Method

Sets the color of a specified corner in a layout cell or cell formatting table.

\[ \text{expression.setCornerColor}(psz\text{Type},\pszVal) \]

\textit{expression}  Required. An expression that returns an \texttt{FPLayoutRegion} object.

\textit{pszType}  Required \texttt{String}. Specifies which corner. Can be one of the following \texttt{String} values: "top-left", "top-right", "bottom-left", "bottom-right", and "all".

\textit{pszVal}  Required \texttt{String}. Specifies the color value. Can be any Web safe color value (for example, "#FFFFFF") or a named color value (for example, "red").
Remarks

Use the `getCornerColor` method to return the color of a specified corner in a layout cell or cell formatting table.
setCornerHeight Method

Sets the height of a specified corner in a layout cell or cell formatting table.

```plaintext
expression.setCornerHeight(pszType, ival)
```

**expression**  Required. An expression that returns an [FPLayoutRegion](#) object.

**pszType**  Required *String*. Specifies which corner. Can be one of the following *String* values: "top-left", "top-right", "bottom-left", "bottom-right", or "all".

**ival**  Required *Integer*. Specifies the height of the corner in pixels.
Remarks

Use the `getCornerHeight` method to return the height of a specified corner in a layout cell or cell formatting table.
**setCornerImageUrl Method**

Sets the path and file name for an image to display for a specified corner in a layout cell or cell formatting table.

`expression.setCornerImageUrl(pszType, ppszVal)`

*expression*  Required. An expression that returns an `FPLayoutRegion` object.

*pszType*  Required `String`. Specifies which corner. Can be one of the following `String` values: "top-left", "top-right", "bottom-left", "bottom-right", or "all".

*ppszVal*  Required `String`. Sets the path and file name of the corner image.
Remarks

Use the `getCornerImageUrl` method to return the path and file name for an image to display for a specified corner in a layout cell or cell formatting table.
**setCornerWidth Method**

Sets the width of a specified corner in a layout cell or cell formatting table.

```
expression.setCornerWidth(pszType, ival)
```

*expression*  Required. An expression that returns an [FPLLayoutRegion](#) object.

*pszType*  Required [String](#). Specifies which corner. Can be one of the following [String](#) values: "top-left", "top-right", "bottom-left", "bottom-right", or "all".

*ival*  Required [Integer](#). Specifies the width in pixels.
Remarks

Use the `getCornerWidth` method to return the width of a specified corner in a layout cell or cell formatting table.
**setEndPoint Method**

Sets the end point of one range based on the end point of another range.

**expression.setEndPoint**( **how**, **SourceRange** )

**expression**  Required. An expression that returns one of the objects in the Applies To list.

Compares an end point of an **IHTMLTxtRange** object with an end point of another **IHTMLTxtRange** object. Returns a **Long** that represents the comparison of the specified points.

**expression.compareEndPoints**( **how**, **SourceRange** )

**expression**  Required. An expression that returns an **IHTMLTxtRange** object.

**how**  Required **String**. Specifies the end point to transfer. **May be one of four values**.

The **String** value of the **how** parameter can be one of the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartToStart</td>
<td>Moves the start of the specified <strong>IHTMLTxtRange</strong> object to the start of the <strong>IHTMLTxtRange</strong> object specified in the <strong>SourceRange</strong> parameter.</td>
</tr>
<tr>
<td>StartToEnd</td>
<td>Moves the start of the specified <strong>IHTMLTxtRange</strong> object to the end of the <strong>IHTMLTxtRange</strong> object specified in the <strong>SourceRange</strong> parameter.</td>
</tr>
<tr>
<td>EndToStart</td>
<td>Moves the end of the specified <strong>IHTMLTxtRange</strong> object to the start of the <strong>IHTMLTxtRange</strong> object specified in the <strong>SourceRange</strong> parameter.</td>
</tr>
<tr>
<td>EndToEnd</td>
<td>Moves the end of the specified <strong>IHTMLTxtRange</strong> object to the end of the <strong>IHTMLTxtRange</strong> object specified in the <strong>SourceRange</strong> parameter.</td>
</tr>
</tbody>
</table>
**SourceRange**  Required **IHTMLTxtRange**. Specifies the range from which the source end point is to be taken.
**setFooterColor Method**

Sets the color of the footer for a layout cell or cell formatting table.

```expression.setFooterColor(pszVal)```

**expression** Required. An expression that returns an **FPLLayoutRegion** object.

**pszVal** Required **String**. Specifies the color value. Can be any Web safe color value (for example, "#FFFFFF") or a named color value (for example, "red").
Remarks

Use the `getFooterColor` method to return the color of the footer for a layout cell or cell formatting table.
setFooterHtml Method

Sets the HTML text inside a footer for a layout cell or cell formatting table.

\[ \text{expression.setFooterHtml(pszVal)} \]

- **expression**  Required. An expression that returns an [FPLayoutRegion](#) object.
- **pszVal**  Required [String](#). Specifies the HTML code and text.
Remarks

Use the `getFooterHtml` method to return the HTML text inside a footer for a layout cell or cell formatting table.

The `setFooterHtml` method will return an error if a footer is not present. Use the `setFooterSize` method to add a footer to a layout cell or cell formatting table.
setFooterPadding Method

Sets the cell padding in a footer for a layout cell or cell formatting table.

\[expression.setFooterPadding(ival)\]

**expression**  Required. An expression that returns an [FPLayoutRegion](#) object.

**ival**  Required [Integer](#). Specifies the size of the padding. Corresponds to the [cellpadding](#) attribute.
Remarks

Use the `getFooterPadding` method to return the cell padding in a footer for a layout cell or cell formatting table.
setFooterSize Method

Sets the height of the footer for a layout cell or cell formatting table.

expression.setFooterSize(ival)

expression  Required. An expression that returns an FPLLayoutRegion object.

ival  Required Integer. Specifies the size of the footer. A zero value removes a footer.
Remarks

Use the `getFooterSize` method to return the height of the footer for a layout cell or cell formatting table.
setFooterVAlign Method

Sets the vertical alignment for text inside a footer for a layout cell or cell formatting table.

expression.setFooterVAlign(pszVal)

expression Required. An expression that returns an FPLayoutRegion object.

pszVal Required String. Specifies the vertical alignment. Can be one of the following String values: "top", "middle", or "bottom".
Remarks

Use the `getFooterVAlign` method to return the vertical alignment for text inside a footer for a layout cell or cell formatting table.
**setHeaderColor Method**

Sets the color of the header for a layout cell or cell formatting table.

```
expression.setHeaderColor(pszVal)
```

*expression*  Required. An expression that returns an **FPLayoutRegion** object.

*pszVal*  Required **String**. Specifies the color value. Can be any Web safe color value (for example, "#FFFFFF") or a named color value (for example, "red").
Remarks

Use the `getHeaderColor` method to return the color of the header for a layout cell or cell formatting table.
setHeaderHtml Method

Sets the HTML text inside a header for a layout cell or cell formatting table.

(expression).setHeaderHtml(pszVal)

expression  Required. An expression that returns an FPLayoutRegion object.

pszVal  Required String. Specifies the HTML code and text.
Remarks

Use the `getHeaderHtml` method to return the HTML text inside a header for a layout cell or cell formatting table.

The `setHeaderHtml` method will return an error if a header is not present. Use the `setHeaderSize` method to add a header to a layout cell or cell formatting table.
setHeaderPadding Method

Sets the cell padding in a header for a layout cell or cell formatting table.

(expression.setHeaderPadding(ival))

expression Required. An expression that returns an FPLayoutRegion object.

ival Required Integer. Specifies the size of the padding. Corresponds to the cellpadding attribute.
Remarks

Use the `getHeaderPadding` method to return the cell padding in a header for a layout cell or cell formatting table.
setHeaderSize Method

Sets the height of the header for a layout cell or cell formatting table.

\[ \text{expression} \cdot \text{setHeaderSize}(ival) \]

**expression**  Required. An expression that returns an [FPLayoutRegion](#) object.

**ival**  Required [Integer](#). Specifies the size of the header. A zero value removes a header.
Remarks

Use the `getHeaderSize` method to return the height of the header for a layout cell or cell formatting table.
**setHeaderVAlign Method**

Sets the vertical alignment for text inside a header for a layout cell or cell formatting table.

\[ \text{expression}\cdot\text{setHeaderVAlign}(pszVal) \]

*expression* Required. An expression that returns an [FPLayoutRegion](#) object.

*pszVal* Required [String](#). Specifies the vertical alignment. Can be one of the following [String](#) values: "top", "middle", or "bottom".
Remarks

Use the getHeaderVAlign method to return the vertical alignment for text inside a header for a layout cell or cell formatting table.
SetHeadMapping Method

Specifies the mapping that is applied to regions that are in the head section of a Dynamic Web Template.

expression.SetHeadMapping(bzOldMaster, bzNewMaster, bzOldRegion, bzNewRegion)

expression Required. An expression that returns one of the objects in the Applies To list.

bzOldMaster Required String. The relative path and file name of the old Dynamic Web Template file. Use an empty String if there is currently no Dynamic Web Template attached to the document.

bzNewMaster Required String. The relative path and file name of the new Dynamic Web Template file.

bzOldRegion Required String. The region name in the old Dynamic Web Template file from which to map. Use an empty String if there is currently no Dynamic Web Template attached to the document.

bzNewRegion Required String. The region name in the new Dynamic Web Template file to which to map the old region.
Example

The following example maps the regions from the body and head sections of the document, and then attaches the Dynamic Web Template file.

Dim objState As DynamicTemplateState
Set objState = Application.CreateDynamicTemplateState

objState.SetBodyMapping "", "template.dwt", "", "A1"
objState.SetHeadMapping "", "template.dwt", "", "metadata"

ActiveDocument.ApplyDynamicTemplate "template.dwt", objState
setHeight Method

Sets the height of a layout cell or cell formatting table.

\[ \textit{expression}.setHeight(ival) \]

\textit{expression} Required. An expression that returns an \texttt{FPLayoutRegion} object.

\textit{ival} Required \texttt{Integer}. Specifies the height in pixels.
Remarks

Use the \texttt{getHeight} method to return the height of a layout cell or cell formatting table.
setInterval Method

The `setInterval` method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
setMargin Method

Sets the margin settings for a layout cell or cell formatting table.

definition

`expression.setMargin(pszType, ival)`

*expression*  Required. An expression that returns an `FPLayoutRegion` object.

*pszType*  Required *String*. Specifies which margin. Can be one of the following *String* values: "top", "bottom", "left", or "right".

*ival*  Required *Integer*. Specifies the width of the margin in pixels. A zero value removes a previously set margin.
Remarks

Use the `getMargin` method to return the margin settings for a layout cell or cell formatting table.
setPartProperty Method

Sets a property for a Web Part or an Extensible Stylesheet Language (XSL) Data View Web Part.

`expression.setPartProperty(strPropName, strPropNamespace, PropValue)`

`expression` Required. An expression that returns one of the objects in the Applies To list.

`strPropName` Required `String`. Specifies the name of the property.

`strPropNamespace` Optional `String`. Specifies the namespace for the property.

`PropValue` Required `Variant`. Specifies the value of the property.
**setProperty Method**

Sets the value of the specified attribute for an [IFPStyleState](#) object.

```plaintext
expression.setProperty(strAttributeName, AttributeValue)
```

*expression* Required. An expression that returns an [IFPStyleState](#) object.

*strAttributeName* Required [String](#). The name of the style attribute to be modified. You can get a list of style attributes from the [CSS Reference](#) on the Microsoft Developer Network (MSDN) Web site.

*AttributeValue* Required [Variant](#). The setting for the specified style attribute.
Remarks

Use the apply method to apply the changes to the text range or element.

For color property settings, the AttributeValue parameter can be an integer from 0 (black) to 16777215 (white) or a Microsoft Visual Basic color constant, such as vbBlue. For a list of Visual Basic color constants, see Color Constants on the MSDN Web site.
Example

The following example creates a SPAN element with a style attribute that sets the background color for the element to blue for each element in the active document that contains text.

Dim objSs As IFPStyleState
Dim objRng As IHTMLTxtRange

Set objRng = ActiveDocument.body.createTextRange
Set objSs = ActiveDocument.createStyleState

With objSs
    .gather objRng
    .setProperty "background-color", vbBlue
    .Apply
End With
**setShadowColor Method**

Sets the color of a drop shadow border for a layout cell or cell formatting table.

```
expression.setShadowColor(pszVal)
```

**expression** Required. An expression that returns an `FPLayoutRegion` object.

**pszVal** Required `String`. Specifies the color value. Can be any Web safe color value (for example, "#FFFFFF") or a named color value (for example, "red").
Remarks

Use the `getShadowColor` method to return the color of a drop shadow border for a layout cell or cell formatting table.
**setShadowPosition Method**

Sets the position of the drop shadow border for a layout cell or cell formatting table.

*expression*.setShadowPosition(*pszType*)

*expression*  Required. An expression that returns an **FPLayoutRegion** object.

*pszType*  Required **String**. Specifies where to place the drop shadow. Can be one of the following **String** values: "top-left", "top-right", "bottom-left", "bottom-right", or "none".
Remarks

Use the `getShadowPosition` method to return the position of the drop shadow border for a layout cell or cell formatting table.
setShadowSize Method

Sets the thickness of the drop shadow border for a layout cell or cell formatting table.

\textit{expression.setShadowSize(ival)}

\textit{expression}   Required. An expression that returns an \texttt{FPLayoutRegion} object.

\textit{ival}   Required \texttt{Integer}. Specifies the thickness of the drop shadow border.
Remarks

Use the `getShadowSize` method to return the thickness of the drop shadow border for a layout cell or cell formatting table.
**setShadowSmooth Method**

Sets how smooth or pixelated to render a drop shadow border for a layout cell or cell formatting table.

`expression.setShadowSmooth(ival)`

*expression* Required. An expression that returns an *FPLayoutRegion* object.

*ival* Required *Integer*. Specifies the level of pixelation. Can be an *Integer* from zero to 100. Zero indicates no smoothing effect; 100 indicates full smoothing effect.
Remarks

Use the `getShadowSmooth` method to return how smooth or pixelated to render a drop shadow border for a layout cell or cell formatting table.
setTimeout Method

The `setTimeout` method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**setToDefault Method**

Sets a specified style attribute to its default value.

*expression*.**SetToDefault**(strAttributeName)

*expression*   Required. An expression that returns an **IFPStyleState** object.

strAttributeName   Required **String**. The name of the style attribute. You can get a list of style attributes from the **CSS Reference** on the Microsoft Developer Network (MSDN) Web site.
Remarks

Use the `apply` method to apply the changes to the text range or element.
Example

The following example resets the background color and font size of the selected text to its default value.

Dim objSs As IFPStyleState
Dim objRng As IHTMLTxtRange

Set objRng = ActiveDocument.Selection.createRange
Set objSs = ActiveDocument.createStyleState

With objSs
  .gather objRng
  .setToDefault "background-color"
  .setToDefault "font-size"
  .Apply
End With
setWidth Method

Sets the width of a layout cell or cell formatting table.

```
expression.setWidth(ival)
```

**expression** Required. An expression that returns an `FPLayoutRegion` object.

**ival** Required `Integer`. Specifies the width in pixels.
Remarks

Use the `getWidth` method to return the width of a layout cell or cell formatting table.
**showHelp Method**

The *showHelp* method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
showModalDialog Method

The `showModalDialog` method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
start Method

The **start** method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
Stop Method

The Stop method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
submit Method

The submit method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
tags Method

Returns a `IHTMLElementCollection` collection that represents a collection of specific HTML elements that are contained within the specified object.

`expression.tags(tagName)`

`expression`  Required. An expression that returns one of the objects in the Applies To list.

`tagName`  Required `Variant`. Specifies the name of the HTML element.
Remarks

Use the **all** property to return a collection of all HTML elements within a specified object and then use the **tags** method to return a collection of elements of a specific type. For example, in the following line, the **tags** method returns a collection of P elements in the active document.

```
ActiveDocument.all.tags("p")
```

Once you've returned the collection of elements, you can use the **item** method to specify which element in the collection you want to work with. For example, in the following line, the **item** method returns the first paragraph in the collection of P elements in the active document.

```
ActiveDocument.all.tags("p").Item(0)
```
**Example**

The following example inserts a list box into the active document and then uses the **tags** method to return a collection of SELECT elements and the **item** method to return the specific SELECT element that represents the newly added list box.

```vba
Sub AddListBox()
    Dim objListBox As FPHTMLSelectElement
    Dim strHTML As String

    strHTML = "<SELECT ID=""pets"">" & "<OPTION VALUE=""1"">Cat" & _
             vbCrLf & "<OPTION VALUE=""2"">Dog" & vbCrLf & _
             "<OPTION VALUE=""3"">Snake" & vbCrLf & "</SELECT>"

    ActiveDocument.body.insertAdjacentHTML _
        where:="beforeend", HTML:=strHTML

    Set objListBox = ActiveDocument.all.tags("select").Item("pets")

    With objListBox
        .multiple = True
        .Size = "6"
        .onchange = "fnChange()"
    End With
End Sub
```
taintEnabled Method

The *taintEnabled* method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
toString Method

The toString method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
updateDynamicTemplate Method

Updates the pages that are attached to a Dynamic Web Template so that any changes to the Dynamic Web Template are applied to pages that reference the Dynamic Web Template.

expression.updateDynamicTemplate(state)

does not apply, Required. An expression that returns one of the objects in the Applies To list.

state Required DynamicTemplateState. Specifies the region mapping to use when updating the Dynamic Web Template. Use the SetHeadMapping and SetBodyMapping methods of the DynamicTemplateState object to customize region mapping.
Remarks

If an error occurs, the update process will terminate unless the `SkipOnQuery` property is set to `True`. 
Example

The following example updates the Dynamic Web Template references in the active document.

Dim objState As DynamicTemplateState
Set objState = Application.CreateDynamicTemplateState

If ActiveDocument.DynamicTemplate <> "" Then _
    ActiveDocument.UpdateDynamicTemplate objState
End Sub
write Method

The write method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
writeFile Method

Saves the page to the specified path and file name.

\[ expression.writeFile(path) \]

**expression**  Required. An expression that returns one of the objects in the Applies To list.

**path**  Required **String**. Specifies the path and file name of where to save the page.
Example

The following example saves the active document to the specified path and filename.

```javascript
ActiveDocument.writeFile "c:\newpage.htm"
```
writeln Method

The `writeln` method is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
xmlTree Method

Returns an **IXMLDOMNode** object (which is part of the Microsoft XML, version 2.0 object model interface) that represents the parent node of the Extensible Markup Language (XML) tree for a Data View web part.

*expression.xmlTree*

*expression*  Required. An expression that returns one of the objects in the Applies To list.
xslInstruction Method

Returns an **IXMLDOMNode** object (which is part of the Microsoft XML, version 2.0 object model interface) that represents the nearest Extensible Stylesheet Language (XSL) instruction for the specified text range.

*expression*.xslInstruction(*Range*)

*expression*  Required. An expression that returns one of the objects in the Applies To list.

*Range*  Required **IHTMLTxtrange**. Specifies a range or selection of text and elements.
**xslNode Method**

Returns an **IXMLDOMNode** object (which is part of the Microsoft XML, version 2.0 object model interface) that represents the specified Extensible Stylesheet Language (XSL) element in an XSL Data View Web Part.

*expression*.xslNode

*expression*  Required. An expression that returns one of the objects in the Applies To list.
xslNodeSetIterator Method

Returns an IXMLDOMNode object (which is part of the Microsoft XML, version 2.0 object model interface) that represents the first node set iterator Extensible Stylesheet Language (XSL) construct (for example, xsl:apply-templates or xsl:for-each) in the specified range.

expression.xslNodeSetIterator(Range)

expression  Required. An expression that returns one of the objects in the Applies To list.

Range  Required IHTMLTxtRange. Specifies a range or selection of text and elements.
xslTemplateBody Method

Returns an IXMLDOMNode object (which is part of the Microsoft XML, version 2.0 object model interface) that represents the first parent template or Extensible Stylesheet Language (XSL) xsl:for-each construct of a Data View Web Part.

expression.xslTemplateBody(Range)

expression  Required. An expression that returns one of the objects in the Applies To list.

Range  Required IHTMLTxtRange. Specifies a range or selection of text and elements.
**xslTree Method**

Returns an **IXMLDOMNode** object (which is part of the Microsoft XML, version 2.0 object model interface) that represents the parent node of the Extensible Stylesheet Language (XSL) tree for the Data View Web Part.

*expression*.xslTree

*expression* Required. An expression that returns one of the objects in the Applies To list.
Show All
action Property

As it applies to the SearchInfo object.

Sets or returns a fpSearchAction that represents the type of search to perform.

fpSearchAction can be one of the following fpSearchAction constants.

- fpSearchFindTag: Searches for matching text in the HTML tags.
- fpSearchFindText: Searches for matching text in the document.
- fpSearchReplaceAllText: Searches and replaces all matching text in the document.
- fpSearchReplaceText: Replaces the text and searches for the next occurrence.

expression.action

description

expression  Required. An expression that returns a SearchInfo object.

As it applies to all other objects in the Applies To list.

Sets or returns a String that represents the URL where the content of a form is sent for processing. This URL could represent a database, an e-mail address, or an ASP page.

expression.action

description

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The action property sets the action attribute for the FORM element. The way the form submits data depends on the value of the method and encoding properties.
Example

As it applies to the `SearchInfo` object.

The following example selects the next occurrence of the P element, if one is found in the active document.

```vba
Dim objSearch As SearchInfo
Dim blnFound As Boolean
Dim objRange As IHTMLTxtRange

Set objSearch = Application.CreateSearchInfo
objSearch.Find = "p"
objSearch.Action = fpSearchFindTag

Set objRange = Application.ActiveDocument.selection.createRange
blnFound = Application.ActiveDocument.Find(objSearch, Nothing, objRange)
If blnFound = True Then objRange.Select
```

As it applies to the `FPHTMLFormElement` object.

The following example sets the URL for the `action`, `method`, and `encoding` properties for the specified form.

```vba
Sub SetFormAction(objForm As FPHTMLFormElement, __
    strAction As String, strMethod As String, __
    strEncoding As String)

    With objForm
        .action = strAction
        .method = strMethod
        .encoding = strEncoding
    End With
End Sub
```

Use the following subroutine to call the preceding subroutine.

```vba
Sub CallSetFormAction()
    Dim objForm As FPHTMLFormElement

    ActiveDocument.body.insertAdjacentHTML "beforeend", __
        "<form id="newform">"</form>"
    Set objForm = ActiveDocument.all.tags("form").Item("newform")
```
Call SetFormAction(objForm, "mailto: someone@microsoft.com", "post", "application/x-www-form-urlencoded")
End Sub
activeElement Property

Returns an object that represents the currently selected text or the location of the insertion point in the document.

```
expression.activeElement
```

**expression**  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The type of object returned depends on the location of the insertion point or the text selected in Microsoft FrontPage. For example, if you have a form button selected, the activeElement property returns an IHTMLInputButtonElement object; if you have a portion of text selected within a paragraph tag, it returns an IHTMLParaElement object; if you have several paragraphs selected, it returns an FPHTMLBody object. If you are unsure of the object that will be returned, you can use the IHTMLElement object to capture and manipulate the returned object, as shown in the example.
Example

The following example sets the className property for the active element if the active element is of the specified tag.

Sub SetClassForActiveElement(strTag As String, strClass As String)
    Dim objElement As IHTMLElement
    Dim strElement As String

    Set objElement = ActiveDocument. activeElement

    With objElement
        If LCase(.tagName) = LCase(strTag) Then
            If .className = "" Then
                .className = strClass
            End If
        End If
    End With
End Sub

Use the following subroutine to call the previous subroutine. This example assumes that you have a style called "blue" in the specified document or in a cascading style sheet reference in the specified document.

Sub CallSetClass()
    Call SetClassForActiveElement("p", "blue")
End Sub
Align Property

Sets or returns a `String` that represents how an object is aligned with adjacent text.

`expression.align`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

When applied to an FPHTMLFieldSetElement, FPHTMLIFrame, FPHTMLImg, FPHTMLObjectElement, IHTMLElement, HTMLEmbedElement, IHTMLFieldSetElement, IHTMLIFrameElement, IHTMLImgElement, IHTMLInputImage, IHTMLObjectElement, or IHTMLSelectElement object, the align property's String value can be one of the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>absbottom</td>
<td>Positions the bottom of the object with the absolute bottom of the surrounding text. The absolute bottom is equal to the baseline of the text minus the height of the largest descender in the text.</td>
</tr>
<tr>
<td>absmiddle</td>
<td>Positions the middle of the object with the absolute middle of the surrounding text. The absolute middle is the midpoint between the absolute bottom and top of the surrounding text.</td>
</tr>
<tr>
<td>baseline</td>
<td>Positions the bottom of the object with the baseline of the surrounding text.</td>
</tr>
<tr>
<td>bottom</td>
<td>Positions the bottom of the object with the bottom of the surrounding text. The bottom is equal to the baseline minus the standard height of a descender in the text.</td>
</tr>
<tr>
<td>left</td>
<td>Positions the object to the left of the surrounding text. All preceding and subsequent text flows to the right of the object.</td>
</tr>
<tr>
<td>middle</td>
<td>Positions the middle of the object in the middle of the surrounding text. The middle is the midpoint between the bottom and top of the surrounding text.</td>
</tr>
<tr>
<td>right</td>
<td>Positions the object to the right of the surrounding text. All subsequent text flows to the left of the object.</td>
</tr>
<tr>
<td>texttop</td>
<td>Positions the top of the object with the absolute top of the surrounding text. The absolute top is the baseline plus the height of the largest ascender in the text.</td>
</tr>
<tr>
<td>top</td>
<td>Positions the top of the object with the top of the text. The top of the text is the baseline plus the standard height of an ascender in the text.</td>
</tr>
</tbody>
</table>

When applied to an FPHTMLLegendElement, FPHTMLTableCaption, IHTMLLegendElement, or IHTMLTableCaption object, the align property's
String value can be one of the following:

**Value**  **Description**
bottom  Aligns bottom-center.
center  Aligns center.
left  Aligns left.
right  Aligns right.
top  Aligns top-center.

When applied to an **FPHTMLTable** or **IHTMLTable** object, the **align** property’s String value can be one of the following:

**Value** **Description**
left  Aligns to the left edge.
center  Aligns to the center.
right  Aligns to the right edge.

When applied to an **FPHTMLTableCol**, **IHTMLTableCol**, **FPHTMLDivElement**, **FPHTMLDivPosition**, **IHTMLDivElement**, **IHTMLDivPosition**, **FPHTMLHRElement**, **IHTMLHRElement**, **FPHTMLParaElement**, **IHTMLParaElement**, **FPHTMLTableCell**, **IHTMLTableCell**, **FPHTMLTableRow**, or **IHTMLTableRow** object, the **align** property’s String value can be one of the following:

**Value** **Description**
center  Aligns to the center.
justify  Aligns to the left and right edges.
left  Aligns to the left edge.
right  Aligns to the right edge.
Example

The following example aligns text around a specified image. This example takes a custom enumerated type called fpAlignType, also shown in the example below, and converts the specified constant to a String that it then uses to set the String value of the align property.

**Note** Custom enumerated types must be placed in the public declarations section of a code module.

Public Enum fpAlignType
    fpAlignAbsBottom
    fpAlignAbsMiddle
    fpAlignBaseline
    fpAlignBottom
    fpAlignLeft
    fpAlignMiddle
    fpAlignRight
    fpAlignTextTop
    fpAlignTop
End Enum

Sub AlignImageWithText(objImg As FPHTMLImg, eWhere As fpAlignType)
    Dim strAlign As String

    Select Case eWhere
        Case fpAlignAbsBottom
            strAlign = "absbottom"
        Case fpAlignAbsMiddle
            strAlign = "absmiddle"
        Case fpAlignBaseline
            strAlign = "baseline"
        Case fpAlignBottom
            strAlign = "bottom"
        Case fpAlignLeft
            strAlign = "left"
        Case fpAlignMiddle
            strAlign = "middle"
        Case fpAlignRight
            strAlign = "right"
        Case fpAlignTextTop
            strAlign = "texttop"
        Case fpAlignTop
            strAlign = "Top"
    End Select
End Sub
Use the following subroutine to call the preceding example. This subroutine assumes you have at least one `IMG` element in the specified document.

```vbscript
Sub CallAlignImageWithText()
    Dim objImage As FPHTMLImg

    Set objImage = ActiveDocument.all.tags("img").Item(0)

    Call AlignImageWithText(objImage, fpAlignBottom)
End Sub
```
**aLink Property**

Sets or returns a **String** that represents the color of all active links in the element.

`expression.aLink`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The aLink property sets and returns the same results as the aLinkColor property.
Example

The following example sets the active, viewed, and regular links, and sets the background color for the active document.

Function ChangeLinkColors(objDoc As FPHTMLDocument, Optional strALink As String, Optional strVLink As String, Optional strLink As String, Optional strBGColor As String) As Boolean

If strALink <> "" Or strVLink <> "" Or strLink <> "" Or strBGColor <> "" Then
   With objDoc.body
      .aLink = strALink
      .vLink = strVLink
      .link = strLink
      .bgColor = strBGColor
   End With
   ChangeLinkColors = True
Else
   ChangeLinkColors = False
End If
End Function

Use the following example to call the preceding function.

Sub CallChangeLinkColors()
   Call ChangeLinkColors(ActiveDocument, _
                         "blue", "yellow", "green", "aqua")
End Sub
**alinkColor Property**

Sets or returns a String that represents the value of the alinkColor attribute, which represents the color of all active links in the document.

`expression.alinkColor`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **alinkColor** property sets and returns the same results as the **aLink** property.
Example

The following example sets the active link for the active document.

Sub SetAlinkColor(objDoc As FPHTMLDocument, strColor As String)
    objDoc.alinkColor = strColor
End Sub

Use the following example to call the preceding subroutine.

Sub CallSetALinkColor()
    Call SetAlinkColor(ActiveDocument, "aqua")
End Sub
all Property

Returns an IHTMLElementCollection object that represents a reference to the collection of all elements contained within a specified object.

expression.all

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The IHTMLElementCollection object includes one element object for each valid HTML tag. If a valid tag has a matching end tag, both tags are represented by the same element object.

The collection returned when you use the all property of an FPHTMLDocument or IHTMLDocument object always includes a reference to the HTML, HEAD, TITLE, and BODY elements, regardless of whether the tags are present in the document.

If the document contains invalid or unknown tags, the collection includes one element object for each. Unlike valid end tags, unknown end tags are represented by their own IHTMLElement objects. The order of the elements in an IHTMLElementCollection object is the same order in which the elements occur in the HTML source code. Although the IHTMLElementCollection object indicates the order of the tags, it does not indicate hierarchy. This behavior is different from the children property, which returns an IHTMLElementCollection object that represents only the top-level elements that are direct descendants of the specified element and not any nested elements.
Example

The following example builds and inserts a table into the specified document. The function takes an IHTMLElement object (which can be any element in a document), two integers for the number of rows and columns, and a string for the id attribute of the table, and then returns an FPHTMLTable object that represents the newly created table.

Function InsertTable(objElement As IHTMLElement, intRows As Integer, intCols As Integer, strID As String) As FPHTMLTable
Dim objTable As FPHTMLTable
Dim strTable As String
Dim intRow As Integer
Dim intCol As Integer

strTable = "<TABLE id=""" & strID & """>" & vbCrLf
For intRow = 0 To intRows - 1
    strTable = strTable & vbTab & "<TR>" & vbCrLf
    For intCol = 0 To intCols - 1
        strTable = strTable & vbTab & vbTab & "<TD width=""50"">&nbsp;</TD>" & vbCrLf
    Next
    strTable = strTable & vbTab & "</TR>" & vbCrLf
Next
strTable = strTable & "</TABLE>"

If objElement.tagName = ActiveDocument.activeElement.tagName Then
    objElement.insertAdjacentHTML "afterend", strTable
Else
    objElement.insertAdjacentHTML "beforeend", strTable
End If

Set InsertTable = objElement.Document.all.tags("table").Item(1)
End Function

Use the following example to call the preceding subroutine.

Sub CallInsertTable()
    Dim objTbl As FPHTMLTable
    Set objTbl = InsertTable(ActiveDocument.activeElement, 4, 3, "testtbl")

    objTbl.bgColor = "red"
End Sub
End Sub
allowCustomization Property

Sets or returns a **Boolean** that represents whether properties on Web Parts within a zone that affect all users can be customized when designing a Web Part Page inside of the browser.

**Note** The **allowCustomization** property applies only to pages within a Web site based on Microsoft Windows SharePoint Services.

`expression.allowCustomization`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
allowPersonalization Property

Sets or returns a **Boolean** that represents whether individual users can customize their own properties for Web Parts within a zone when designing a Web Part Page inside of the browser.

**Note** The `allowPersonalization` property applies only to pages within a Web site based on Microsoft Windows SharePoint Services.

`expression.allowPersonalization`

`expression` Required. An expression that returns one of the objects in the Applies To list.
**alt Property**

Sets or returns a **String** that represents the text to display as an alternative to a graphic.

*expression*.alt

*expression*    Required. An expression that returns one of the objects in the Applies To list.
Remarks

The `alt` property sets or returns the text corresponding to the `alt` attribute of an IMG element. The text is used to replace the graphic for text-only browsers, to display in the window before the graphic has loaded, and to act as a ToolTip when the user rests the mouse pointer on the graphic.
Example

The following example returns a `Boolean` that indicates whether all IMG elements in the specified document have text assigned to the `alt` attribute.

Function `AllImagesHaveAltText(objDoc As FHTMLDocument) As Boolean`

Dim objImages As IHTMLElementCollection
Dim objImg As IHTMLElement
Dim intCount As Integer
Dim blnAlt As Boolean

Set objImages = objDoc.images

If objImages.Length > 0 Then
    For intCount = 0 To objImages.Length - 1
        Set objImg = objImages.Item(intCount)
        If objImg.alt = "" Then
            blnAlt = False
            Exit For
        Else
            blnAlt = True
        End If
    Next
Else
    blnAlt = True
End If

AllImagesHaveAltText = blnAlt
End Function

Use the following example to call the preceding function.

Sub CallAllImagesHaveAltText()
    MsgBox AllImagesHaveAltText(ActiveDocument)
End Sub
altKey Property

Returns **True** if the ALT key is pressed.

`expression.altKey`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Example

The following example displays a message if the user is pressing the ALT key when the user resizes the application window. This example must be placed in a class module, and it assumes you have declared an `FPHTMLWindow2` object variable called "winFP," as shown.

```vbnet
Private WithEvents winFP As FPHTMLWindow2

Private Sub winFP_onresize()
    Dim objEvent As IHTMLEventObj
    Set objEvent = winFP.event
    If objEvent.altKey = True Then
        MsgBox "You are pressing the ALT key."
    End If
End Sub
```
anchors Property

Returns an [IHTMLCollection](https://developer.mozilla.org/en-US/docs/Web/API/HTMLCollection) object that represents all of the bookmarks in a document. Bookmarks are `<A>` tags that have an associated name attribute.

`expression.anchors`

*expression* Required. An expression that returns one of the objects in the Applies To list.
Remarks

The anchors property returns only bookmarks, which are <A> tags that have the name attribute. Use the links property to return a collection of hyperlinks, which are <A> tags that have the href attribute.
Example

The following example returns an array of strings that represent the names of bookmarks in the specified document.

Function GetBookmarks(objDoc As FPHTMLDocument) As String()
Dim objAnchors As IHTMLCollection
Dim objAnchor As FPHTMLAnchorElement
Dim intCount As Integer
Dim strBookmarks() As String

Set objAnchors = objDoc.anchors
ReDim strBookmarks(objAnchors.Length - 1)

For intCount = 0 To objAnchors.Length - 1
    Set objAnchor = objAnchors.Item(intCount)
    If objAnchor.Name <> "" Then
        strBookmarks(intCount) = objAnchor.Name
    Else
        strBookmarks(intCount) = objAnchor.Id
    End If
Next

GetBookmarks = strBookmarks
End Function

Use the following code to call the preceding subroutine.

Sub CallGetBookmarks()
Dim strBookmarks() As String
Dim strBookmark As String
Dim intCount As Integer

On Error Resume Next

strBookmarks = GetBookmarks(ActiveDocument)

For intCount = 0 To UBound(strBookmarks)
    strBookmark = strBookmark & strBookmarks(intCount) & vbCrLf
Next

MsgBox strBookmark
End Sub
**appCodeName Property**

Returns a `String` that represents the Microsoft FrontPage application's code name.

`expression.appCodeName`

`expression` Required. An expression that returns an `FPHTMLNavigator` object.
Example

The following example displays the code name, application name, and current version number of the FrontPage application.

Sub GetBrowserInfo()
    Dim objBrowser As FPHTMLNavigator

    Set objBrowser = ActiveDocument.parentWindow.navigator

    With objBrowser
        MsgBox "Code Name = " & .appCodeName & vbCrLf & _
            "Application Name = " & .appName & vbCrLf & _
            "Version = " & .appVersion
    End With
End Sub
applets Property

Returns an IHTMLElementCollection object that represents the applets attached to a specified document.

expression.applets

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example takes an **FPHTMLDocument** object and returns a **String** array that represents the value of the **id** attribute for each of the APPLET elements in the specified document.

```vba
Function GetApplets(objDoc As FPHTMLDocument) As String()
    Dim objApplet As IHTMLElement
    Dim intCount As Integer
    Dim strApplets() As String

    If objDoc.applets.Length > 0 Then
        ReDim strApplets(objDoc.applets.Length - 1)
        For intCount = 0 To objDoc.applets.Length - 1
            Set objApplet = objDoc.applets.Item(intCount)
            strApplets(intCount) = objApplet.Id
        Next
    End If

    GetApplets = strApplets
End Function
```

Use the following example to call the preceding function.

```vba
Sub CallGetApplets()
    Dim strApplets() As String
    Dim intCount As Integer

    On Error Resume Next

    strApplets = GetApplets(ActiveDocument)

    For intCount = 0 To UBound(strApplets)
        MsgBox strApplets(intCount)
    Next
End Sub
```
appMinorVersion Property

The appMinorVersion property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**appName Property**

Returns a **String** that represents the name of the Microsoft FrontPage application.

*expression*.appName

*expression* Required. An expression that returns an **FPHTMLNavigator** object.
Example

The following example displays the code name, application name, and current version number of the FrontPage application.

Sub GetBrowserInfo()
    Dim objBrowser As FPHTMLNavigator

    Set objBrowser = ActiveDocument.parentWindow.navigator

    With objBrowser
        MsgBox "Code Name = " & .appCodeName & vbCrLf & _
            "Application Name = " & .appName & vbCrLf & _
            "Version = " & .appVersion
    End With
End Sub
**appVersion Property**

Returns a String that represents the version number of Microsoft FrontPage.

`expression.appVersion`

`expression` Required. An expression that returns an `FPHTMLNavigator` object.
Example

The following example displays the code name, application name, and current version number of the FrontPage application.

Sub GetBrowserInfo()
    Dim objBrowser As FPHTMLNavigator

    Set objBrowser = ActiveDocument.parentWindow.navigator

    With objBrowser
        MsgBox "Code Name = " & .appCodeName & vbCrLf & _
            "Application Name = " & .appName & vbCrLf & _
            "Version = " & .appVersion
    End With
End Sub
areas Property

Returns an IHTMLAreasCollection object that represents the collection of area settings in an FPHTMLMapElement object or an IHTMLMapElement object.

expression.areas

expression Required. An expression that returns an FPHTMLMapElement object or an IHTMLMapElement object.
Remarks

Use the Add method to add or remove FPHTMLAreaElement objects to or from the IHTMLAreasCollection object. Use the Item method or id property to reference an IHTMLAreaElement object.
Example

The following example returns a String array containing the values of the href property, which is equivalent to a hyperlink, for all the FPHTMLAreaElement objects in the specified FPHTMLMapElement object.

Function GetAreaHREF(objMap As FPHTMLMapElement) As String()
    Dim objArea As FPHTMLAreaElement
    Dim strAreas() As String
    Dim intCount As Integer

    ReDim strAreas(objMap.areas.Length - 1)

    For intCount = 0 To objMap.areas.Length - 1
        Set objArea = objMap.areas.Item(intCount)
        strAreas(intCount) = objArea.href
    Next

    GetAreaHREF = strAreas
End Function

Use the following example to call the preceding function. This example assumes that there is at least one MAP element in the specified document.

Sub CallGetAreaHREF()
    Dim objMap As FPHTMLMapElement
    Dim strHREFs() As String
    Dim intCount

    Set objMap = ActiveDocument.all.tags("map").Item(0)

    strHREFs = GetAreaHREF(objMap)

    For intCount = 0 To UBound(strHREFs)
        MsgBox strHREFs(intCount)
    Next
End Sub
assembly Property

Returns a **String** that represents the assembly attribute of the Register directive (i.e., "<%@ Register … %>") for the Web Part.

**expression.assembly**

**expression**  Required. An expression that returns one of the objects in the Applies To list.
atomic Property

The atomic property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**availHeight Property**

The `availHeight` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
The `availWidth` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
background Property

Sets or returns a **String** that represents up to five separate background properties of a specified object.

`expression.background`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

Use the **background** property to set one, all, or any of the values of the background properties at one time. The **background** property can specify values for up to five of the following space-delimited items, in any order:

- **color** Any value available to the **backgroundColor** property.
- **image** Any value available to the **backgroundImage** property. When you use the **background** property to specify a background image, surround the image filename with "url" and parentheses. For example, url(graphics/image.gif).
- **repeat** Any value available to the **backgroundRepeat** property.
- **attachment** Any value available to the **backgroundAttachment** property.
- **position** Any value available to the **backgroundPosition** property.
Example

The following example sets the background color or image for the body of the specified document. The strBackground argument can be a color name, a Red-Green-Blue (RGB) color, the path to an image file, or a string specifying any or all of the preceding space-delimited items.

Function SetBackground(objDoc As FHTMLDocument, _
                       strBackground As String) As Boolean

    On Error GoTo SetBackgroundColorError
    objDoc.body.Style.Background = strBackground
    SetBackgroundColor = True

ExitSetBackgroundColor:
    Exit Function

SetBackgroundColorError:
    SetBackgroundColor = False
    GoTo ExitSetBackgroundColor
End Function

Use the following example to call the preceding function. This example assumes that you have an image file called "picture.jpg" located in the same directory as the specified document.

Sub CallSetBackgroundColor()
    MsgBox SetBackgroundColor(ActiveDocument, _
                                "url(picture.jpg) no-repeat")
End Sub
**backgroundAttachment Property**

Returns or sets *String* that represents how the background image is displayed in the specified object. The object may be the entire document or an element within the document, such as a table. The `backgroundAttachment` property corresponds to the `background-attachment` property for a cascading style sheet.

`expression.backgroundAttachment`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value of the `backgroundAttachment` property can be one of the following:

- **scroll**
  Background image scrolls with the object as the document is scrolled. For a background image applied to the entire page, the background image scrolls when the page is scrolled.

- **fixed**
  Background image stays fixed within the viewable area of the object. For a background image applied to the entire page, the background image stays in a fixed position while the rest of the page scrolls up and down, left and right.

You can use the `background` property to set the `backgroundAttachment`, `backgroundColor`, `backgroundImage`, `backgroundPosition`, and `backgroundRepeat` properties. The following code shows what the example below would look like if you were using the `background` property to set each of these properties.

```javascript
ActiveDocument.body.Style.Background = "fixed " & _
    "url(graphics/chelan.jpg) blue top center no-repeat"
```
Example

The following example sets the background color and image settings for the active document.

With ActiveDocument.body.Style
    .backgroundAttachment = "fixed"
    .backgroundImage = "graphics/chelan.jpg"
    .backgroundColor = "blue"
    .backgroundPosition = "top center"
    .backgroundRepeat = "no-repeat"
End With
**backgroundColor Property**

Returns or sets a **String** that represents the background color for a specified object. The object may be the entire document or an element within the document, such as a table. The **backgroundColor** property corresponds to the **background-color** property for a cascading style sheet.

\[expression\].**backgroundColor**

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** for the **backgroundColor** property may be a color name or a red-green-blue (RGB) color value. For more information about setting colors, see the HTML Color Table.

The **backgroundColor** property is similar to the **bgColor** property; however, the **backgroundColor** property corresponds to the **backgroundcolor** attribute for the STYLE element, while the **bgColor** property corresponds to the value of the **bgcolor** attribute.

You can use the **background** property to set the **backgroundAttachment**, **backgroundColor**, **backgroundImage**, **backgroundPosition**, and **backgroundRepeat** properties. The following code shows what the example below would look like if you were using the **background** property to set each of these properties.

```javascript
ActiveDocument.body.Style.Background = "fixed " & _
    "url(graphics/chelan.jpg) blue center top no-repeat"
```
Example

The following example sets the background color and image settings for the active document.

With ActiveDocument.body.Style
    .backgroundAttachment = "fixed"
    .backgroundImage = "graphics/chelan.jpg"
    .backgroundColor = "blue"
    .backgroundPosition = "center top"
    .backgroundRepeat = "no-repeat"
End With
**backgroundImage Property**

Returns or sets a **String** that represents the path and file name for the background image for a specified object. The object may be the entire document or an element within the document, such as a table. The `backgroundImage` property corresponds to the `background-image` property for a cascading style sheet.

`expression.backgroundImage`

*expression* Required. An expression that returns one of the objects in the Applies To list.
Remarks

You can use the `background` property to set the `backgroundAttachment`, `backgroundColor`, `backgroundImage`, `backgroundPosition`, and `backgroundRepeat` properties. The following code shows what the example below would look like if you were using the `background` property to set each of these properties.

```plaintext
ActiveDocument.body.Style.Background = "fixed " & _
  "url(graphics/chelan.jpg) blue top center no-repeat"
```
Example

The following example sets the background color and image settings for the active document.

With ActiveDocument.body.Style
    .backgroundAttachment = "fixed"
    .backgroundImage = "graphics/chelan.jpg"
    .backgroundColor = "blue"
    .backgroundPosition = "top center"
    .backgroundRepeat = "no-repeat"
End With
**backgroundPosition Property**

Returns or sets a **String** that represents the position of the background image for an object, such as a document or table. If the **backgroundImage** property is not set, this property will do nothing. The **backgroundPosition** property corresponds to the **background-position** property for a cascading style sheet.

`expression.backgroundPosition`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value of the **backgroundPosition** property can be one or more of the following:

Sets the horizontal or vertical position of the background image to an exact location. Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). Positions the left and top edges of the background image at a specific spot on a page.

**length**

The following code sets the `backgroundPosition` property to 50px and 10px, which positions the left edge of the background image to 50 pixels from the left edge of the page and positions the top edge of the background image 10 pixels from the top edge of the page.

```
ActiveDocument.body.Style.backgroundPosition = "50px 10px"
```

Integer, followed by a percent sign (%). The value is a percentage of the width or height of the object.

**percentage**

For example, the following code sets the left edge of the background image to 10 percent of the width of the page and the top edge of the background image to 25 percent of the height of the page.

```
ActiveDocument.body.Style.backgroundPosition = "10% 25%"
```

Vertical alignment value consisting of one of the following:

- **top** Vertical alignment is at the top.
- **center** Vertical alignment is centered.
- **bottom** Vertical alignment is at the bottom.

If no setting is specified for **vAlignment**, the default setting is **top**.

**vAlignment**

If **center** is specified for **vAlignment** without an **hAlignment**, the background image is centered horizontally and vertically on the page.
Use the `backgroundPositionY` property to set the vertical position of a background image.

Horizontal alignment value consisting of one of the following:
- `left`  Horizontal alignment is to the left.
- `center` Horizontal alignment is centered.
- `right`  Horizontal alignment is to the right.

If no setting is specified for `hAlignment`, the default setting is `left`.

If `center` is specified for `hAlignment` without a `vAlignment`, the background image is centered horizontally and vertically on the page.

Use the `backgroundPositionX` property to set the vertical position of a background image.

**Note**  If one measurement is provided for `length` or `percentage`, then the horizontal position is set to the measurement provided; if two measurements are provided, then the horizontal and vertical positions are set to the measurements provided. The horizontal position is set equal to the first measurement; the vertical position is set equal to the second measurement. For example, the following code sets the horizontal position of the background image to 15 percent of the page width and the vertical position of the background image to 25 pixels from the top edge of the page.

`ActiveDocument.body.Style.backgroundPosition = "15% 25px"`

When the `backgroundRepeat` property is set to repeat (the default setting), the left and top edges of the image start at the specified position (or the default setting if no setting is specified) and then repeat around the image in all directions. For example, if you set the `backgroundPosition` property for a background image to `15px 50px`, the left edge of the background image will be 15 pixels from the left edge of the browser window and the top edge will be 50 pixels from the top edge of the browser window. However, with the `backgroundRepeat` property set to repeat, the right and bottom edges of the image will display above and to the left of the starting position as well as to the right and below of the starting position.
You can use the `background` property to set the `backgroundAttachment`, `backgroundColor`, `backgroundImage`, `backgroundPosition`, and `backgroundRepeat` properties. The following code shows what the example below would look like if you were using the `background` property to set each of these properties.

ActiveDocument.body.Style.Background = "fixed " & _
   "url(graphics/chelan.jpg) blue top center no-repeat"
Example

The following example sets the background color and image settings for the active document.

With ActiveDocument.body.Style
  .backgroundAttachment = "fixed"
  .backgroundImage = "graphics/chelan.jpg"
  .backgroundColor = "blue"
  .backgroundPosition = "top center"
  .backgroundRepeat = "no-repeat"
End With
**backgroundPositionX Property**

Sets or returns a **String** that represents the horizontal position of the background image.

*expression*.**backgroundPositionX**

**expression**  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value of the **backgroundPositionX** property can be one of the following:

- **length**
  - Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex).
  
  The following code sets the **backgroundPositionX** property to 50px which positions the left edge of the background image to 50 pixels from the left edge of the page.

    ```javascript
    ActiveDocument.body.Style.backgroundPositionX = "50px 10px"
    ```

- **percentage**
  - Integer, followed by a %. The value is a percentage of the width or height of the object.

  For example, the following code sets the left edge of the background image to 10 percent of the width of the page.

    ```javascript
    ActiveDocument.body.Style.backgroundPositionX = "10%"
    ```

- **hAlignment**
  - Horizontal alignment value consisting of one of the following:

    - **left** Horizontal alignment is to the left.
    - **center** Horizontal alignment is centered.
    - **right** Horizontal alignment is to the right.

Use the **backgroundPositionY** property to set the vertical position of the background image. You can use the **backgroundPosition** property to set the horizontal and vertical position of the background image. The following code shows what the example below would look like if you were using the **backgroundPosition** property to set the horizontal and vertical positions.

    ```javascript
    ActiveDocument.body.Style.backgroundPosition = "center top"
    ```
Example

The following example sets the position of the background image for the active document. This example assumes you have a background image set for the active document.

With ActiveDocument.body.Style
    .backgroundColorX = "center"
    .backgroundColorY = "top"
End With
backgroundPositionY Property

Returns or sets a String that represents the vertical position of the background image.

expression.addEventListener.backgroundPositionY

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value of the **backgroundPositionY** property can be one of the following:

- **length**: Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex).

  The following code sets the **backgroundPositionY** property 10px, which positions the top edge of the background image 10 pixels from the top edge of the page.

  ```javascript
  ActiveDocument.body.Style.backgroundPositionY = "10px"
  ```

- **percentage**: Integer, followed by a %. The value is a percentage of the width or height of the object.

  For example, the following code sets the top edge of the background image to 25 percent of the height of the page.

  ```javascript
  ActiveDocument.body.Style.backgroundPositionY = "10% 25%"
  ```

- **vAlignment**: Vertical alignment value consisting of one of the following:
  - **top**: Vertical alignment is at the top.
  - **center**: Vertical alignment is centered.
  - **bottom**: Vertical alignment is at the bottom.

Use the **backgroundPositionX** property to set the horizontal position of the background image. You can use the **backgroundPosition** property to set the horizontal and vertical position of the background image. The following code shows what the example below would look like if you were using the **backgroundPosition** property to set the horizontal and vertical positions.

  ```javascript
  ActiveDocument.body.Style.backgroundPosition = "center top"
  ```
**Example**

The following example sets the position of the background image for the active document. This example assumes you have a background image set for the active document.

```vba
With ActiveDocument.body.Style
    .backgroundPositionX = "center"
    .backgroundPositionY = "top"
End With
```
backgroundRepeat Property

Returns or sets a String that represents how a background image tiles for an object, such as a document or table. The backgroundRepeat property corresponds to the background-repeat property for a cascading style sheet.

expression.backgroundRepeat

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **backgroundRepeat** property and can contain one of the following **String** values:

- **repeat** Image is repeated horizontally and vertically. Default.
- **no-repeat** Image is not repeated.
- **repeat-x** Image is repeated horizontally a specified number of times.
- **repeat-y** Image is repeated vertically a specified number of times.

You can use the **background** property to set the **backgroundAttachment**, **backgroundColor**, **backgroundImage**, **backgroundPosition**, and **backgroundRepeat** properties. The following code shows what the example below would look like if you were using the **background** property to set each of these properties.

```javascript
ActiveDocument.body.Style.Background = "fixed " & _
  "url(graphics/chelan.jpg) blue top center no-repeat"
```
Example

The following example sets the background color and image settings for the active document.

With ActiveDocument.body.Style
    .backgroundAttachment = "fixed"
    .backgroundImage = "graphics/chelan.jpg"
    .backgroundColor = "blue"
    .backgroundPosition = "top center"
    .backgroundRepeat = "no-repeat"
End With
balance Property

Retrieves an **Integer** between -10,000 and +10,000 that represents how the volume of the background sound is divided between the left and right speakers.

*expression*.balance

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

A -10,000 value indicates that all sound is directed to the left speaker. A +10,000 value indicates that all sound is directed to the right speaker. A 0 value indicates that the sound is balanced between the left and right speakers.
Example

The following example creates a **BGSOUND** element in the specified document, if one doesn't already exist, and formats the **src**, **balance**, **loop**, and **volume** properties.

```
Sub InsertSound(ByVal objDoc As FPHTMLDocument, ByVal strSoundFile As String, _
ByVal intBalance As Integer, ByVal intLoop As Integer, _
ByVal intVolume As Integer)
  Dim objHead As IHTMLElement
  Dim objSound As FPHTMLBGsound
  Set objSound = objDoc.all.tags("bgsound").Item(0)
  If objSound Is Nothing Then
    Set objHead = objDoc.all.tags("head").Item(0)
    objHead.insertAdjacentHTML "beforeEnd", "<BGSOUND id="bgsound"
  Set objSound = objHead.all.tags("bgsound").Item("bgsound")
  End If
  With objSound
    .src = strSoundFile
    .balance = intBalance
    .loop = intLoop
    .volume = intVolume
  End With
End Sub
```

Use the following example to call the preceding subroutine. This example assumes that you have a sound file named "cashreg.wav" in a folder named "sounds" in the active Web site.

```
Sub CallInsertSound()
  Call InsertSound(objDoc:=ActiveDocument, strSoundFile:="sounds/cashreg.wav", _
                  intBalance:=0, intLoop:=1, intVolume:=0)
End Sub
```
BaseHref Property

Returns a **String** that represents the base location for links and references of a specified object.

`expression.BaseHref`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
**Example**

The following example displays a message with the value of the **BaseHref** property of the first object in the active document. This example assumes there is at least one **OBJECT** element in the active document.

```vba
Sub SetObjectBaseHref()
    Dim objObject As FPHTMLObjectElement
    Set objObject = ActiveDocument.all.tags("object").Item(0)
    MsgBox objObject.BaseHref
End Sub
```
behavior Property

Returns or sets a String that represents how text scrolls in a marquee.

expression.behavior

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** for the **behavior** property can be one of the following values:

- **scroll**: Marquee scrolls in the direction specified by the **direction** property. The text scrolls off the end and starts over. Default value.

- **alternate**: Marquee's scroll direction reverses when its content reaches the edge of the container.

- **slide**: Marquee scrolls in the direction specified by the **direction** property. The text scrolls to the end and stops.
Example

The following example inserts a MARQUEE element into the specified document at the insertion point, and then sets the scroll behavior and direction, the number of times to loop, and the height, width, and font and border formatting of the new element.

Sub InsertMarquee(ByVal objDoc As FPHTMLDocument, ByVal strText As String, _
strBehavior As String, strDirection As String, intLoop As Integer, _
strHeight As String, strWidth As String, strFont As String, _
blnBold As Boolean, blnItalic As Boolean, strBorderColor As String)

Dim objRange As IHTMLTxtRange
Dim objMarquee As FPHTMLMarqueeElement
Dim intCount As Integer
Dim strID As String

intCount = objDoc.all.tags("marquee").Length
strID = "marquee" & intCount + 1

Set objRange = objDoc.Selection.createRange

objRange.collapse
objRange.pasteHTML "<marquee id="" & strID & ""'></marquee>"

Set objMarquee = objDoc.all.tags("marquee").Item(CVar(strID))

With objMarquee
    .behavior = strBehavior
    .direction = strDirection
    .loop = intLoop
    .Height = strHeight
    .Width = strWidth
    With .Style
        .fontFamily = strFont
        If blnBold = True Then .fontWeight = "bold"
        If blnItalic = True Then .fontStyle = "italic"
        .Border = strBorderColor
    End With
    .innerText = strText
End With

End Sub

Use the following example to call the preceding subroutine.
Sub CallInsertMarquee()
    Call InsertMarquee(objDoc:=ActiveDocument, strText:="This is my
dir: alternate", strDir:="up", intLoop:="-1",
strWidth:="10%", strFont:="broadway", blnBold:=True, blnItal
strBorderColor:="red")
End Sub
**bgColor Property**

Returns or sets a **String**, specifying a color name or red-green-blue (RGB) value, that represents the color behind the content for a specified object.

\[ expression.bgColor \]

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **bgColor** property is similar to the **backgroundColor** property; however, the **bgColor** property corresponds to the value of the **bgcolor** attribute, while the **backgroundColor** property corresponds to the **backgroundcolor** attribute for the STYLE element.

For more information about setting colors, see the [HTML Color Table](#).
Example

The following example takes a `FPHTMLDocument` object and a string, and then sets the background color for the specified document if the background color is not already set.

```vba
Sub SetBackgroundColor(ByRef objDoc As FPHTMLDocument, ByRef strColor As String)
    If objDoc.bgColor = "NoColor" Then objdoc.bgColor = strColor
End Sub

Sub CallSetBackgroundColor()
    Call SetBackgroundColor(objDoc:=ActiveDocument, strColor:="aqua"
End Sub
```
**bgProperties Property**

Returns or sets a **String** that represents the properties of the background image.

\[expression.bgProperties\]

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The String value of the bgProperties property can be one of the following values:

"""  Background can scroll. Default.
fixed  Background is fixed and cannot scroll.

The bgProperties property is similar to the backgroundAttachment property.
Example

The following example sets the background image for the specified document and specifies the behavior of the background image.

Sub SetBackgroundImageProperties(ByRef objDoc As FPHTMLDocument, _
                          ByVal strImage As String, Optional ByVal strBehavior As String)
    With objDoc.body
        .Style.backgroundImage = strImage
        .bgProperties = strBehavior
    End With
End Sub

Use the following example to call the preceding subroutine. This example assumes that you have an image file called "background.gif."

Sub CallSetBackgroundImageProperties()
    Call SetBackgroundImageProperties(objDoc:=ActiveDocument, _
                                      strImage:="background.gif", strBehavior:="scroll")
End Sub
**bididir Property**

Returns or sets a String that represents the allowed direction of the text in a given text range or element. Read/write.

`expression.bididir`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

If the `bididir` property is set to "true", the text range allows bi-directional text rendering. A setting of "false" indicates that the text range doesn't allow bi-directional text rendering.
Example

The following example sets the bididir property of a given text range to "true", thus allowing text to appear either left-to-right or right-to-left.

Sub SetDirection()
    Dim objDoc As FHTMLDocument
    Dim objSs As IFPStyleState
    Dim objRng As IHTMLTxtRange

    Set objDoc = ActiveDocument
    objDoc.body.innerHTML = "<p><i><b>Heading 1</b></i></p>"
    Set objSs = objDoc.createStyleState
    Set objRng = objDoc.body.createTextRange

    With objSs
        .gather objRng
        .setProperty "background-color", vbYellow
        .bididir = "true"
        .Apply
    End With
End Sub
border Property

border property as it applies to the FPHTMLStyle, IHTMLRuleStyle, and IHTMLStyle objects.

Returns or sets a String that represents the border style for the specified object.

expression.border

expression  Required. An expression that returns one of the above objects.
Remarks

The **String** value for the **border** property can be one or more of the following space-delimited values:

- **width**  Any of the values available to the **borderWidth** property.
- **style**  Any of the values available to the **borderStyle** property.
- **color**  Any of the values available to the **borderColor** property.

**border property as it applies to the FPHTMLFrameBase, FPHTMLFrameElement, FPHTMLFrameSetSite, FPHTMLIFrame, FPHTMLImg, FPHTMLInputImage, FPHTMLTable, IHTMLFrameBase, IHTMLFrameSetElement, IHTMLImgElement, IHTMLInputImage, and IHTMLTable objects.**

Returns or sets a **Variant** that represents the width of the object's border in pixels.

**expression.border**

**expression**  Required. An expression that returns one of the above objects.
Remarks

Setting the `border` property to zero causes no border to be displayed.
Example

As it applies to the **FPHTMLStyle** and **IHTMLStyle** objects.

The following example inserts an opening and closing `<P>` tag to the active document before the closing BODY element, and then formats the paragraph to include a thick, red, dashed border.

```vba
Sub SetParagraphBorderProperties()
    Dim objPara As FPHTMLParaElement

    ActiveDocument.body.insertAdjacentHTML where:="beforeend",
    HTML:="<p id="newparagraph"'></p>"

    Set objPara = ActiveDocument.body.all.tags("p").Item("newparagraph")

    objPara.Style.**Border** = "thick red dashed"
End Sub
```

As it applies to the **FPHTMLImg** object.

The following example inserts an image at the end of the active document and sets the width of the image border to 10 pixels. This example assumes that you have an image named "venglobe.gif" in a folder named "images." If you do not, replace the path and file name with an image you do have.

```vba
Sub SetImageBorderProperties()
    Dim objImage As FPHTMLImg

    ActiveDocument.body.insertAdjacentHTML where:="beforeend",
    HTML:="<img src="images/venglobe.gif" id="venus"">"

    Set objImage = ActiveDocument.body.all.tags("img").Item("venus")

    objImage.**Border** = "10"
End Sub
```
**borderBottom Property**

Returns or sets a String that represents the properties for the bottom border of a specified object.

`expression.borderBottom`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** for the `borderBottom` property can be one or more of the following space-delimited values:

- **width** Any value available to the `borderBottomWidth` property.
- **style** Any value available to the `borderBottomStyle` property.
- **color** Any value available to the `borderBottomColor` property.
**Example**

The following example sets the bottom, left, right, and top borders for the specified element.

```vba
Sub SetTopBottomLeftRightBorders(ByRef objElement As IHTMLElement, _
    ByRef strBorder As String)
    With objElement.Style
        .borderBottom = strBorder
        .borderLeft = strBorder
        .borderRight = strBorder
        .borderTop = strBorder
    End With
End Sub
```

Use the following example to call the preceding subroutine.

```vba
Sub CallSetTopBottomLeftRightBorders()
    Call SetTopBottomLeftRightBorders(objElement:=ActiveDocument._
        .all.tags("p").Item(0), strBorder:="yellow dashed medium")
End Sub
```
borderBottomColor Property

Sets or returns a String, specifying a color name or red-green-blue (RGB) value, that represents the color of the bottom border of the specified object.

expression.borderBottomColor

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

For more information about setting colors, see the HTML Color Table.
Example

The following example sets the color, style, and size for the bottom, left, right, and top, borders of the specified IHTMLElement object.

Sub SetBorders(ByRef objElement As IHTMLElement, ByRef strColor As String, ByRef strStyle As String, ByRef strWidth As String)
    With objElement.Style
        'Sets the bottom border properties.
        .borderBottomColor = strColor
        .borderBottomStyle = strStyle
        .borderBottomWidth = strWidth

        'Sets the left border properties.
        .borderLeftColor = strColor
        .borderLeftStyle = strStyle
        .borderLeftWidth = strWidth

        'Sets the right border properties.
        .borderRightColor = strColor
        .borderRightStyle = strStyle
        .borderRightWidth = strWidth

        'Sets the top border properties.
        .borderTopColor = strColor
        .borderTopStyle = strStyle
        .borderTopWidth = strWidth
    End With
End Sub

Use the following example to call the preceding subroutine.

Sub CallSetBorders()  
    Call SetBorders(objElement:=ActiveDocument.body.all.tags("p") _
                    .Item(0), strColor:="blue", strStyle:="dashed", strWidth:="thick")
End Sub
**borderBottomStyle Property**

Returns or sets a **String** that represents the line style of the bottom border of a specified object.

```
expression.borderBottomStyle
```

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** for the **borderBottomStyle** property can be one of the following values:

- **none**  
  Border is not drawn.

- **dotted**  
  Border is a dotted line on the Macintosh platform with Microsoft Internet Explorer version 4.01 or later, and a solid line on the Microsoft Windows and Unix platforms.

- **dashed**  
  Border is a dashed line on the Macintosh platform with Internet Explorer 4.01 or later, and a solid line on the Windows and Unix platforms.

- **solid**  
  Border is a solid line.

- **double**  
  Border is a double line drawn on top of the background of the object. The sum of the two single lines and the space between equals the value of the **borderBottomWidth** property. The **borderBottomWidth** property must be at least 3 pixels wide to draw a double border.

- **groove**  
  3-D groove is drawn in colors based on the value of the border color.

- **ridge**  
  3-D ridge is drawn in colors based on the value of the border color.

- **inset**  
  3-D inset is drawn in colors based on the value of the border color.

- **outset**  
  3-D outset is drawn in colors based on the value of the border color.
Example

The following example sets the color, style, and size for the bottom, left, right, and top borders of the specified IHTMLElement object.

Sub SetBorders(objElement As IHTMLElement, strColor As String, _
               strStyle As String, strWidth As String)
    With objElement.Style
        'Sets the bottom border properties.
        .borderBottomColor = strColor
        .borderBottomStyle = strStyle
        .borderBottomWidth = strWidth

        'Sets the left border properties.
        .borderLeftColor = strColor
        .borderLeftStyle = strStyle
        .borderLeftWidth = strWidth

        'Sets the right border properties.
        .borderRightColor = strColor
        .borderRightStyle = strStyle
        .borderRightWidth = strWidth

        'Sets the top border properties.
        .borderTopColor = strColor
        .borderTopStyle = strStyle
        .borderTopWidth = strWidth
    End With
End Sub

Use the following example to call the preceding subroutine.

Sub CallSetBorders()
    Call SetBorders(ActiveDocument.body.all._
                    .tags("p").Item(0), "blue", "dashed", "thick")
End Sub
**borderBottomWidth Property**

Returns or sets a String that represents the width of the bottom border of a specified object.

`expression.borderBottomWidth`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value for the **borderBottomWidth** property can be one of the following:

- **medium**  Default width.
- **thin**    Less than the default width.
- **thick**   Greater than the default width.
- **width**   Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex).
Example

The following example sets the color, style, and size for the bottom, left, right, and top borders of the specified **IHTMLElement** object.

Sub SetBorders(ByRef objElement As IHTMLElement, ByRef strColor As String, ByRef strStyle As String, ByRef strWidth As String)

    With objElement.Style
        'Sets the bottom border properties.
        .borderBottomColor = strColor
        .borderBottomStyle = strStyle
        .**borderBottomWidth** = strWidth

        'Sets the left border properties.
        .borderLeftColor = strColor
        .borderLeftStyle = strStyle
        .borderLeftWidth = strWidth

        'Sets the right border properties.
        .borderRightColor = strColor
        .borderRightStyle = strStyle
        .borderRightWidth = strWidth

        'Sets the top border properties.
        .borderTopColor = strColor
        .borderTopStyle = strStyle
        .borderTopWidth = strWidth
    End With
End Sub

Use the following example to call the preceding subroutine.

Sub CallSetBorders()
    Call SetBorders(objElement:=ActiveDocument.body.all.tags("p") .Item(0), strColor:="blue", strStyle:="dashed", strWidth:="thick")
End Sub
borderCollapse Property

Returns or sets a **String** that determines if borders within a table appear collapsed.

```
expression.borderCollapse
```

**expression**  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example creates a table and sets the `borderCollapse` property to "true". Any adjacent cells now appear with collapsed borders.

Sub SetBorders()
    Dim objSS As IFPStyleState
    Dim objDoc As FPHTMLDocument
    Dim objRng As IHTMLTxtRange

    Set objDoc = ActiveDocument
    objDoc.body.innerHTML = "<table><tr><td>Cell 1</td><td>Cell 2</td></tr></table>"
    Set objSS = objDoc.createStyleState
    Set objRng = objDoc.body.createTextRange

    objSS.gather objRng
    objSS.borderCollapse = "true"
    objSS.borderBottomWidth.Value = 10
    objSS.backgroundColor = vbBlue
    objSS.apply
End Sub
**borderColor Property**

Returns or sets a **String**, specifying a color name or red-green-blue (RGB) value, that represents the border color of the object.

`expression.borderColor`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

For more information about setting colors, see the HTML Color Table.
Example

The following example modifies the border color, style, and width of the specified element.

Sub SetElementBorders(ByRef objStyle As FPHTMLStyle, ByRef strColor As String, ByRef strStyle As String, ByRef strWidth As String)
    With objStyle
        .borderColor = strColor
        .borderStyle = strStyle
        .borderWidth = strWidth
    End With
End Sub

Use the following example to call the preceding subroutine.

Sub CallSetElementBorders()
    Call SetElementBorders(objStyle:=ActiveDocument.all.tags("p").Item(0).Style, strColor:="red", strStyle:="double", strWidth:="thin")
End Sub
**borderColorDark Property**

Returns or sets a String, specifying a color name or red-green-blue (RGB) value, that represents the color for the darker of the two colors used to draw a three-dimensional border around a specified object.

*expression*.**borderColorDark**

*expression*   Required. An expression that returns one of the objects in the Applies To list.
Remarks

For more information about setting colors, see the HTML Color Table.
Example

The following example sets the light and dark colors for the 3-D border around the first table in the active document.

Sub Set3DBorder()
    Dim objTable As FPHTMLTable

    Set objTable = ActiveDocument.all.tags("table").Item(0)

    With objTable
        .Border = "7" 'Indicates the thickness of the border in pixels
        .borderColorLight = "blue"
        .BorderColorDark = "navy"
    End With
End Sub
**borderColorLight Property**

Returns or sets a **String**, specifying a color name or red-green-blue (RGB) value, that represents the color for the lighter of the two colors used to draw a three-dimensional border around a specified object.

*expression*.**borderColorLight**

*expression*    Required. An expression that returns one of the objects in the Applies To list.
Remarks

For more information about setting colors, see the [HTML Color Table](#).
Example

The following example sets the light and dark colors for the 3-D border around the first table in the active document.

Sub Set3DBorder()
    Dim objTable As FPHTMLTable

    Set objTable = ActiveDocument.all.tags("table").Item(0)

    With objTable
        .Border = "7" 'Indicates the thickness of the border in pixels
        .borderColorLight = "blue"
        .borderColorDark = "navy"
    End With
End Sub
**borderLeft Property**

Returns or sets a **String** that represents the properties for the left border of a specified object.

```
expression.borderLeft
```

*expression* Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** for the `borderLeft` property can be one or more of the following space-delimited values:

- **width**  Any value available to the `borderLeftWidth` property.
- **style**  Any value available to the `borderLeftStyle` property.
- **color**  Any value available to the `borderLeftColor` property.
**Example**

The following example sets the bottom, left, right, and top borders for the specified element.

```vba
Sub SetTopBottomLeftRightBorders(ByRef objElement As IHTMLElement, _
    ByRef strBorder As String)
    With objElement.Style
        .borderBottom = strBorder
        .borderLeft = strBorder
        .borderRight = strBorder
        .borderTop = strBorder
    End With
End Sub
```

Use the following example to call the preceding subroutine.

```vba
Sub CallSetTopBottomLeftRightBorders()
    Call SetTopBottomLeftRightBorders(objElement:=ActiveDocument._
        .all.tags("p").Item(0), strBorder:="yellow dashed medium")
End Sub
```
borderLeftColor Property

Returns or sets a String, specifying a color name or red-green-blue (RGB) value, that represents the color of the left border of the specified object.

expression.borderLeftColor

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

For more information about setting colors, see the HTML Color Table.
Example

The following example sets the color, style, and size for the bottom, left, right, and top borders of the specified **IHTMLElement** object.

Sub SetBorders(ByRef objElement As IHTMLElement, ByRef strColor As String, ByVal strStyle As String, ByVal strWidth As String)
    With objElement.Style
        'Sets the bottom border properties.
        .borderBottomColor = strColor
        .borderBottomStyle = strStyle
        .borderBottomWidth = strWidth

        'Sets the left border properties.
        .borderLeftColor = strColor
        .borderLeftStyle = strStyle
        .borderLeftWidth = strWidth

        'Sets the right border properties.
        .borderRightColor = strColor
        .borderRightStyle = strStyle
        .borderRightWidth = strWidth

        'Sets the top border properties.
        .borderTopColor = strColor
        .borderTopStyle = strStyle
        .borderTopWidth = strWidth
    End With
End Sub

Use the following example to call the preceding subroutine.

Sub CallSetBorders()
    Call SetBorders(objElement:=ActiveDocument.body.all.tags("p")
                   .Item(0), strColor:="blue", strStyle:="dashed", strWidth:="thick")
End Sub
**borderLeftStyle Property**

Returns or sets a **String** that represents the line style of the left border of a specified object.

*expression*.borderLeftStyle

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** for the **borderLeftStyle** property can be one of the following values:

- `none`  Border is not drawn.
  Border is a dotted line on the Macintosh platform with Microsoft Internet Explorer version 4.01 or later, and a solid line on the Microsoft Windows and Unix platforms.
- `dotted`  Border is a dashed line on the Macintosh platform with Internet Explorer 4.01 or later, and a solid line on the Windows and Unix platforms.
- `solid`  Border is a solid line.
  Border is a double line drawn on top of the background of the object. The sum of the two single lines and the space between equals the value of the **borderLeftWidth** property. The **borderLeftWidth** property must be at least 3 pixels wide in order to draw a double border.
- `double`  3-D groove is drawn in colors based on the value of the border color.
- `groove`  3-D ridge is drawn in colors based on the value of the border color.
- `ridge`  3-D inset is drawn in colors based on the value of the border color.
- `inset`  3-D outset is drawn in colors based on the value of the border color.
- `outset`
Example

The following example sets the color, style, and size for the bottom, left, right, and top borders of the specified *IHTMLElement* object.

Sub SetBorders(ByRef objElement As IHTMLElement, ByRef strColor As String, ByRef strStyle As String, ByRef strWidth As String)
With objElement.Style
  'Sets the bottom border properties.
  .borderBottomColor = strColor
  .borderBottomStyle = strStyle
  .borderBottomWidth = strWidth

  'Sets the left border properties.
  .borderLeftColor = strColor
  .**borderLeftStyle** = strStyle
  .borderLeftWidth = strWidth

  'Sets the right border properties.
  .borderRightColor = strColor
  .borderRightStyle = strStyle
  .borderRightWidth = strWidth

  'Sets the top border properties.
  .borderTopColor = strColor
  .borderTopStyle = strStyle
  .borderTopWidth = strWidth
End With
End Sub

Use the following example to call the preceding subroutine.

Sub CallSetBorders()
  Call SetBorders(objElement:=ActiveDocument.body.all.tags("p")
    .Item(0), strColor:="blue", strStyle:="dashed", strWidth:="thick")
End Sub
**borderLeftWidth Property**

Returns or sets a **String** that represents the width of the left border of a specified object.

`expression.borderLeftWidth`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value for the **borderLeftWidth** property can be one of the following:

- **medium**: Default width.
- **thin**: Less than the default width.
- **thick**: Greater than the default width.
- **width**: Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex).
Example

The following example sets the color, style, and size for the bottom, left, right, and top borders of the specified `IHTMLElement` object.

```vbscript
Sub SetBorders(ByRef objElement As IHTMLElement, ByRef strColor As String, ByVal strStyle As String, ByVal strWidth As String)
    With objElement.Style
        'Sets the bottom border properties.
        .borderBottomColor = strColor
        .borderBottomStyle = strStyle
        .borderBottomWidth = strWidth

        'Sets the left border properties.
        .borderLeftColor = strColor
        .borderLeftStyle = strStyle
        .borderLeftWidth = strWidth

        'Sets the right border properties.
        .borderRightColor = strColor
        .borderRightStyle = strStyle
        .borderRightWidth = strWidth

        'Sets the top border properties.
        .borderTopColor = strColor
        .borderTopStyle = strStyle
        .borderTopWidth = strWidth
    End With
End Sub
```

Use the following example to call the preceding subroutine.

```vbscript
Sub CallSetBorders()
    Call SetBorders(objElement:=ActiveDocument.body.all.tags("p") .Item(0), strColor:="blue", strStyle:="dashed", strWidth:="thick")
End Sub
```
**borderRight Property**

Returns or sets a **String** that represents the properties for the right border of a specified object.

`expression.borderRight`

*expression* Required. An expression that returns one of the objects in the Applies To list.
Remarks

The String for the `borderRight` property can be one or more of the following space-delimited values:

- **width** Any value available to the `borderRightWidth` property.
- **style** Any value available to the `borderRightStyle` property.
- **color** Any value available to the `borderRightColor` property.
Example

The following example sets the bottom, left, right, and top borders for the specified element.

Sub SetTopBottomLeftRightBorders(ByRef objElement As IHTMLElement, _
    ByRef strBorder As String)
    With objElement.Style
        .borderBottom = strBorder
        .borderLeft = strBorder
        .borderRight = strBorder
        .borderTop = strBorder
    End With
End Sub

Use the following example to call the preceding subroutine.

Sub CallSetTopBottomLeftRightBorders()
    Call SetTopBottomLeftRightBorders(objElement:=ActiveDocument._
        .all.tags("p").Item(0), strBorder:="yellow dashed medium")
End Sub
**borderRightColor Property**

Returns or sets a **String**, specifying a color name or red-green-blue (RGB) value, that represents the color of the right border of the specified object.

`expression.borderRightColor`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

For more information about setting colors, see the HTML Color Table.
**Example**

The following example sets the color, style, and size for the bottom, left, right, and top borders of the specified `IHTMLElement` object.

```vbs
Sub SetBorders(ByRef objElement As IHTMLElement, ByRef strColor As String, ByVal strStyle As String, ByVal strWidth As String)
    With objElement.Style
        'Sets the bottom border properties.
        .borderBottomColor = strColor
        .borderBottomStyle = strStyle
        .borderBottomWidth = strWidth

        'Sets the left border properties.
        .borderLeftColor = strColor
        .borderLeftStyle = strStyle
        .borderLeftWidth = strWidth

        'Sets the right border properties.
        .borderRightColor = strColor
        .borderRightStyle = strStyle
        .borderRightWidth = strWidth

        'Sets the top border properties.
        .borderTopColor = strColor
        .borderTopStyle = strStyle
        .borderTopWidth = strWidth
    End With
End Sub
```

Use the following example to call the preceding subroutine.

```vbs
Sub CallSetBorders()
    Call SetBorders(objElement:=ActiveDocument.body.all.tags("p")_ .Item(0), strColor:="blue", strStyle:="dashed", strWidth:="t"
End Sub
```
borderRightStyle Property

Returns or sets a **String** that represents the line style of the right border of a specified object.

*expression*.borderRightStyle

*expression*    Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** for the **borderRightStyle** property can be one of the following values:

- **none**: Border is not drawn.
- **dotted**: Border is a dotted line on the Macintosh platform with Microsoft Internet Explorer version 4.01 or later, and a solid line on the Microsoft Windows and Unix platforms.
- **dashed**: Border is a dashed line on the Macintosh platform with Internet Explorer 4.01 or later, and a solid line on the Windows and Unix platforms.
- **solid**: Border is a solid line.
- **double**: Border is a double line drawn on top of the background of the object. The sum of the two single lines and the space between equals the value of the **borderRightWidth** property. The **borderRightWidth** property must be at least 3 pixels wide in order to draw a double border.
- **groove**: 3-D groove is drawn in colors based on the value of the border color.
- **ridge**: 3-D ridge is drawn in colors based on the value of the border color.
- **inset**: 3-D inset is drawn in colors based on the value of the border color.
- **outset**: 3-D outset is drawn in colors based on the value of the border color.
Example

The following example sets the color, style, and size for the bottom, left, right, and top borders of the specified IHTMLElement object.

Sub SetBorders(ByRef objElement As IHTMLElement, ByRef strColor As String, ByRef strStyle As String, ByRef strWidth As String)
    With objElement.Style
        'Sets the bottom border properties.
        .borderBottomColor = strColor
        .borderBottomStyle = strStyle
        .borderBottomWidth = strWidth

        'Sets the left border properties.
        .borderLeftColor = strColor
        .borderLeftStyle = strStyle
        .borderLeftWidth = strWidth

        'Sets the right border properties.
        .borderRightColor = strColor
        .borderRightStyle = strStyle
        .borderRightWidth = strWidth

        'Sets the top border properties.
        .borderTopColor = strColor
        .borderTopStyle = strStyle
        .borderTopWidth = strWidth
    End With
End Sub

Use the following example to call the preceding subroutine.

Sub CallSetBorders()
    Call SetBorders(objElement:=ActiveDocument.body.all.tags("p")
                   .Item(0), strColor:="blue", strStyle:="dashed", strWidth:="thick")
End Sub
**borderRightWidth Property**

Returns or sets a `String` that represents the width of the right border of a specified object.

*expression*.borderRightWidth

*expression*   Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value for the **borderRightWidth** property can be one of the following:

- **medium**: Default width.
- **thin**: Less than the default width.
- **thick**: Greater than the default width.
- **width**: Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex).
**Example**

The following example sets the color, style, and size for the bottom, left, right, and top borders of the specified `IHTMLElement` object.

```vba
Sub SetBorders(ByRef objElement As IHTMLElement, ByRef strColor As String, ByRef strStyle As String, ByRef strWidth As String)

    With objElement.Style
        'Sets the bottom border properties.
        .borderBottomColor = strColor
        .borderBottomStyle = strStyle
        .borderBottomWidth = strWidth

        'Sets the left border properties.
        .borderLeftColor = strColor
        .borderLeftStyle = strStyle
        .borderLeftWidth = strWidth

        'Sets the right border properties.
        .borderRightColor = strColor
        .borderRightStyle = strStyle
        .borderRightWidth = strWidth

        'Sets the top border properties.
        .borderTopColor = strColor
        .borderTopStyle = strStyle
        .borderTopWidth = strWidth
    End With
End Sub
```

Use the following example to call the preceding subroutine.

```vba
Sub CallSetBorders()
    Call SetBorders(objElement:=ActiveDocument.body.all.tags("p") _
                   .Item(0), strColor:="blue", strStyle:="dashed", strWidth:="thick")
End Sub
```
**borderStyle Property**

Returns or sets a **String** that represents the line style of the border of a specified object.

`expression.borderStyle`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** for the **borderStyle** property can be one of the following values:

- **none**: Border is not drawn.
- **dotted**: Border is a dotted line on the Macintosh platform with Microsoft Internet Explorer version 4.01 or later, and a solid line on the Microsoft Windows and Unix platforms.
- **dashed**: Border is a dashed line on the Macintosh platform with Internet Explorer 4.01 or later, and a solid line on the Windows and Unix platforms.
- **solid**: Border is a solid line.
- **double**: Border is a double line drawn on top of the background of the object. The sum of the two single lines and the space between equals the value of the **borderWidth** property. The **borderWidth** property must be at least 3 pixels wide to draw a double border.
- **groove**: 3-D groove is drawn in colors based on the value in the **borderColor** property.
- **ridge**: 3-D ridge is drawn in colors based on the value in the **borderColor** property.
- **inset**: 3-D inset is drawn in colors based on the value in the **borderColor** property.
- **outset**: 3-D outset is drawn in colors based on the value in the **borderColor** property.
Example

The following example sets the border width, style, and color of the specified element.

Sub SetElementBorders(ByRef objStyle As FPHTMLStyle)
    With objStyle
        .borderWidth = "thick"
        .borderStyle = "double"
        .borderColor = "red"
    End With
End Sub

Use the following example to call the preceding subroutine.

Sub CallSetElementBorders()
    Call SetElementBorders(objStyle:=ActiveDocument.all.tags("p").Item(1).Style)
End Sub
**borderTop Property**

Returns or sets a **String** that represents the properties for the top border of a specified object.

*expression*.**borderTop**

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The String for the `borderTop` property can be one or more of the following space-delimited values:

- **width** Any value available to the `borderTopWidth` property.
- **style** Any value available to the `borderTopStyle` property.
- **color** Any value available to the `borderTopColor` property.
Example

The following example sets the bottom, left, right, and top borders for the specified element.

Sub SetTopBottomLeftRightBorders(ByRef objElement As IHTMLElement, _
    ByVal strBorder As String)
    With objElement.Style
        .borderBottom = strBorder
        .borderLeft = strBorder
        .borderRight = strBorder
        .borderTop = strBorder
    End With
End Sub

Use the following example to call the preceding subroutine.

Sub CallSetTopBottomLeftRightBorders()
    Call SetTopBottomLeftRightBorders(objElement:=ActiveDocument._
        .all.tags("p").Item(0), strBorder:="yellow dashed medium")
End Sub
borderTopColor Property

Returns or sets a String, specifying a color name or red-green-blue (RGB) value, that represents the color of the top border of the specified object.

expression.borderTopColor

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

For more information about setting colors, see the [HTML Color Table](#).
Example

The following example sets the color, style, and size for the bottom, left, right, and top borders of the specified IHTMLElement object.

Sub SetBorders(ByRef objElement As IHTMLElement, ByRef strColor As String, ByRef strStyle As String, ByRef strWidth As String)
    With objElement.Style
        'Sets the bottom border properties.
        .borderBottomColor = strColor
        .borderBottomStyle = strStyle
        .borderBottomWidth = strWidth

        'Sets the left border properties.
        .borderLeftColor = strColor
        .borderLeftStyle = strStyle
        .borderLeftWidth = strWidth

        'Sets the right border properties.
        .borderRightColor = strColor
        .borderRightStyle = strStyle
        .borderRightWidth = strWidth

        'Sets the top border properties.
        .borderTopColor = strColor
        .borderTopStyle = strStyle
        .borderTopWidth = strWidth
    End With
End Sub

Use the following example to call the preceding subroutine.

Sub CallSetBorders()
    Call SetBorders(objElement:=ActiveDocument.body.all.tags("p")_.
                   Item(0), strColor:="blue", strStyle:="dashed", strWidth:="thick")
End Sub
**borderTopStyle Property**

Returns or sets a `String` that represents the line style of the top border of a specified object.

`expression.borderTopStyle`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** for the `borderTopStyle` property can be one of the following values:

- **none**  Border is not drawn.
  - Border is a dotted line on the Macintosh platform with Microsoft Internet Explorer version 4.01 or later, and a solid line on the Microsoft Windows and Unix platforms.
- **dotted** Border is a dashed line on the Macintosh platform with Internet Explorer 4.01 or later, and a solid line on the Windows and Unix platforms.
- **dashed** Border is a solid line.
  - Border is a double line drawn on top of the background of the object. The sum of the two single lines and the space between equals the value of the `borderTopWidth` property. The `borderTopWidth` property must be at least 3 pixels wide in order to draw a double border.
- **solid**
- **double**
- **groove** 3-D groove is drawn in colors based on the value of the border color.
- **ridge** 3-D ridge is drawn in colors based on the value of the border color.
- **inset** 3-D inset is drawn in colors based on the value of the border color.
- **outset** 3-D outset is drawn in colors based on the value of the border color.
Example

The following example sets the color, style, and size for the bottom, left, right, and top borders of the specified IHTMLElement object.

Sub SetBorders(ByRef objElement As IHTMLElement, ByRef strColor As String, ByRef strStyle As String, ByRef strWidth As String)

    With objElement.Style
        'Sets the bottom border properties.
        .borderBottomColor = strColor
        .borderBottomStyle = strStyle
        .borderBottomWidth = strWidth

        'Sets the left border properties.
        .borderLeftColor = strColor
        .borderLeftStyle = strStyle
        .borderLeftWidth = strWidth

        'Sets the right border properties.
        .borderRightColor = strColor
        .borderRightStyle = strStyle
        .borderRightWidth = strWidth

        'Sets the top border properties.
        .borderTopColor = strColor
        .borderTopStyle = strStyle
        .borderTopWidth = strWidth
    End With
End Sub

Use the following example to call the preceding subroutine.

Sub CallSetBorders()
    Call SetBorders(objElement:=ActiveDocument.body.all.tags("p") .Item(0), strColor:="blue", strStyle:="dashed", strWidth:="thick")
End Sub
**borderTopWidth Property**

Returns or sets a **String** that represents the width of the top border of a specified object.

\[ expression \cdot \text{borderTopWidth} \]

*expression* Required. An expression that returns one of the objects in the Applies To list.
Remarks

The String value for the borderTopWidth property can be one of the following:

- medium: Default width.
- thin: Less than the default width.
- thick: Greater than the default width.
- width: Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex).
Example

The following example sets the color, style, and size for the bottom, left, right, and top borders of the specified IHTMLElement object.

Sub SetBorders(objElement As IHTMLElement, strColor As String, _
    strStyle As String, strWidth As String)
    With objElement.Style
        'Sets the bottom border properties.
        .borderBottomColor = strColor
        .borderBottomStyle = strStyle
        .borderBottomWidth = strWidth

        'Sets the left border properties.
        .borderLeftColor = strColor
        .borderLeftStyle = strStyle
        .borderLeftWidth = strWidth

        'Sets the right border properties.
        .borderRightColor = strColor
        .borderRightStyle = strStyle
        .borderRightWidth = strWidth

        'Sets the top border properties.
        .borderTopColor = strColor
        .borderTopStyle = strStyle
        .borderTopWidth = strWidth
    End With
End Sub

Use the following example to call the preceding subroutine.

Sub CallSetBorders()
    Call SetBorders(ActiveDocument.body.all tags("p").Item(0), "blue", "dashed", "thick")
End Sub
**borderWidth Property**

Returns or sets a *String* that represents the thickness of the left, right, top, and bottom borders for the specified object.

`expression.borderWidth`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

You can set the `borderWidth` property to one of the following `String` values:

- **medium** Border is the default width.
- **thin** Border is less than the default width.
- **thick** Border is greater than the default width.
- **value** Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex).
Example

The following example modifies the border color, style, and width of the specified element.

Sub SetElementBorders(ByRef objStyle As FPHTMLStyle, ByRef strColor As String, ByRef strStyle As String, ByRef strWidth As String)
    With objStyle
        .borderColor = strColor
        .borderStyle = strStyle
        .borderWidth = strWidth
    End With
End Sub

Use the following example to call the preceding subroutine.

Sub CallSetElementBorders()
    Call SetElementBorders(objStyle:=ActiveDocument.all.tags("p") _
                            .Item(0).Style, strColor:="red", strStyle:="double", strWidth:="thin")
End Sub
bottom Property

Returns or sets an **IFPStyleLength** object that defines the bottom length of the element. Read/write.

*expression*.bottom

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example creates a new **IFPStyleState** object and sets the **value** and **units** properties of its associated **IFPStyleLength** object.

```vba
Sub SetStyleLength()
    Dim objSS As IFPStyleState
    Dim objDoc As FPHTMLDocument
    Dim objRng As IHTMLTxtRange
    Dim objSL As IFPStyleLength

    Set objDoc = Application.ActiveDocument
    objDoc.body.innerHTML = "<b>Heading 1</b>"
    Set objSS = objDoc.createStyleState
    Set objRng = objDoc.body.createTextRange

    objSS.gather objRng
    Set objSL = objSS.bottom
    objSL.units = fpLengthUnitsPT
    objSL.Value = 4
    objSS.bottom = objSL
    objSS.Apply

End Sub
```
bottomMargin Property

Returns or sets a String that represents the number of pixels for the bottom margin of the page.

expression.bottomMargin

expression   Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the bottom, left, right, and top margins to 50 pixels.

Sub SetMargins()
    Dim objBody As FPHTMLBody

    Set objBody = ActiveDocument.body

    With objBody
        .bottomMargin = "50"
        .leftMargin = "50"
        .rightMargin = "50"
        .topMargin = "50"
    End With
End Sub
bufferDepth Property

The bufferDepth property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**button Property**

Returns a `Long` that represents whether the user is pressing the mouse button.

`expression.button`

*expression* Required. An expression that returns one of the objects in the Applies To list.
Remarks

This property is generally used with the onmousedown, onmouseup, and onmousemove events.
Example

The following example displays a message when a user moves the mouse pointer over the application window, telling the user whether the mouse button is being clicked. This example must be placed in a class module.

Note Place the winFP and docFP variables declared using the WithEvents keyword in the General Declarations section of the class module.

```vba
Private WithEvents winFP As FPHTMLWindow2
Private WithEvents docFP As FPHTMLDocument

Private Sub docFP_onmousemove()
    Dim objEvent As IHTMLEventObj
    Set objEvent = winFP.event
    Select Case objEvent.Button
        Case 0
            MsgBox "No mouse button is being clicked."
        Case 1
            MsgBox "You are clicking your mouse button."
    End Select
End Sub
```
cancelBubble Property

Returns or sets a **Boolean** that represents whether the current event should bubble up the hierarchy of event handlers. **True** disables bubbling for this event, preventing the next event handler in the hierarchy from receiving the event.

`expression.cancelBubble`

- `expression` Required. An expression that returns one of the objects in the Applies To list.
Remarks

Using this property to cancel bubbling for an event does not affect subsequent events.
Example

The following example displays a message when a user resizes the application window, and then cancels bubbling the event up the hierarchy of events if the user clicks the **No** button in the message box. This example must be placed in a class module.

**Note** Place the winFP object variable declared with the **WithEvents** keyword in the General Declarations section of the class module.

```vbscript
Private WithEvents winFP As FPHTMLWindow2

Private Sub winFP_onresize()
    Dim objEvent As IHTMLEventObj
    Dim intResponse As Integer

    Set objEvent = window.event

    intResponse = MsgBox("Are you having fun?", vbYesNo)

    If intResponse = vbNo Then
        objEvent.cancelBubble = True
    Else
        objEvent.cancelBubble = False
    End If
End Sub
```
caption Property

Returns an IHTMLTableCaption object that represents a CAPTION element in a table.

expression.caption

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example adds a CAPTION element to the first table in the active document and sets the caption text.

```vbscript
Dim objTable As FPHTMLTable
Dim objCaption As FPHTMLTableCaption

Set objTable = ActiveDocument.all.tags("table").Item(0)
objTable.insertAdjacentHTML "afterbegin", "<caption></caption>

Set objCaption = objTable.Caption
objCaption.innerText = "Table Caption Text"
```
cellFormattingID Property

Returns a **Long** that represents the value of the `mscellformattingtableid` attribute as it appears in the post-comment tag, for example, `<--MSCellFormattingTableID="ID" -->`.

```expression.cellFormattingID```

**expression** Required. An expression that returns an **FPLayoutRegion** object.
cellIndex Property

Returns a Long that represents the position of the specified cell in a row.

expression.cellIndex

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example formats the background color of the third cell based on the position of the cell in the table.

Sub ChangeColorOfCell(ByRef objCell As FPHTMLTableCell)
    Select Case objCell.CellIndex
        Case 0
            objCell.bgColor = "red"
        Case 1
            objCell.bgColor = "blue"
        Case 2
            objCell.bgColor = "yellow"
        Case Else
            objCell.bgColor = "navy"
    End Select
End Sub

Use the following example to call the preceding subroutine.

Sub CallChangeColorOfCell()
    If ActiveDocument.activeElement.tagName = "td" Then
        Call ChangeColorOfCell(objCell:=ActiveDocument.activeElement)
    End If
End Sub
cellPadding Property

Returns or sets a **String** that represents the amount of space between the border of the cell and the contents of the cell.

`expression.cellPadding`

*expression* Required. An expression that returns one of the objects in the Applies To list.
Remarks

The String for the cellPadding property can be one of the following values:

- **padding**: Integer that specifies, in pixels, the amount of space between the border and the content.
- **percentage**: Integer, followed by a %. The value is a percentage of the available amount of space between the border and the content.
Example

The following example sets the padding and spacing for the specified table.

Sub ChangeTableSpacing(ByRef objTable As FPHTMLTable, _
                    ByRef strPadding As String, ByRef strSpacing As String)
    With objTable
        .cellPadding = strPadding
        .cellSpacing = strSpacing
    End With
End Sub

Use the following example to call the preceding subroutine. This example assumes that there is at least one table in the active document.

Sub CallChangeTableSpacing()
    Call ChangeTableSpacing(objTable:=ActiveDocument.all._
                            .tags("table").Item(0), strPadding:="5", strSpacing:="10")
End Sub
**cells Property**

Returns an **IHTMLElementCollection** object that represents a collection of cells in a specified table row or in an entire table.

*expression.cells*

*expression*  Required. An expression that returns one of the objects in the Applies To list.
**Example**

The following example inserts the specified text into each cell in the specified table.

```vbscript
Sub AddTextToTable(ByRef objTable As FPHTMLTable, ByRef strText As String)
    Dim intRow As Integer
    Dim intCell As Integer

    For intRow = 0 To objTable.rows.Length - 1
        For intCell = 0 To objTable.rows.Item(intRow).cells.Length - 1
            objTable.rows.Item(intRow).cells.Item(intCell).innerText = strText
        Next
    Next
End Sub
```

Use the following example to call the preceding subroutine. This example assumes that you have at least one table in the active document.

```vbscript
Sub CallAddTextToTable()
    Call AddTextToTable(objTable:=ActiveDocument.all.tags("Table").Item(0), strText:="Test")
End Sub
```
**cellSpacing Property**

Sets or returns a **String** that represents the amount of space between cells in a table.

*expression.cellSpacing*

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** for the **cellSpacing** property can be one of the following values:

padding  Integer that specifies, in pixels, the amount of space between cells.
percentage Integer, followed by a %. The value is a percentage of the available amount of space between the border and the content.
Example

The following example sets the padding and spacing for the specified table.

Sub ChangeTableSpacing(ByRef objTable As FPHTMLTable, _
    ByRef strPadding As String, ByRef strSpacing As String)
    With objTable
        .cellPadding = strPadding
        .cellSpacing = strSpacing
    End With
End Sub

Use the following example to call the preceding subroutine. This example assumes that there is at least one table in the active document.

Sub CallChangeTableSpacing()
    Call ChangeTableSpacing(objTable:=ActiveDocument.all._
        .tags("table").Item(0), strPadding:="5", strSpacing:="10")
End Sub
**charset Property**

Returns or sets a **String** that represents the character set for an external style sheet.

`expression.charset`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

For information about character sets, see HTML Character Sets.
Example

The following example inserts a <META> tag that contains the character set to use for the active document.

```vba
Sub InsertCharset(ByRef objDoc As FPHTMLDocument, ByRef strID As String, ByRef strHTTP As String, ByRef strContent As String, ByRef strCharset As String)
    Dim objMeta As FPHTMLMetaElement
    objDoc.all.tags("head").Item(0).innerHTML = "</META id="" & strID & """>
    Set objMeta = ActiveDocument.all.tags("meta").Item(CVar(strID))
    With objMeta
        .httpEquiv = strHTTP
        .content = strContent
        .Charset = strCharset
    End With
End Sub
```

Use the following example to call the preceding subroutine.

```vba
Sub CallInsertCharset()
    Call InsertCharset(objDoc:=ActiveDocument, strID:="iso_content", strHTTP:="Content-Type", strContent:="text/html", strCharset
End Sub
```
children Property

Returns an IHTMLElementCollection collection that represents a collection of elements that are direct descendants of a specified object.

expression.children

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

If the document contains invalid or unknown tags, the IHTMLCollection object includes one element object for each. Unlike valid end tags, unknown end tags are represented by their own IHTMLElement objects. When you use the children property, the order of the elements in an object indicates the hierarchy and includes only the top-level elements that are direct descendants of the specified element and not any nested elements. This behavior is different from the all property, which returns an IHTMLCollection object that represents all tags regardless of hierarchy.
Example

The following example specifies the title of the page and the contents of the first heading to the specified string. Use the CallSetTitleAndFirstHeading subroutine following to call this function.

Function SetTitleAndFirstHeading(ByRef objBody As FPHTMLBody, ByVal strTitle As String) As Boolean
    Dim objHeading As IHTMLElement
    setTitleAndFirstHeading = False
    Set objBody = ActiveDocument.body
    If InStr(1, UCase(objBody.innerHTML), UCase("h1")) < 1 Then
        objBody.insertAdjacentHTML "afterBegin", "<h1>" & strTitle "</h1>
    Else
        Set objHeading = objBody.Children.tags("h1").Item(0)
        objHeading.innerText = strTitle
    End If
    objBody.Document.Title = strTitle
    setTitleAndFirstHeading = True
End Function

The following example calls the preceding SetTitleAndFirstHeading function and displays a message indicating whether the function was successful.

Sub CallSetTitleAndFirstHeading()
    Dim blnResponse as Boolean
    blnResponse = SetTitleAndFirstHeading(objBody:=ActiveDocument.body, strTitle:="FrontPage Developer's Home Page")
    If blnResponse = True Then
        MsgBox "You have successfully changed the title " & _
        "and first heading of the current page."
    Else
        MsgBox "Title and first heading were not changed."
    End If
End Sub
clear Property

Returns or sets a String that represents whether the object allows floating objects on its left and/or right sides, so that the next line of text is displayed past the floating objects.

expression.cleart

text

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

The *String* for the *clear* property can be one of the following values:

- none  Floating objects are allowed on both sides.
- left Object is moved below any floating object on the left side.
- right Object is moved below any floating object on the right side.
- both Object is moved below any floating object.
The following example inserts a new paragraph into the active document before the closing BODY element and sets the **clear** attribute to allow floating objects to be displayed on the right side of the paragraph.

```vbscript
Sub SetClearProperty()
    Dim objPara As FPHtmlParaElement
    ActiveDocument.body.insertAdjacentHTML where:="beforeend", HTML:="<p id="newpara">Test paragraph</p>"
    Set objPara = ActiveDocument.body.all.tags("P").Item("newpara")
    objPara.clear = "right"
End Sub
```
clearLeft Property

Returns or sets a **Boolean** that determines which sides of an element's box or boxes may not be adjacent to an earlier floating box. Read/write.

`expression.clearLeft`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

Use the `clearRight` property to clear the right side of a table cell.
**Example**

The following example creates a table and sets the `clearLeft` property of the cell in the right column to **True**. The cell or column to the left of this cell will now appear empty.

Sub ClearLeftSide()
    Dim objSS As IFPStyleState
    Dim objLine1 As IHTMLElement
    Dim objLine2 As IHTMLElement
    Dim strHTML As String

    strHTML = "<table><tr><td><h1>This is line 1</h1></td>" & "<td>This is line 2</td></tr></table>"

    With ActiveDocument
        .body.innerHTML = strHTML
        Set objLine1 = .all.tags("td").Item(0)
        Set objLine2 = .all.tags("td").Item(1)
        Set objSS = .createStyleState
    End With

    With objSS
        .GatherFromElement objLine2
        .clearLeft = True
        .textDecorationUnderline = True
        .Apply
    End With
End Sub
clearRight Property

Returns or sets a **Boolean** that determines which sides of an element's box or boxes may not be adjacent to an earlier floating box. Read/write.

`expression.clearRight`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Remarks

Use the `clearLeft` property to clear the right side of a table cell.
Example

The following example creates a table and then sets the `clearRight` property of the cell on the left hand side of the table. As a result, the right hand side of the table adjacent to the cell will be empty.

```vbscript
Sub ClearRightSide()
  Dim objSS As IFPStyleState
  Dim objLine1 As IHTMLElement
  Dim objLine2 As IHTMLElement
  Dim strHTML As String

  strHTML = "<table><tr><td><h1>This is line 1</h1></td>" _
           & "<td>This is line 2</td></tr></table>"

  With ActiveDocument
    .body.innerHTML = strHTML
    Set objLine1 = .all.tags("td").Item(0)
    Set objLine2 = .all.tags("td").Item(1)
    Set objSS = .createStyleState
    End With
  End With

  With objSS
    .GatherFromElement objLine2
    .clearRight = True
    .textDecorationUnderline = True
    .Apply
  End With
End Sub
```
**clientHeight Property**

Returns a **Long** that represents the height in pixels of a specified object without any margin, border, scroll bar, or padding settings that have been applied.

`expression.clientHeight`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example displays a message containing the height, width, top, and left measurements of the body of the active document.

Sub GetClientInfo()
    Dim objBody As FPHTMLBody

    Set objBody = ActiveDocument.body

    With objBody
        MsgBox "height" & vbTab & .clientHeight & vbCrLf & _
               "width" & vbTab & .clientWidth & vbCrLf & _
               "top" & vbTab & .clientTop & vbCrLf & _
               "left" & vbTab & .clientLeft & vbCrLf
    End With
End Sub
clientInformation Property

The `clientInformation` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
clientLeft Property

Returns a Long that represents the position in pixels of the left edge of a specified object without any margin, border, scroll bar, or padding settings that have been applied.

`expression.clientLeft`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Example

The following example displays a message containing the height, width, top, and left measurements of the body of the active document.

```
Sub GetClientInfo()
    Dim objBody As FPHTMLBody

    Set objBody = ActiveDocument.body

    With objBody
        MsgBox "height" & vbTab & .clientHeight & vbCrLf & _
             "width" & vbTab & .clientWidth & vbCrLf & _
             "top" & vbTab & .clientTop & vbCrLf & _
             "left" & vbTab & .clientLeft & vbCrLf
    End With
End Sub
```
clientTop Property

Returns a Long that represents the distance in pixels between the offsetTop property and the true top of the client area.

expression.clientTop

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example displays a message containing the height, width, top, and left measurements of the body of the active document.

Sub GetClientInfo()
    Dim objBody As FPHTMLBody

    Set objBody = ActiveDocument.body

    With objBody
        MsgBox "height" & vbTab & .clientHeight & vbCrLf & _
            "width" & vbTab & .clientWidth & vbCrLf & _
            "top" & vbTab & .clientTop & vbCrLf & _
            "left" & vbTab & .clientLeft & vbCrLf
    End With
End Sub
**clientWidth Property**

Returns a **Long** that represents the width in pixels of a specified object without any margin, border, scroll bar, or padding settings that have been applied.

*expression*.clientWidth

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example displays a message containing the height, width, top, and left measurements of the body of the active document.

Sub GetClientInfo()
    Dim objBody As FPHTMLBody
    
    Set objBody = ActiveDocument.body
    
    With objBody
        MsgBox "height" & vbTab & .clientHeight & vbCrLf & _
            "width" & vbCrLf & .clientWidth & vbCrLf & _
            "top" & vbCrLf & .clientTop & vbCrLf & _
            "left" & vbCrLf & .clientLeft & vbCrLf
    
    End With
End Sub
**clientX Property**

Returns an **Integer** that represents the horizontal position of the mouse pointer relative to the client area of the window, excluding window decorations and scroll bars.

`expression.clientX`

*expression*  Required. An expression that returns one of the objects in the `Applies To` list.
Remarks

Since the `clientX` property returns a value relative to the client, you can add the value of the `scrollLeft` property to determine the distance from the left edge of the BODY element.
Example

The following example displays a message that contains the horizontal and vertical position of the mouse pointer every time a user resizes the Microsoft FrontPage application window. This example must be placed in a class module, and it assumes you have declared an **FPHTMLWindow2** object variable called "window."

Private Sub window_onresize()
    Dim wEvent As IHTMLEventObj

    Set wEvent = window.event

    MsgBox "Your mouse is located at:" & vbCrLf & _
    "Left: " & wEvent.clientX & vbCrLf & _
    "Top: " & wEvent.clientY & vbCrLf & _
    "Are you having fun?", vbYesNo

End Sub
**clientY Property**

Returns an **Integer** that represents the vertical position of the mouse pointer relative to the client area of the window, excluding window decorations and scroll bars.

`expression.clientY`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

Since the `clientY` property returns a value relative to the client, you can add the value of the `scrollTop` property to determine the distance from the top edge of the BODY element.
Example

The following example displays a message that contains the horizontal and vertical position of the mouse pointer every time a user resizes the Microsoft FrontPage application window. This example must be placed in a class module, and it assumes you have declared an `FPHTMLWindow2` object variable called "window."

```vbscript
Private Sub window_onresize()
    Dim wEvent As IHTMLEventObj
    Set wEvent = window.event

    MsgBox "Your mouse is located at:" & vbCrLf & _
    "Left: " & wEvent.clientX & vbCrLf & _
    "Top: " & wEvent.clientY & vbCrLf & _
    "Are you having fun?", vbYesNo

End Sub
```
clip Property

Returns or sets a String that represents which part of a positioned object is visible.

expression.clip

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** for the **clip** property can be one of the following values:

- **auto**
  - Exposes entire object. Default value.
- **rect(top right bottom left)**
  - Specifies top, right, bottom, and left **Integer** values, any of which may be replaced by auto, leaving that side unclipped.
Example

The following example inserts an image into the active document, and then sets the `clip` property of the `Style` attribute for the new image.

```vba
Sub ClipImage()
    Dim objImage As FPHTMLImg
    ActiveDocument.body.insertAdjacentHTML "afterbegin", _
        "<img src="chelan.jpg" style="position:absolute"">
    Set objImage = ActiveDocument.body.all.tags("img").Item(0)
    objImage.Style.clip = "rect(auto 100 50 auto)"
End Sub
```
clipBottom Property

The `clipBottom` property is not implemented in Microsoft FrontPage and is therefore unavailable for programmatic access.
clipLeft Property

The clipLeft property is not implemented in Microsoft FrontPage and is therefore unavailable for programmatic access.
clipRight Property

The `clipRight` property is not implemented in Microsoft FrontPage and is therefore unavailable for programmatic access.
The `clipTop` property is not implemented in Microsoft FrontPage and is therefore unavailable for programmatic access.
closed Property

Returns a **Boolean** that represents whether the referenced window is closed. **True** indicates that the window is closed.

`expression.closed`

`expression` Required. An expression that returns one of the objects in the Applies To list.
**Example**

The following example displays a message stating whether the parent window of the specified document is closed. Use the `CallIsWindowClosed` subroutine below to call this subroutine.

```vba
Sub IsWindowClosed(ByRef objDoc As FPHTMLDocument)
    Select Case objDoc.parentWindow.closed
        Case False
            MsgBox "The parent window of the specified document is opened."
        Case True
            MsgBox "The parent window of the specified document is closed."
    End Select
End Sub
```

Use the following example to call the `IsWindowClosed` subroutine above.

```vba
Sub CallIsWindowClosed()
    Call IsWindowClosed(objDoc:=ActiveDocument)
End Sub
```
**code Property**

Returns or sets a **String** that represents the URL of the file containing the compiled Java class.

*expression.code*

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example specifies the URL of the Java class file and code type, and assigns an `id` attribute for the first object in the active document. This example assumes you have at least one OBJECT element in the active document.

Sub SetJavaCodeURL()
    Dim objJavaCode As FPHTMLObjectElement
    Set objJavaCode = ActiveDocument.all.tags("object").Item(0)

    With objJavaCode
        .code = "javacode.class"
        .codeType = "ASCII"
        .Id = "Java Code File"
    End With
End Sub
codeBase Property

Returns or sets a String that represents the URL of the specified Web component.

expression.codeBase

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** for the **codeBase** property can be appended with #Version=a,b,c,d, which indicates the version of the component. Version values (a,b,c,d) are indicated below. If the client computer has a newer version installed, no download occurs.

a. High-order word of the major version of the component available at the specified URL.
b. Low-order word of the major version of the component available at the specified URL.
c. High-order word of the minor version of the component available at the specified URL.
d. Low-order word of the minor version of the component available at the specified URL.
Example

The following example inserts an OBJECT element into the active document and then modifies the values for the id, width, height, and codeBase properties.

Sub SetObject()
    Dim objObject As FPHTMLObjectElement

    ActiveDocument.body.innerHTML = "<object id=""newobject"">"</object>

    Set objObject = ActiveDocument.all.tags("object") _
        .Item("newobject")

    With objObject
        .Id = "CommonDialog1"
        .Width = "32"
        .Height = "32"
        .codeBase = "http://activex.microsoft.com/controls/" & _
            "vb5/comdlg32.cab#Version=1,0,0,0"
    End With
End Sub
codeType Property

Returns or sets a String that represents the Internet media type, or MIME type, for the code associated with the object.

$expression.codeType$

$expression$ Required. An expression that returns one of the objects in the Applies To list.
Example

The following example specifies the URL of the Java class file and code type, and assigns an **id** attribute for the first OBJECT element in the active document. This example assumes that you have at least one OBJECT element in the active document.

```vba
Sub SetJavaCodeURL()
    Dim objJavaCode As FPHTMLObjectElement

    Set objJavaCode = ActiveDocument.all.tags("object").Item(0)

    With objJavaCode
        .code = "javacode.class"
        .codeType = "ASCII"
        .Id = "Java Code File"
    End With
End Sub
```
**color Property**

Sets or returns a **String** specifying a color name or red-green-blue (RGB) value, which represents the color of a specified object.

*expression*.color

*expression* Required. An expression that returns one of the objects in the Applies To list.
Remarks

When text is involved, the color property indicates the color of the text for the specified object. Otherwise, the color property indicates the color of the object itself, for example, in the case of lines and borders.

Color values can be specified as color names or Web safe Red-Green-Blue (RGB) color values. For more information on available color names and RGB color values, see the HTML Color Table.
**Example**

The following example adds a horizontal line to the active document and then formats the color, size, width, and alignment of the line.

```vba
Sub InsertLineBefore(ByRef objDoc As FHTMLDocument, _
                     ByRef strColor As String, ByRef strSize As String, _
                     bByRef strWidth As String, ByRef strAlign As String)
    Dim objElement As IHTMLElement
    Dim intLines As Integer
    Dim strID As String
    Dim objLine As FHTMLHRElement

    intLines = objDoc.all.tags("hr").Length
    strID = "Line" & intLines

    Set objElement = objDoc.activeElement
    objElement.insertAdjacentHTML where:="beforebegin", _
        HTML:="<HR id=""" & strID & """>"

    Set objLine = objDoc.body.all.tags("hr").Item(CVar(strID))

    With objLine
        .Color = strColor
        .Size = strSize
        .Width = strWidth
        .Align = strAlign
    End With
End Sub
```

Use the following example to call the preceding subroutine.

```vba
Sub CallInsertLineBefore()
    Call InsertLineBefore(objDoc:=ActiveDocument, strColor:="red", _
                          strSize:="15", strWidth:="75%", strAlign:="right")
End Sub
```
colorDepth Property

The `colorDepth` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
cols Property

cols property as it applies to the FPHTMLTable, FPHTMLTextAreaElement, IHTMLTable, and IHTMLTextAreaElement objects.

Returns or sets a Long that represents the number of columns in a table or the width of a TEXTAREA element, which corresponds to the value of the cols attribute.

expression.cols

expression  Required. An expression that returns one of the above objects.
Remarks

When applied to an FPHTMLTable or IHTMLTable object, the cols property specifies the number of columns in the table. When applied to an FPHTMLTextAreaElement or IHTMLTextAreaElement object, the cols property specifies the width of the TEXTAREA element.

cols property as it applies to the FPHTMLFrameSetSite and IHTMLFrameSetElement objects.

Sets or returns a String that represents the number and width of vertical frames in a FRAMESET element, which corresponds to the value of the cols attribute.

expression.cols

expression  Required. An expression that returns one of the above objects.
Remarks

The **String** value for the **cols** property can be one or more of the following comma-delimited values:

- **width**
  - Integer that specifies the frame width, in pixels.
- **percentage**
  - Integer, followed by a %. The value is a percentage of total available width.
- *****
  - Integer that specifies the frame width as a relative value. After allocating pixel or percentage values, the remaining space is divided among all relative-sized frames.

The number of comma-separated items is equal to the number of vertical frames contained within the FRAMESET, while the value of each item determines the frame width.
Example

As it applies to the **FPHTMLTable** object.

The following example sets the number of columns, background color, height, and width of the specified table.

```vbscript
Sub SetTableColumns(objTable As FPHTMLTable, strCols As String, _
                   strColor As String, strHeight As String, strWidth As String)
    With objTable
        .cols = strCols
        .bgColor = strColor
        .Height = strHeight
        .Width = strWidth
    End With
End Sub
```

Use the following example to call the preceding subroutine.

```vbscript
Sub CallSetTableColumns()
    Dim objTable As FPHTMLTable

    Set objTable = ActiveDocument.all.tags("table").Item(0)

    Call SetTableColumns(objTable, "3", "#883399", "200", "75%")
End Sub
```

As it applies to the **FPHTMLFrameSetSite** object.

The following example replaces the active document's current HTML with a frameset, and then specifies the number of rows or columns contained in each frameset.

```vbscript
Sub CreateFrameSet()
    Dim objFrames As FPHTMLFrameSetSite

    ActiveDocument.body.innerHTML = "<frameset id=""topframe"">" & vbTab & 
    "<frame id=""top"">" & vbCrLf & _
    vbTab & "<frameset id=""mainframe"">" & vbCrLf & _
    vbTab & vbTab & "<frame id=""left"">" & vbCrLf & _
    vbTab & vbTab & "<frame id=""right"">" & vbCrLf & _
    vbTab & vbTab & 
    "</frameset>" & vbCrLf & _
    "</frameset>"
```
Set objFrames = ActiveDocument.body.all.tags("frameset").Item("t"
objFrames.rows = "75,*"

Set objFrames = ActiveDocument.body.all.tags("frameset").Item("m"
objFrames.cols = "145,*"

End Sub
colSpan Property

Returns or sets a Long that represents the number of columns a table cell will span.

expression.colSpan

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the first cell in the first row of the first table to span two columns. This example assumes that you have at least one TABLE element in the active document.

Sub TableCellsSpanColumns()
    Dim objCell As FPHTMLTableCell

    Set objCell = ActiveDocument.body.all.tags("table")
                .Item(0).rows.Item(0).cells.Item(0)

    objCell.colSpan = 2
End Sub
compact Property

Returns or sets a **Boolean** representing whether to remove extra space between items in a specified list.

*expression*.compact

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example inserts an ordered list into the active document and then formats the list to remove spacing between list items and to set the numbering format to capital letters.

Sub AddSetList()
    Dim objList As FPHTMLLOListElement

    ActiveDocument.body.insertAdjacentHTML "beforeend", "<ol id="newlist"">" & vbCrLf & _
        "<li>item 1</li>" & vbCrLf & _
        "<li>item 2</li>" & vbCrLf & _
        "<li>item 3</li>" & vbCrLf & _
    "</ol>"

    Set objList = ActiveDocument.body.all.tags("ol").Item("newlist")

    With objList
        .compact = True
        .Type = "A"
    End With
End Sub
complete Property

Returns a **Boolean** that represents whether a specified object is fully loaded.

`expression.complete`

*expression* Required. An expression that returns one of the objects in the Applies To list.
Example

The following example inserts an image into the active document and then displays a message specifying whether the image file is fully loaded. This example assumes that you have a folder called "images" with an image file called "sun.gif."

Sub IsImageLoaded()
    Dim objImage As FPHTMLImg
    ActiveDocument.body.insertAdjacentHTML "beforeend", _
        "<img id=""newimage"" src=""images/sun.gif"">"
    Set objImage = ActiveDocument.all.tags("img").Item("newimage")
    If objImage.complete = False Then
        MsgBox "Please wait until the image is fully loaded."
    Else
        MsgBox "Thank you for waiting. The image is now loaded."
    End If
End Sub
containerWidth Property

Sets or returns a **String** that represents the expected width of a zone within a Web Part. Can be any valid CSS width value.

*expression.containerWidth*

*expression*  Required. An expression that returns one of the objects in the Applies To list.
content Property

Returns or sets a String that represents the value of the content attribute that is associated with the http-equiv or name attribute of a META element.

expression.content

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **content** property can be one or more of the following values:

- **description**: Meta-information. Integer consisting of the number of seconds to elapse before the document is refreshed. This value requires the **HTTP-EQUIV** attribute to be set with the *refresh* value.
- **url**: Location that is loaded when the document is refreshed. This value requires the **HTTP-EQUIV** attribute to be set with the *refresh* value.
- **mimetype**: MIME type used for the *charset* value.
- **charset**: Character set of the document. This value requires the *mimetype* value to be set with a valid MIME type, and the **HTTP-EQUIV** attribute to be set with the *Content-Type* value.
**Example**

The following example inserts a `<META>` tag that contains the character set to use for the active document.

```vbscript
Sub InsertCharset(ByVal objDoc As FPHTMLDocument, ByVal strID As String,
                   ByVal strHTTP As String, ByVal strContent As String, ByVal strCharset As String)
    Dim objMeta As FPHTMLMetaElement
    objDoc.all.tags("head").Item(0).innerHTML = "<META id="""" & strID & """>"
    Set objMeta = ActiveDocument.all.tags("meta").Item(CVar(strID))
    With objMeta
        .httpEquiv = strHTTP
        .content = strContent
        .Charset = strCharset
    End With
End Sub
```

Use the following example to call the preceding subroutine.

```vbscript
Sub CallInsertCharset()
    Call InsertCharset(objDoc:=ActiveDocument, strID:="iso_content",
                       strHTTP:="Content-Type", strContent:="text/html", strCharset
End Sub
```
cookie Property

The **cookie** property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
cookieEnabled Property

The `cookieEnabled` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
coords Property

Returns or sets a **String** that represents the coordinates of an AREA element within a MAP element.

*expression.coords*

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The format of the **String** value of the **coords** property depends on the value of the **Shape** property (corresponding to the **shape** attribute) as follows:

<table>
<thead>
<tr>
<th>shape property</th>
<th>coords property</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHAPE = &quot;circ&quot; or &quot;circle&quot;</td>
<td>COORDS = &quot;x1,y1,r&quot; – Where x1,y2 are the coordinates of the center of the circle, and r is the radius of the circle.</td>
</tr>
<tr>
<td>SHAPE = &quot;poly&quot; or &quot;polygon&quot;</td>
<td>COORDS = &quot;x1,y1,x2,y2...xn,yn&quot; – Where each x,y pair contains the coordinates of one vertex of the polygon.</td>
</tr>
<tr>
<td>SHAPE = &quot;rect&quot; or &quot;rectangle&quot;</td>
<td>COORDS = &quot;x1,y1,x2,y2&quot; – Where x1,y1 are the coordinates of the upper-left corner of the rectangle and x2,y2 are the coordinates of the lower-right coordinates of the rectangle.</td>
</tr>
</tbody>
</table>
Example

The following example replaces the text in the active document with a graphic, and then inserts an image map with one AREA element and specifies its share, coordinates, and hyperlink URL.

Sub SetImageCoords()
    Dim objImage As FPHTMLImg
    Dim objArea As FPHTMLAreaElement
    
    ActiveDocument.body.innerHTML = _
    "<img src="graphics/chelan.jpg" id="chelan">" & vbCrLf
    
    Set objImage = ActiveDocument.all.tags("img").Item("chelan")
    objImage.useMap = "#ImageMap"
    
    ActiveDocument.body.insertAdjacentHTML where:="beforeend", _
    HTML:="<map name="ImageMap">" & vbCrLf & _
    "<area id="Area1">" & vbCrLf & "</map>" & vbCrLf
    
    Set objArea = ActiveDocument.body.all.tags("area").Item("Area1")
    
    With objArea
        .Shape = "rect"
        .coords = "5, 16, 151, 286"
        .href = "http://www.microsoft.com"
    End With
End Sub
cpuClass Property

The cpuClass property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
cssText Property

Returns or sets a String that represents the value of an inline style sheet for a specified element.

(expression).cssText

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The value of the `cssText` property is the cascading style sheet property name followed by a colon and the value of the property. For example, "background-color:yellow" sets the background color of an element to yellow.

For a list of possible properties and values see the [CSS Reference](https://msdn.microsoft.com) on the Microsoft Developer Network (MSDN) Web site.
Example

The following example adds an inline style that changes the background color of the first paragraph in the active document to yellow.

Dim objPara As FPHTMLParaElement

Set objPara = ActiveDocument.all.tags("p").Item(0)
objPara.Style.cssText = "background-color:yellow"
**ctrlKey Property**

Returns a **Boolean** that represents whether the CTRL key on the keyboard is pressed. **True** if the CTRL key is pressed. **False** if it is not.

`expression.ctrlKey`

**expression**  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example displays a message every time the user clicks on the active document telling whether the user is pressing the CTRL key. This example must be placed in a class module, and it assumes that you have declared in the General Declarations section of a class module an FPHTMLDocument object variable called "doc" using the WithEvents keyword.

Private Function doc_onclick() As Boolean
    Dim wEvent As IHTMLEventObj

    Set wEvent = window.event

    Select Case wEvent.ctrlKey
        Case False
            MsgBox "You are not pressing your ctrl key."
        Case True
            MsgBox "You are pressing your ctrl key."
    End Select
End Function
currentStyleState Property

Returns an IFPStyleState object that represents the current style state of the specified document.

expression.currentStyleState

expression  Required. An expression that returns one of the objects in the Applies To list.
**Example**

The following example returns the style state of the current document, and then displays a message containing the display type of the current style state.

```vbscript
Sub StyleState()
    Dim objStyleState As IFPStyleState
    Set objStyleState = ActiveDocument.currentStyleState
    MsgBox "The display type of the current style state is """" & _
            objStyleState.display & ".""
End Sub
```
cursor Property

Sets or returns a **String** that represents how to display the mouse pointer as it moves over a specified object when a page is displayed in the browser window.

`expression.cursor`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
# Remarks

The **String** for the **cursor** property can be one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all-scroll</td>
<td>Internet Explorer 6 and later. Arrows pointing up, down, left, and right with a dot in the middle, indicating that the page can be scrolled in any direction.</td>
</tr>
<tr>
<td>auto</td>
<td>Default. Browser determines which cursor to display based on the current context.</td>
</tr>
<tr>
<td>col-resize</td>
<td>Internet Explorer 6 and later. Arrows pointing left and right with a vertical bar separating them, indicating that the item/column can be resized horizontally.</td>
</tr>
<tr>
<td>crosshair</td>
<td>Simple cross hair.</td>
</tr>
<tr>
<td>default</td>
<td>Platform-dependent default cursor; usually an arrow.</td>
</tr>
<tr>
<td>hand</td>
<td>Hand with the first finger pointing up, as when the user moves the pointer over a link.</td>
</tr>
<tr>
<td>help</td>
<td>Arrow with question mark, indicating help is available.</td>
</tr>
<tr>
<td>move</td>
<td>Crossed arrows, indicating something is to be moved.</td>
</tr>
<tr>
<td>no-drop</td>
<td>Internet Explorer 6 and later. Hand with a small circle with a line through it, indicating that the dragged item cannot be dropped at the current cursor location.</td>
</tr>
<tr>
<td>not-allowed</td>
<td>Internet Explorer 6 and later. Circle with a line through it, indicating that the requested action will not be carried out.</td>
</tr>
<tr>
<td>pointer</td>
<td>Internet Explorer 6 and later. Hand with the first finger pointing up, as when the user moves the pointer over a link. Identical to hand.</td>
</tr>
<tr>
<td>progress</td>
<td>Internet Explorer 6 and later. Arrow with an hourglass next to it, indicating that a process is running in the background. User interaction with the page is unaffected.</td>
</tr>
<tr>
<td>row-resize</td>
<td>Internet Explorer 6 and later. Arrows pointing up and down with a horizontal bar separating them, indicating that the item/row can be resized vertically.</td>
</tr>
<tr>
<td>text</td>
<td>Editable text; usually an I-bar.</td>
</tr>
</tbody>
</table>

Internet Explorer 6 and later. Cursor is defined by the author, using a custom Uniform Resource Identifier (URI), such as
url(uri)  url('mycursor.cur'). Cursors of type .CUR and .ANI are the only supported cursor types.

vertical-text  Internet Explorer 6 and later. Editable vertical text, indicated by a horizontal I-bar.

wait  Hourglass or watch, indicating that the program is busy and the user should wait.

*-resize  Arrows, indicating an edge is to be moved; the asterisk (*) can be N, NE, NW, S, SE, SW, E, or W—each representing a compass direction.
Example

The following example sets the mouse pointer for all images in the specified object to display as a hand when the mouse is positioned over the image and the page is displayed in a browser window.

Sub ChangeMousePointerForImages(objDoc As FHTMLDocument)
    Dim objImage As FHTMLImg
    For Each objImage In objDoc.images
        objImage.Style.cursor = "hand"
    Next
End Sub

Use the following example to call the preceding subroutine.

Sub CallChangeMousePointerForImages()
    Call ChangeMousePointerForImages(ActiveDocument)
End Sub
**dataPageSize Property**

Returns or sets a **Long** that represents the number of records displayed in a table bound to a data source. Corresponds to the **datapagesize** attribute for a TABLE element.

`expression.dataPageSize`

*expression* Required. An expression that returns one of the objects in the Applies To list.
The `defaultSelected` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
defaultStatus Property

The `defaultStatus` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
defer Property

Sets or retrieves a **String** that represents the status of an **FPHTMLScriptElement** or **IHTMLScriptElement** object. Corresponds to the **defer** attribute for the SCRIPT element.

`expression.defer`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The value of the `defer` property can be one of the following string values:

- `true` Script execution is deferred.
- `false` Script execution is not deferred.

Use the `defer` property to set the defer attribute for a SCRIPT element that contains only function declarations. Setting the `defer` attribute to "true" tells the browser that it doesn't need to read the entire script before loading and displaying the page. Use the `defer` attribute only when the SCRIPT element contains only functions and when subsequent scripts that are immediately executed do not call any of these functions as immediately executed code defined within a deferred script can react unpredictably.
**Example**

The following example defers execution of the first script in the active document when the document is displayed in a browser.

```vbnet
Dim objScript As FPHTMLScriptElement
Set objScript = ActiveDocument.Scripts.Item(0)
objScript.defer = "true"
```
The `dialogArguments` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**dialogHeight Property**

The `dialogHeight` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
The `dialogLeft` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
The `dialogTop` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**dialogWidth Property**

The `dialogWidth` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
dir Property

Returns or sets a String that represents the reading order of text in an HTML element.

expression.dir

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The `dir` property can be one of the following `String` values:

- `ltr` Text flow is left to right.
- `rtl` Text flow is right to left.
Example

The following example specifies right-to-left reading order for the first form in the active document.

Dim objForm As FPHTMLFormElement
Set objForm = ActiveDocument.forms(0)
objForm.Dir = "rtl"
direction Property

Sets or returns a String that represents the direction in which text scrolls in a marquee.

`expression.direction`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** for the **direction** property can be one of the following values:

- **left**  Marquee scrolls left. Default value.
- **right** Marquee scrolls right.
- **down**  Marquee scrolls down.
- **up**    Marquee scrolls up.
Example

The following example inserts a MARQUEE element into the active document, and then sets the behavior, direction, number of times to loop, height, width, vertical alignment, and border of the new element.

Sub InsertMarquee(strBehavior As String, strDirection As String, strLoop As String, strHeight As String, strWidth As String, strVAlign As String, strBorder As String, strText As String)
Dim objMarquee As FPHTMLMarqueeElement

ActiveDocument.body.insertAdjacentHTML where:="afterbegin", _
HTML:="<marquee id="newmarquee"></marquee>"

Set objMarquee = ActiveDocument.all.tags("marquee").Item("newmarq

With objMarquee
  .behavior = strBehavior
  .direction = strDirection
  .loop = strLoop
  .Height = strHeight
  .Width = strWidth
  With .Style
    .verticalAlign = strVAlign
    .Border = strBorder
  End With
  .innerText = strText
End With
End Sub

Use the following example to call the preceding subroutine.

Sub CallInsertMarquee()
  Call InsertMarquee("slide", "up", "5", _
                "100%", "10%", "middle", "dashed thick red", _
                "This is a scrolling marquee.")
End Sub
disabled Property

Sets or returns a Boolean that represents the status of a specified object. True indicates that the object is enabled.

expression.disabled

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example inserts a check box into the active document, and then disables the check box.

Sub DisableCheckBox()
    Dim objCheckBox As FPHTMLInputButtonElement
    ActiveDocument.body.insertAdjacentHTML where:="beforeend", _
    HTML:="<input type=""checkbox"" id=""newcheckbox"">" & vbCrLf
    Set objCheckBox = ActiveDocument.body.all.tags("input") _
        .Item("newcheckbox")
    objCheckBox.disabled = True
End Sub
**display Property**

Sets or returns a **String** that represents whether or not the browser displays the object.

`expression.display`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value for the **display** property can be one of the following:

- **block**: The browser displays the object as a block element.
- **none**: The browser doesn't display the object.
- **inline**: The browser displays the object as an inline element sized by the dimensions of the content.
- **list-item**: The browser displays the object as a block element and includes a list-item marker.
- **table-header-group**: The browser displays the table header before all other rows and row groups, after any captions, and spans the top of each page with the table header.
- **table-footer-group**: The browser displays the table footer before all other rows and row groups, before any captions, and spans the top of each page with the table footer.
Example

The following example inserts an image, and then specifies that the image will not display in the browser. This example assumes that you have an image file called "chelan.jpg" in a folder called "images." When you run this example, replace the image file name with an actual image file name.

```vba
Sub SetDisplayProperty()
    Dim objImage As FPHTMLImg

    ActiveDocument.activeElement.insertAdjacentHTML where:="afterbegin",
    HTML:="<img id=""newimage"" src=""chelan.jpg"">"

    Set objImage = ActiveDocument.body.all.tags("img").Item("newimag" objImage.Style.display = "none"
End Sub
```
Document Property (Page Object Model)

Returns an **Object** that represents the Web page displayed in the Microsoft FrontPage application window.

*expression*.Document

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

Use the Document property to retrieve information about a document, to examine and modify the HTML elements and text within the document, and to process events.
Example

The following example specifies the title of the page and the contents of the first heading to the specified string. To call this function, use the CallSetTitleAndFirstHeading subroutine that follows.

Function SetTitleAndFirstHeading(ByRef objBody As FPHTMLBody, ByVal strTitle As String) As Boolean
    Dim objHeading As IHTMLElement
    setTitleAndFirstHeading = False
    Set objBody = ActiveDocument.body
    If InStr(1, UCase(objBody.innerHTML), UCase("h1")) < 1 Then
        objBody.insertAdjacentHTML "afterBegin", "<h1>" & strTitle & "</h1>
    Else
        Set objHeading = objBody.Children.tags("h1").Item(0)
        objHeading.innerText = strTitle
    End If
    objBody.Document.Title = strTitle
    setTitleAndFirstHeading = True
End Function

The following example calls the preceding SetTitleAndFirstHeading function and displays a message indicating whether the function was successful.

Sub CallSetTitleAndFirstHeading()
    Dim blnResponse as Boolean

    blnResponse = SetTitleAndFirstHeading(ActiveDocument.body, _
        "FrontPage Developer's Home Page")

    If blnResponse = True Then
        MsgBox "You have successfully changed the title " & _
            "and first heading of the current page."
    Else
        MsgBox "Title and first heading were not changed."
    End If
End Sub
DocumentHTML Property

Sets or returns a **String** that represents the HTML tags and content of the specified **Document** object.

```
expression.DocumentHTML
```

**expression**  Required. An expression that returns an **FPHTML.Document** object.
Example

The following example uses the Microsoft Visual Basic **Replace** function to place a blue border around all paragraphs in the active document.

```vba
Sub ReplaceHTMLText()
    ActiveDocument.**DocumentHTML** = Replace _
    (ActiveDocument.**DocumentHTML**, "<p>", _
    "<p style="border: thick dashed blue">")
End Sub
```
domain Property

Returns a String that represents the name of the server on which the specified document is located or the Web address with which it is associated.

expression.domain

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

While the Object Browser in the Visual Basic Editor shows the **domain** property as Read/Write, if you try to set the property, you will receive an error.

If the **domain** property returns an empty **String**, this generally means the page is part of a disk-based Web site or hasn't been published to a server.
Example

The following example displays a message informing the user that the page hasn't been published if the domain name for the current document is empty.

If ActiveDocument.domain = "" Then
    MsgBox "The document " & ActiveDocument.Url & " has not been published to a server."
End If
**dynsrc Property**

Sets or returns a **String** that represents the address of a video clip or VRML world to be displayed in an IMG element.

`expression.dynsrc`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
**Example**

The following specifies the video clip to display for the first image in the active document. This example assumes there is at least one image in the active document.

Dim objImage As FPHTMLImg

Set objImage = ActiveDocument.images.Item(0)
objImage.*dynsrc* = "video/video.mov"
elements Property

Returns an IHTMLCollection object that represents a collection of elements contained in a form.

expression.elements

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example inserts an **id** attribute for each of the elements contained in the first form in the active document.

```vbscript
Sub SetIDForFormElements()
    Dim objForm As FPHTMLFormElement
    Dim objElement As IHTMLElement
    Dim intElement As Integer

    Set objForm = ActiveDocument.forms.Item(0)

    For Each objElement In objForm.elements
        intElement = intElement + 1
        objElement.Id = "FormElement" & intElement
    Next objElement
End Sub
```
embeds Property

Returns an IHTMLElementCollection object that represents the EMBED elements contained in the specified page.

expression.embeds

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The embeds property returns the same collection as the plugins property.
Example

The following example sets an object reference to the first item in the collection of embedded objects in the active document.

Dim objEmbed As FPHTMLEmbed
Set objEmbed = ActiveDocument.embeds.Item(0)
encoding Property

Returns or sets a String that represents the MIME encoding for a FORM element. Corresponds to the enctype attribute.

expression.encoding

type

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the enctype attribute of the first form in the active document to the default MIME encoding type for a form.

Dim objForm As FPHTMLFormElement

Set objForm = ActiveDocument.forms.Item(0)
objForm.encoding = "application/x-www-form-urlencoded"
event Property

As it applies to the **FPHTMLScriptElement** and **IHTMLScriptElement** objects.

Returns a **String** that represents the name of the event handler for which a script is written.

```
expression.event
```

**expression**  Required. An expression that returns one of the above objects.

As it applies to the **FPHTMLWindow2** and **IHTMLWindow2** objects.

Returns an **IHTMLEventObj** object that represents a user action that occurs on a document, window, or element within a document.

```
expression.event
```

**expression**  Required. An expression that returns one of the above objects.
Example

As it applies to the `FPHTMLScriptElement` object.

The following example sets the event attribute to the onmousemove event for the first SCRIPT element in the active document.

```vbscript
Dim objScript As FPHTMLScriptElement
Set objScript = ActiveDocument.Scripts.Item(0)
objScript.event = "onmousemove"
```

As it applies to the `FPHTMLWindow2` object.

The following example displays a message every time the user clicks on the active document telling whether the user is pressing the SHIFT key. This example must be placed in a class module, and it assumes that you have declared in the General Declarations section of a class module an `FPHTMLDocument` object variable called "doc" using the ` WithEvents` keyword.

```vbscript
Private Function objDoc_onclick() As Boolean
    Dim objEvent As IHTMLEventObj
    Set objEvent = Window.event

    Select Case objEvent.shiftKey
        Case False
            MsgBox "You are not pressing your SHIFT key."
        Case True
            MsgBox "You are pressing your SHIFT key."
    End Select
End Function
```
**external Property**

Returns a PageWindowEx object.

`expression.external`
Example

This example returns the active page window, and if the page has been changed, it saves the page.

Sub SaveChangedPage()
    Dim objDoc As IFPDocument
    Dim objWindow As PageWindowEx

    On Error GoTo UnableToSavePage

    Set objDoc = ActiveDocument
    Set objWindow = ActivePageWindow._ActiveFrameWindow.Window.external

    If objWindow.IsDirty Then objWindow.Save

ExitSub:
    Exit Sub

UnableToSavePage:
    MsgBox "Unable to save the page. " & _
    "If you haven't saved the page previously, " & _
    vbCrLf & "you need to save it first before " & _
    "you can use the Save method."

    GoTo ExitSub
End Sub
**face Property**

Sets or returns a **String** that represents the font for the specified object.

*expression.face*

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example inserts a BASEFONT element into the specified document if one doesn't already exist and formats the font name and, optionally, the font size.

```vbnet
Function SetBasefont(objDoc As FPHTMLDocument, strFontFace As String Optional strFontSize As String) As FPHTMLBaseFontElement
    Dim objBody As FPHTMLBody
    Dim objTemp As FPHTMLBaseFontElement

    Set objBody = objDoc.body
    If objBody.all.tags("basefont").Length <= 0 Then
        objBody.insertAdjacentHTML "afterbegin", "<Basefont>
        Set objTemp = objBody.all.tags("basefont").Item(0)
    Else
        Set objTemp = objBody.all.tags("basefont").Item(0)
    End If

    With objTemp
        .face = strFontFace
        If Len(strFontSize) > 0 Then .Size = strFontSize
    End With

    Set SetBasefont = objTemp
End Function
```

Use the following example to call the preceding subroutine.

```vbnet
Sub CallSetBasefont()
    Call SetBasefont(ActiveDocument, "comic sans ms", "5")
End Sub
```
**fgColor Property**

Sets or returns a **String**, specifying a color name or red-green-blue (RGB) value, that represents the foreground (or text) color of the document.

`expression.fgColor`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The `fgColor` property sets the value of the `text` attribute for the BODY element.

For more information about setting colors, see the HTML Color Table.
Example

The following example changes the font color of the specified document to the color specified in the \textit{strColor} argument.

\begin{verbatim}
Sub ChangeTextColor(objDoc As FPHTMLDocument, strColor As String)
    objDoc.\textit{fgColor} = strColor
End Sub
\end{verbatim}

Use the following example to call the preceding subroutine.

\begin{verbatim}
Sub CallChangeTextColor()
    Call ChangeTextColor(ActiveDocument, "#663399")
End Sub
\end{verbatim}
fileCreatedDate Property

Retrieves a String that represents the date a file was created.

expression.fileCreatedDate

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example inserts a new paragraph into the active document and then inserts the dates the active document was created and last modified.

Sub CheckDate()
    Dim objPara As FPHTMLParaElement

    With ActiveDocument
        .body.insertAdjacentHTML where:="beforeend", _
            HTML:="<p id="newpara1"></p>"

        Set objPara = .body.all.tags("p").Item("newpara1")

        objPara.innerHTML = "Created: " & .fileCreatedDate & _
                          "<BR>Changed: " & .fileModifiedDate
    End With
End Sub
fileModifiedDate Property

Retrieves a String that represents the date a file was last modified.

`expression.fileModifiedDate`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Example

The following example inserts a new paragraph into the active document and then inserts the dates the active document was created and last modified.

Sub CheckDate()
    Dim objPara As FPHTMLParaElement

    With ActiveDocument
        .body.insertAdjacentHTML where:="beforeend", _
            HTML:="<p id="newpara1"></p>"

        Set objPara = .body.all.tags("p").Item("newpara1")

        objPara.innerHTML = "Created: " & .fileCreatedDate & _
            "<BR>Changed: " & .fileModifiedDate
    End With
End Sub
**fileSize Property**

Returns a **String** that represents the file size in bytes.

`expression.fileSize`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example inserts a new paragraph in the active document and then inserts the dates for when the active document was created and last modified and the size of the file.

Sub AddDates()
    Dim objPara As FPHTMLParaElement

    With ActiveDocument
        .body.insertAdjacentHTML where:="beforeend", _
            HTML:="<p id="newpara2"></p>"

        Set objPara = .body.all.tags("p").Item("newpara2")

        objPara.innerHTML = "Created: " & .fileCreatedDate & _
        "<BR>Changed: " & .fileModifiedDate & _
        "<BR>File Size: " & .fileSize & " bytes"

    End With
End Sub
fileUpdatedDate Property

Retrieves a **String** that represents the date a document or image file was last updated.

`expression.fileUpdatedDate`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Example

The following example inserts an image into the active document and then displays a message containing the date the image file was last updated. This example assumes you have an image file named parkbench.jpg. When you run this subroutine, change "parkbench.jpg" to an actual image file in your Web site.

Sub InsertImage()
    Dim objImage As FPHTMLImg

    ActiveDocument.body.insertAdjacentHTML where:="beforeend", _
    HTML:="<img src="parkbench.jpg" id="park bench">"

    Set objImage = ActiveDocument.body.all.tags("img").Item("park be

    MsgBox objImage.fileUpdatedDate
End Sub
filter Property

The filter property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
filters Property

The `filters` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
Find Property

Returns or sets a String that represents the text or element for which to search.

expression.Find

expression Required. An expression that returns a SearchInfo object.
Remarks

If the *Action* property is set to `fpSearchFindText` or `fpSearchReplace` text, the *Find* property defines the text for which to search; if the *Action* property is set to `fpSearchFindTag`, then the *Find* property defines the name of the HTML tag for which to search.
Example

The following example selects the next occurrence of the P element if one is found in the active document.

Dim objSearch As SearchInfo
Dim blnFound As Boolean
Dim objRange As IHTMLTxtRange

Set objSearch = Application.CreateSearchInfo
objSearch.Find = "p"
objSearch.Action = fpSearchFindTag

Set objRange = Application.ActiveDocument.selection.createRange
blnFound = Application.ActiveDocument.Find(objSearch, Nothing, objRange)
If blnFound = True Then objRange.Select
**floatVal Property**

Returns or sets a `String` that represents the side of the object on which the text will flow. Corresponds to the `float` attribute of a cascading style sheet.

`expression.floatVal`

`expression` Required. An expression that returns an `IFPStyleState` object.
Remarks

The **String** value **floatVal** property can be one of the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>Object displays where it appears in the text. Default.</td>
</tr>
<tr>
<td>left</td>
<td>Text flows to the right of the object.</td>
</tr>
<tr>
<td>right</td>
<td>Text flows to the left of the object.</td>
</tr>
<tr>
<td>inherit</td>
<td>Object inherits setting from the parent element.</td>
</tr>
</tbody>
</table>

Use the **floatVal** property to set the **float** attribute for all textual elements, such as form elements and text ranges. Use the **styleFloat** property for all other elements.
**Example**

The following example inserts a DIV element around the current selection in the document and specifies that text will flow along the left side.

```vba
Dim objStyle As IFPStyleState
Dim objRange As IHTMLTxtRange

Set objStyle = ActiveDocument.createStyleState
Set objRange = ActiveDocument.Selection.createRange
objStyle.gather objRange

objStyle.floatVal = "left"
objStyle.Apply
```
font Property

Sets or returns a **String** that represents up to six separate font properties for the specified object.

*expression*.

*expression*.

Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value for the **font** property can be up to six of the space-delimited values listed in the following table, plus one or more of the additional values shown below them:

- **font-style** Any of the range of font-style values available to the **fontStyle** property.
- **font-variant** Any of the range of font-variant values available to the **fontVariant** property.
- **font-weight** Any of the range of font-weight values available to the **fontWeight** property.
- **font-size** Any of the range of font-size values available to the **fontSize** property.
- **line-height** Any of the range of line-height values available to the **lineHeight** property.
- **font-family** Any of the range of font-family values available to the **fontFamily** property.

**Additional values**

The additional values specify font formatting according to the user's system settings. For example, setting the font property to "menu" will display the affected text using the font formatting the user has specified for menus. The following example displays all text in the body of the active document using the font formatting defined for icon labels on the user's desktop.

```javascript
ActiveDocument.body.Style.Font = "icon"
```

- **caption** Text font used in objects that have captions (buttons, labels, and so on).
- **icon** Text font used in icon labels.
- **menu** Text font used in menus.
- **messagebox** Text font used in dialog boxes.
- **smallcaption** Text font used in small controls.
- **statusbar** Text font used in window status bars.
Example

The following example sets the font characteristics for the body of the active document

ActiveDocument.body.Style.Font = "italic 12pt bold tahoma"
**fontFamily Property**

Sets or returns a **String** that represents the name of the font used to format the text within an HTML element.

`expression.fontFamily`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example modifies the **style** attribute of the BODY element in the active document with the specified background, border, font, and margin settings.

`ActiveDocument.body.Style.fontFamily = "Tahoma"`
**fontSize Property**

Sets or returns a **String** that represents the size of the font used for text in the object.

```
expression.fontSize
```

(expression) Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value for the **fontSize** property can be one of the following:

- **absolute-size**
  - Set of keywords that indicate predefined font sizes. Possible keywords include [ xx-small | x-small | small | medium | large | x-large | xx-large ]. Named font sizes scale according to the user's font setting preferences.

- **relative-size**
  - Set of keywords that are interpreted as relative to the font size of the parent object. Possible values include [ larger | smaller ].

- **length**
  - Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex).

- **percentage**
  - Integer, followed by a %. The value is a percentage of the parent object's font size. In Internet Explorer 3.0, the value is calculated as a percentage of the default font size.

The default value of the **fontSize** property is medium. The cascading style sheets (CSS) attribute is inherited.
Example

This example inserts a new paragraph containing the current user's name into the active document, and then formats the font characteristics.

Sub FontFaceSource()
    Dim objPara As FPHTMLParaElement

    ActiveDocument.body.insertAdjacentHTML where:="beforeend", _
        HTML:="<p id="username">" & Application.UserName & "</p>

    Set objPara = ActiveDocument.body.all.tags("p").Item("username")

    With objPara.Style
        .fontFamily = "Tahoma"
        .FONTSIZE = "40pt"
        .fontStyle = Italic
        .fontVariant = "small-caps"
        .fontWeight = "bold"
    End With
End Sub
fontSmoothingEnabled Property

The `fontSmoothingEnabled` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**fontStyle Property**

Sets or returns a **String** that represents the font-style setting for an inline **style** attribute of a specified object.

`expression.fontSize`

**expression**  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value for the `fontStyle` property can be one of the following:

- `normal`  Font is normal. Default.
- `italic`  Font is italic.
- `oblique`  Font is italic.
Example

This example inserts a new paragraph containing the current user's name into the active document, and then formats the font characteristics.

Sub FontFaceSource()
    Dim objPara As FPHTMLParaElement

    ActiveDocument.body.insertAdjacentHTML where:="beforeend", _
        HTML:="<p id="username">" & Application.UserName & "</p>"

    Set objPara = ActiveDocument.body.all.tags("p").Item("username")

    With objPara.Style
        .fontFamily = "Tahoma"
        .FONTSIZE = "40pt"
        .fontStyle = "italic"
        .fontVariant = "small-caps"
        .fontWeight = "bold"
    End With
End Sub
fontVariant Property

Sets or returns a **String** that represents whether the text of the object displays normally or in small capital letters.

`expression.fontVariant`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

**String** that specifies one of the following values:

- **normal**: Font is normal. Default.
- **small-caps**: Font is in small capital letters.
Example

This example inserts a new paragraph containing the current user's name into the active document, and then formats the font characteristics.

Sub FontFaceSource()
    Dim objPara As FPHTMLParaElement

        ActiveDocument.body.insertAdjacentHTML where:="beforeend", _

        HTML:="<p id="username">" & Application.UserName & "</p>"

    Set objPara = ActiveDocument.body.all.tags("p").Item("username")

    With objPara.Style
        .fontFamily = "Tahoma"
        .FONTSIZE = "40pt"
        .fontStyle = Italic
        .fontVariant = "small-caps"
        .fontWeight = "bold"
    End With

End Sub
fontWeight Property

Sets or returns a String that represents the weight of the font for the specified object.

`expression.fontWeight`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value for the **fontWeight** property can be one of the following:

- **normal**: Font is normal. Default.
- **bold**: Font is bold.
- **bolder**: Font is heavier than regular bold.
- **lighter**: Font is lighter than normal.
- **100**: Font is at least as light as the 200 weight.
- **200**: Font is at least as bold as the 100 weight and at least as light as the 300 weight.
- **300**: Font is at least as bold as the 200 weight and at least as light as the 400 weight.
- **400**: Font is normal.
- **500**: Font is at least as bold as the 400 weight and at least as light as the 600 weight.
- **600**: Font is at least as bold as the 500 weight and at least as light as the 700 weight.
- **700**: Font is bold.
- **800**: Font is at least as bold as the 700 weight and at least as light as the 900 weight.
- **900**: Font is at least as bold as the 800 weight.
**Example**

This example inserts a new paragraph containing the current user's name into the active document, and then formats the font characteristics.

```vbscript
Sub FontFaceSource()
    Dim objPara As FPHTMLParaElement

    ActiveDocument.body.insertAdjacentHTML where:="beforeend", _
    HTML:="<p id="username">" & Application.UserName & "</p>

    Set objPara = ActiveDocument.body.all.tags("p").Item("username")

    With objPara.Style
        .fontFamily = "Tahoma"
        .FONTSIZE = "40pt"
        .fontStyle = Italic
        .fontVariant = "small-caps"
        .fontWeight = "bold"
    End With
End Sub
```
form Property

Returns an IHTMLFormElement object that represents the form to which an element belongs.

expression.form

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example replaces the text within the active document with a form containing a text box and two buttons. It then sets the value of the `method` property of the parent FORM element to "post."

Sub CreateForm()
    Dim objTextbox As FPHTMLTextAreaElement

    ActiveDocument.body.innerHTML = "<form id="textform">" & vbCrLf & "<textarea rows="10" cols="40" name="textbox" id="textbox">" & vbCrLf & "</textarea><br>" & vbCrLf & "<input type="submit" value="Submit" name="submit" id="submit">" & vbCrLf & "<input type="reset" value="Reset" name="reset" id="reset">" & vbCrLf & "</form>"

    Set objTextbox = ActiveDocument.body.all.tags("textarea").Item("textbox")
    objTextbox.form.method = "post"
End Sub
forms Property

Returns an IHTMLElementCollection object that represents a collection of all FORM elements contained in a document.

expression.forms

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example replaces the text in the active document with a form, and then adds a new element to the form.

Sub CreateNewForm()
    Dim objForm As FPHTMLFormElement
    ActiveDocument.body.innerHTML = "<form id="textform">
            <textarea rows="10" cols="40" name="textbox" id="textbox">" & vbCrLf & _
            "</textarea><br>" & vbCrLf & _
            "<input type="submit" value="Submit" name="submit" id="submit">" & vbCrLf & _
            "<input type="reset" value="Reset" name="reset" id="reset">" & vbCrLf & _
            "</form>"
    Set objForm = ActiveDocument.forms.Item("textform")
    objForm.insertAdjacentHTML where:="beforeend", _
        HTML:="<input type="checkbox" name="checkbox" value="ON">Save formatted text"
End Sub
frame Property

Sets or returns a String that represents the way the border frame around a specified table is displayed. The frame property corresponds to the frame attribute of a TABLE element.

expression.frame

expression   Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value for the **frame** property can be one of the following:

- **void**  All outside table borders are removed. Default value.
- **above** Border on the top side of the border frame is displayed.
- **below** Border on the bottom side of the table frame is displayed.
- **border** Borders on all sides of the table frame are displayed.
- **box** Borders on all sides of the table frame are displayed.
- **hsides** Borders on the top and bottom sides of the table frame are displayed.
- **1hs** Border on the left side of the table frame is displayed.
- **rhs** Border on the right side of the table frame is displayed.
- **vsides** Borders on the left and right sides of the table frame are displayed.
Example

The following example sets the first table in the active document to display borders on only the left side of each cell in the table. This example assumes that there is at least one table in the active document.

ActiveDocument.all.tags("table").Item(0).frame = "lhs"
frameBorder Property

Returns or sets a **String** that represents whether a border, if present, is drawn as a three-dimensional frame. Corresponds to the **frameborder** attribute for a FRAMESET or FRAME element.

*expression.frameBorder*

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value for the **frameBorder** property can be one of the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No border.</td>
</tr>
<tr>
<td>1</td>
<td>Draws an inset border.</td>
</tr>
<tr>
<td>no</td>
<td>No border.</td>
</tr>
<tr>
<td>yes</td>
<td>Draws an inset border.</td>
</tr>
</tbody>
</table>
frames Property

Returns a **IHTMLFramesCollection2** object that represents the windows in which pages specified in FRAME elements are displayed.

```plaintext
expression.frames
```

*expression* Required. An expression that returns one of the objects in the Applies To list.
Example

The following example accesses the window in which the first frame in the active document is displayed.

Dim objFrame As FPHTMLWindow2

Set objFrame = ActivePageWindow.FrameWindow.frames.Item(0)
frameSpacing Property

Returns or sets a Variant that represents the amount of space between the frames in a FRAMESET element. Corresponds to the framespacing attribute.

expression.frameSpacing

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the space between the frames in the active document to five pixels.

```vba
Dim objFrameSets As IHTMLElementCollection
Dim objFrameSet As FPHTMLFrameSetSite

Set objFrameSets = ActiveDocument.all.tags("frameset")
Set objFrameSet = objFrameSets.Item(0)

objFrameSet.frameSpacing = "5"
```
frameType Property

Returns a **String** that represents how frames within a Web Part zone or within individual Web Parts within a zone are rendered in the browser.

`expression.frameType`

`expression` Required. An expression that returns one of the objects in the Applies To list.
**Remarks**

The **frameType** property can be one of the following **String** values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>empty</td>
<td>Resets the frameType property to &quot;Standard&quot;.</td>
</tr>
<tr>
<td>Standard</td>
<td>Renders both a title bar and a border around a Web Part.</td>
</tr>
<tr>
<td>TitleBarOnly</td>
<td>Renders only a title bar for a Web Part.</td>
</tr>
<tr>
<td>None</td>
<td>Renders all Web Parts in a zone without frames.</td>
</tr>
</tbody>
</table>
fromElement Property

Returns an IHTMLElement object that represents the element the mouse pointer is exiting during an onmouseover or onmouseout event.

expression.fromElement

expression  Required. An expression that returns an IHTMLEventObj object.
Remarks

See also the `srcElement` and `toElement` properties.
Example

The following example makes the text in the specified element bold, if it is not already bold. This example must be placed in a class module, and it assumes that you have declared in the General Declarations section of a class module an *FPHTMLParaElement* object variable called "objPara" using the * WithEvents* keyword.

Private Sub objPara_onmouseover()
    Dim objEvent As IHTMLEventObj
    Dim objElement As IHTMLElement

    Set objEvent = objWindow.event
    Set objElement = objEvent.fromElement

    If objElement.Style.fontWeight <> "bold" Then _
    objElement.Style.fontWeight = "bold"

End Sub
hasSharedBorders Property

Returns a Boolean that represents whether a specified document contains shared borders.

expression.hasSharedBorders

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

A shared border is a region on a Web page that is common to one or more pages in a Web site. A shared border may be a region at the top or bottom of the page (similar to a page header or footer), at the left, or at the right. Use shared borders to place the same content on multiple pages in one step, rather than editing each page.
Example

The following message displays a message to the user about whether the active document uses shared borders.

If ActiveDocument.hasSharedBorders = False Then
    MsgBox "This document doesn't use shared borders."
Else
    MsgBox "This document uses shared borders."
End If
height Property

Returns or sets a String that represents the height of the specified object. The `height` property corresponds to the `height` attribute of the specified object.

`expression.height`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **height** property can be one of the following **String** values:

- **height**: Integer that specifies the height of the object, in pixels.
- **percentage**: Integer, followed by a % sign. The value is a percentage of the height of the parent object.
Example

The following example sets the height of the first table in the active document to 100% of the height of the browser window. This example assumes that there is at least one table in the active document.

Dim objTable As FPHTMLTable
Set objTable = ActiveDocument.all.tags("table").Item(0)
objTable.Height = "100%"
hidden Property

Returns or sets a **String** that represents whether an embedded object is visible when displayed in a browser.

`expression.hidden`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Remarks

The String for the hidden property can be one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>Object is displayed.</td>
</tr>
<tr>
<td>true</td>
<td>Object is hidden.</td>
</tr>
</tbody>
</table>
Example

The following example hides the first EMBED element in the active document so that when the page is viewed in a browser, the embedded object will not be visible.

Dim objEmbed As FPHTMLEmbed

Set objEmbed = ActiveDocument.embeds.Item(0)
objEmbed.hidden = "true"
history Property

The **history** property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
hspace Property

Returns or sets a Long that represents the horizontal margin for the specified object. Corresponds to the hspace attribute.

expression\.hspace

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the horizontal margin for the first image in the active document to 25 pixels.

Dim objImage As FPHTMLImg
Set objImage = ActiveDocument.images(0)

objImage.hspace = "25"
htmlFor Property

Returns or sets a String that represents the object that is bound to a script, label, or form. Corresponds to the for attribute for the SCRIPT, LABEL, and FORM elements.

expression.htmlFor

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The `for` attribute of a SCRIPT, LABEL, or FORM element contains a unique identifier for a control on a page. For example, you might have an input box on a page with an `id` attribute of "email" and a text label next to it that says, "Please enter your email address:". When the `for` attribute of the LABEL element is set to be equal to the `id` attribute of the INPUT element, if a user clicks on the label when the page is viewed in the browser, the browser will give the corresponding INPUT element the focus. The following HTML code illustrates this.

```html
<p><label id="lblemail">email address</label></p>
<p><input type="text" id="email"></p>
```
Example

The following example assigns an access key and specifies the corresponding control on the page to which the LABEL element applies. You can run this example against the above HTML code without error. Otherwise, this example assumes that you have a LABEL element in the active document with an id attribute of "lblemail".

Dim objLabel As FPHTMLLabelElement

Set objLabel = ActiveDocument.all.tags("label").Item("lblemail")
objLabel.accessKey = "e"
obLabel.htmlFor = "email"
**htmlText Property**

Returns a **String** that represents the HTML contained in the specified **IHTMLTxtRange** object, or if text is selected without HTML, returns the selected text with the HTML for the parent element.

*expression*.htmlText

*expression*  Required. An expression that returns an **IHTMLTxtRange** object.
Example

The following example displays the selected text and its surrounding HTML tags.

Dim objRange As IHTMLTxtRange
Set objRange = ActiveDocument.Selection.createRange
MsgBox objRange.htmlText
**htmlViewText Property**

Returns or sets a String that represents the HTML contained in a specified document.

`expression.htmlViewText`

*expression* Required. An expression that returns one of the objects in the Applies To list.
Remarks

The `htmlViewText` property returns the same results as the `DocumentHTML` property. However, when you set the value of the `DocumentHTML` property equal to a `String`, the HTML in the document is changed to the value of the new string, while when you set the `htmlViewText` property equal to a `String`, the value returned is a `Boolean`. This represents the comparison of the value of the HTML contained in the document to the `String` assigned to the `htmlViewText` property. `True` indicates that the two strings are the same. `False` indicates that the two strings are different.

For example, the following code displays a message box that indicates whether the `String` value assigned to the `htmlViewText` property is the same as the HTML contained in the document.

```vbscript
MsgBox ActiveDocument.htmlViewText = "Hello"
```
httpEquiv Property

Sets or returns a **String** that represents the information used to bind the content of a META element to an HTTP response header.

**expression.httpEquiv**

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The `httpEquiv` property contains the value of the `httpEquiv` attribute. Use the `content` and `charset` properties to specify the `content` and `charset` attributes, respectively.
Example

The following example inserts a <META> tag that contains the character set to use for the active document.

Sub InsertCharset(ByRef objDoc As FPHTMLDocument, ByRef strID As String, ByRef strHTTP As String, ByRef strContent As String, ByRef strCharset As String)
    Dim objMeta As FPHTMLMetaElement
    objDoc.all.tags("head").Item(0).innerHTML = "<META id="""" & strID & """">"
    Set objMeta = ActiveDocument.all.tags("meta").Item(CVar(strID))
    With objMeta
        .httpEquiv = strHTTP
        .content = strContent
        .Charset = strCharset
    End With
End Sub

Use the following example to call the preceding subroutine.

Sub CallInsertCharset()
    Call InsertCharset(objDoc:=ActiveDocument, strID:="iso_content", strHTTP:="Content-Type", strContent:="text/html", strCharset
End Sub
ignoreSharedBorders Property

Returns or sets a Boolean that represents whether to display shared borders in a Web page. True indicates that shared borders will not be displayed.

expression.ignoreSharedBorders

expression Required. An expression that returns one of the objects in the Applies To list.
Image Property

The **Image** property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
images Property

Returns an IHTMLElementCollection object that represents all the IMG elements in the specified document.

expression.images

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example changes the src attribute of the first image in the active document. This example assumes that there is at least one image in the active document.

Dim objImage As FPHTMLImg

Set objImage = ActiveDocument.images.Item(0)
objImage.src = "graphics/chelan.jpg"
imports Property

Returns an `FPHTMLStyleSheetsCollection` collection that represents STYLE elements that contain the `@import` statement.

`expression.imports`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Remarks

Use the `addImport` method to add an imported style sheet to the collection of imported style sheets for a STYLE element. Use the `removeImport` method to remove an imported style sheet from the collection.
Example

The following example checks to see if there are any imported style sheets referenced in the first STYLE element in the active document. If there are no imported stylesheets, a new imported style sheet reference is added to the STYLE element.

Dim objStyle As FHTMLStyleSheet
Set objStyle = ActiveDocument.styleSheets.Item(0)

If objStyle.imports.Length = 0 Then
    objStyle.addImport("web.css")
End If
index Property

The index property is unsupported in Microsoft FrontPage and is therefore unavailable for programmatic access.
innerText Property

Sets or returns a **String** that represents the text between the start and end tags of a specified object without any associated HTML.

*expression*.innerText

*expression*   Required. An expression that returns one of the objects in the Applies To list.
Remarks

See also the `innerHTML`, `outerText`, and `outerHTML` properties.
Example

The following example changes the text inside the first <H1> tag in the active document.

Dim objTag As IHTMLElement
Set objTag = ActiveDocument.all.tags("h1").Item(0)

objTag.innerText = "Microsoft Home Page"

The following example retrieves the active element in the active document, creates a String from the text contained in the active element, less any spaces, and then places a bookmark anchor, with a name attribute equal to the new String, around the text the active element contains.

Dim objElement As IHTMLElement
Dim strElement As String

Set objElement = ActiveDocument.activeElement
strElement = Trim(Replace(objElement.innerText, " ", ""))

With objElement
  .innerHTML = "<A name=" & strElement & ">" & .innerHTML & "</A>
End With
**InternalRep Property**

Returns a **Variant** that represents an internal pointer that is used for communicating state information between the browser window and the Microsoft FrontPage editing window.

`expression.InternalRep`

- `expression` Required. An expression that returns a **DynamicTemplateState** object.
**isCell Property**

Returns a **Boolean** that represents whether a layout section in a document is a layout cell in a document.

`expression.isCell`

`expression` Required. An expression that returns an **FPLayoutRegion** object.
**isDirty Property**

Returns **True** if the specified document has been modified since it was last saved. Read/write **Boolean**.

*expression*.external

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example saves the active document if the user has made changes.

Private Sub GetExternal()
    Dim objPageWindow As PageWindow
    Set objPageWindow = ActiveDocument.parentWindow.external
    If objPageWindow.IsDirty = True Then
        objPageWindow.Save
    End If
End Sub
isMap Property

Returns or sets a **Boolean** that represents whether the image uses an image map stored on the Web server. **True** indicates that the image uses a server-side map. Corresponds to the **ismap** attribute of a IMG element.

`expression.isMap`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

HTML provides two types of image maps: server-side and client-side. A server-side map is specified by setting the `ismap` attribute to true in the IMG tag and creating an image map file on the server. Use the `isMap` property to specify whether an image uses a server-side image map.
Example

The following example specifies that the first image in the active document uses a server-side image map. This example assumes that there is at least one image in the active document.

Dim objImage As FPHTMLImg

Set objImage = ActiveDocument.images.Item(0)
objImage.isMap = True
isTable Property

Returns a **Boolean** that represents whether a layout section in a document is a cell formatting table.

*expression*.isTable

*expression*  Required. An expression that returns an **FPLayoutRegion** object.
isTextEdit Property

Returns a `Boolean` that represents whether you can create an `IHTMLTxtRange` object with the given object. `True` indicates that you can create an `IHTMLTxtRange` object.

`expression.isTextEdit`

`expression` Required. An expression that returns one of the objects in the Applies To list.
keyCode Property

Returns or sets a **Long** that represents the Unicode key code associated with the key that caused the event.

*expression*.keyCode

*expression*  Required. An expression that returns an **IHTMLEventObj** object.
**lang Property**

Returns or sets a **String** that represents the language to use for the specified object. Corresponds to the **lang** attribute of the associated HTML element.

`expression.lang`

*expression* Required. An expression that returns one of the objects in the Applies To list.
Remarks

The `lang` property can be any of the language codes available to browsers. For information on language codes, see [Language Codes](#) in the MSDN Library.
**language Property**

Returns or sets a **String** that represents the language in which the script for the specified object is written.

**expression.language**

**expression** Required. An expression that returns one of the objects in the Applies To list.
## Remarks

The **language** property can be one of the following **String** values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C#</td>
<td>Script is written in C#.</td>
</tr>
<tr>
<td>CS</td>
<td>Script is written in C#.</td>
</tr>
<tr>
<td>CSharp</td>
<td>Script is written in C#.</td>
</tr>
<tr>
<td>JScript</td>
<td>Script is written in JScript.</td>
</tr>
<tr>
<td>JavaScript</td>
<td>Script is written in JavaScript.</td>
</tr>
<tr>
<td>JS</td>
<td>Script is written in JavaScript.</td>
</tr>
<tr>
<td>VBScript</td>
<td>Script is written in VBScript.</td>
</tr>
<tr>
<td>VB</td>
<td>Script is written in Microsoft Visual Basic.</td>
</tr>
<tr>
<td>VisualBasic</td>
<td>Script is written in Visual Basic.</td>
</tr>
<tr>
<td>XML</td>
<td>Script is written in XML.</td>
</tr>
<tr>
<td><strong>language</strong></td>
<td>Script is a browser-supported language.</td>
</tr>
</tbody>
</table>
Example

The following example sets the language to "JavaScript" for the first SCRIPT element in the active document. This example assumes that there is at least one script element in the active document.

Dim objScript As FPHTMLScriptElement
Set objScript = ActiveDocument.Scripts.Item(0)
objScript.Language = "JavaScript"
lastModified Property

Returns a String that represents the date the page was last modified, in the form "DD MMM YYYYY hh:mm:ss".

expression:lastModified

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example inserts text at the bottom of the active document specifying when the document was last modified.

ActiveDocument.body.insertAdjacentHTML "beforeend", "This document was last changed on: " & ActiveDocument.LastModif
layoutTablesAndCells Property

Returns an **FPLayoutTablesAndCells** collection that represents a collection of all the layout cells and cell formatting tables in a document.

\[ \text{expression} \cdot \text{layoutTablesAndCells} \]

**expression**  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example inserts a cell formatting table into the active document at
the insertion point.

ActiveDocument.layoutTablesAndCells.insertLayoutTable 100, 100
left Property

As it applies to the FPHTMLStyle and IHTMLRuleStyle objects.

Returns or sets a Variant that represents the position of the left edge of an element for a cascading style sheet.

expression.left

expression Required. An expression that returns one of the objects in the Applies To list.

As it applies to the IFPStyleState object.

Sets or returns an IFPStyleLength object that represents the left edge of an element. Places a DIV element around the specified element.

expression.left

expression Required. An expression that returns an IFPStyleState object.
Remarks

You must also set the `position` property of the `IFPStyleState` object for the `left` property to take effect for an element.
Example

As it applies to the **IFPStyleState** object.

The following example surrounds the first paragraph in the active document with a DIV element positions the left edge of the DIV element 100 pixels from the left edge of the page.

```vba
Dim objStyleState As IFPStyleState
Dim objLeft As IFPStyleLength
Dim rng As IHTMLTxtRange

Set objStyleState = ActiveDocument.createStyleState
Set rng = ActiveDocument.Selection.createRange

objStyleState.gatherFromElement _
    ActiveDocument.all.tags("p").Item(0)

Set objLeft = objStyleState.createStyleLength
objLeft.units = fpLengthUnitsPX
objLeft.Value = 100

objStyleState.Position = "Absolute"
objStyleState.Left = objLeft

objStyleState.Apply
```
leftMargin Property

Sets or returns a String that represents the number of pixels for the left margin of the page.

expression.leftMargin

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the bottom, top, left, and right margins to 50 pixels.

```plaintext
With ActiveDocument.body
    .bottomMargin = "50"
    .leftMargin = "50"
    .rightMargin = "50"
    .topMargin = "50"
End With
```
Length Property

Returns a Long that represents the number of objects in a specified collection.

expression.Length

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example inserts a MARQUEE element that contains the selected text. It uses the **Length** property to get the number of MARQUEE elements already in the document.

```vbs
Sub InsertMarqueeWithSelectedText()
    Dim objElement As IHTMLElement
    Dim objRange As IHTMLTxtRange
    Dim objMarquee As FPHTMLMarqueeElement
    Dim lngCount As Long
    Dim strID As String

    lngCount = ActiveDocument.all.tags("marquee").Length
    strID = "marquee" & lngCount + 1

    Set objRange = ActiveDocument.selection.createRange

    objRange.pasteHTML "<marquee id="" & strID & """>" & _
            objRange.Text & "</marquee>"

End Sub
```
letterSpacing Property

As it applies to the FPHTMLStyle and IHTMLRuleStyle objects

Returns or sets a Variant that represents the amount of space between letters in the specified object.

expression.letterSpacing

expression Required. An expression that returns one of the objects in the Applies To list.
# Remarks

The **String** for the **letterSpacing** property can be one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>Default spacing. Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). For more information about the supported length units, see CSS Length Units.</td>
</tr>
</tbody>
</table>

As it applies to the **IFPStyleState** object.

Returns or sets an **IFPStyleLength** object that represents the amount of space between letters in the specified object.

```
expression.letterSpacing
```

*expression* Required. An expression that returns one of the objects in the Applies To list.
Example

As it relates to the FPHTMLStyle object

The following example sets the spacing for the letters in the first paragraph in the active document to 5 pixels.

Dim objPara As FPHTMLParaElement

Set objPara = ActiveDocument.all.tags("p").Item(0)

objPara.Style.letterSpacing = "5px"
lineHeight Property

As it applies to the FPHTMLStyle and IHTMLRuleStyle objects.

Returns or sets a Variant that represents the distance between lines in the object. Corresponds to the line-height property for the style attribute.

expression.lineHeight

expression  Required. An expression that returns one of the objects in the Applies To list.
## Remarks

The String value for the `lineHeight` property can be one of the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>normal</code></td>
<td>Default height.</td>
</tr>
<tr>
<td><code>height</code></td>
<td>A floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex).</td>
</tr>
<tr>
<td><code>percentage</code></td>
<td>A integer, followed by a %. The value is a percentage of the height of the parent object.</td>
</tr>
</tbody>
</table>

As it applies to the `IFPStyleState` object.

Returns an `IFPStyleLength` object that represents the line height for the specified object.

`expression.lineHeight`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Example

As it applies to the **FPHTMLStyle** object.

The following example changes the line height in the active document to 10 pt.

```vba
Sub SetLineHeight()
    ActiveDocument.body.Style.lineHeight = "10pt"
End Sub
```
link Property

Returns or sets a Variant that represents the color of hyperlinks contained in a document. Corresponds to the link attribute for the BODY element.

expression.link

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

Performs the same function as the linkColor property.
Example

The following example makes the hyperlinks in the active document red.

Sub SetLinkColor()
    Dim objBody As FPHTMLBody
    Set objBody = ActiveDocument.body
    objBody.link = "red"
End Sub
linkColor Property

Returns or sets a **Variant** that represents the color of hyperlinks in the specified document. Corresponds to the **link** attribute for the BODY element.

*expression*.linkColor

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

Performs the same function as the link property.
Example

The following example makes the hyperlink color in the active document aqua.

Sub SetHyperlinkColor()
    Dim objDocument As FPHTMLDocument

    Set objDocument = ActiveDocument

    objDocument/linkColor = "aqua"
End Sub
links Property

Returns an IHTMLElementCollection collection that represents the collection of hyperlinks in a page.

expression.links

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example loops through all the hyperlinks in the active document and adds an id attribute to it.

Sub LoopThroughLinks()
    Dim objLink As FPHTMLAnchorElement
    Dim intCount As Integer
    Dim objLinks As IHTMLElementCollection

    Set objLinks = ActiveDocument.Links

    For intCount = 0 To objLinks.Length - 1
        Set objLink = objLinks.Item(intCount)

        objLink.Id = "hyperlink" & intCount + 1
    Next
End Sub
listStyle Property

Returns or sets a String that represents the setting for a liststyle attribute for an inline style.

expression.listStyle

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value for the **listStyle** property can be a space-delimited list of up to three of the following, in any order:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>type</code></td>
<td>Any of the range of type values available to the <strong>listStyleType</strong> property.</td>
</tr>
<tr>
<td><code>position</code></td>
<td>Any of the range of position values available to the <strong>listStylePosition</strong> property.</td>
</tr>
<tr>
<td><code>image</code></td>
<td>Any of the range of image values available to the <strong>listStyleImage</strong> property.</td>
</tr>
</tbody>
</table>
Example

The following example inserts an ordered list into the active document and sets the list style properties to show an uppercase Roman numeral so that wrapped lines are not even with the number.

Sub SetOrderedListProperties()
    Dim objList As FPHTMLOListElement
    Dim strList As String

    strList = "<ol id="neworderedlist">" & _

    ActiveDocument.activeElement.insertAdjacentHTML _
        where:="afterbegin", HTML:=strList

    Set objList = ActiveDocument.all.tags("ol").Item _
        ("neworderedlist")

    objList.Style.listStyle = "upper-roman outside"
End Sub
listStyleImage Property

Returns or sets a String that represents the image to use as a bullet in an unordered list.

*expression*.listStyleImage

*expression*  Required. An expression that returns one of the objects in the Applies To list.
**Remarks**

String that specifies one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>No image is specified.</td>
</tr>
<tr>
<td>url(strURL)</td>
<td>Location of the image, where strURL is an absolute or relative URL.</td>
</tr>
</tbody>
</table>
Example

The following example inserts an unordered list into the active document before the currently selected element, and then specifies an image to display for the bullet, specifies that the bullet is displayed even with the wrapped lines, and then specifies that if the image file cannot be found that a circle bullet displays in its place.

Sub SetBulletedListProperties()
  Dim objList As FPHTMLULListElement
  Dim strList As String


  ActiveDocument.activeElement.insertAdjacentHTML _
  where:="beforebegin", HTML:=strList

  Set objList = ActiveDocument.all.tags("ul").Item("newlist")

  With objList.Style
    .listStyleImage = "sqbullet.gif"
    .listStylePosition = "inside"
    .listStyleType = "circle"
  End With
End Sub
listStylePosition Property

Returns or sets a String that represents the position of the bullet in relation to where text is wrapped for each line in a list.

expression.listStylePosition

expression  Required. An expression that returns one of the objects in the Applies To list.
**Remarks**

The **String** value for the **listStylePosition** property can be any of the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>outside</td>
<td>Places the bullet outside the text, and any wrapping text is not aligned</td>
</tr>
<tr>
<td></td>
<td>under the marker.</td>
</tr>
<tr>
<td>inside</td>
<td>Places the bullet inside the text, and any wrapping text is aligned under</td>
</tr>
<tr>
<td></td>
<td>the marker</td>
</tr>
</tbody>
</table>
Example

The following example inserts an unordered list into the active document before the currently selected element, and then specifies an image to display for the bullet, specifies that the bullet is displayed even with the wrapped lines, and then specifies that if the image file cannot be found that a circle bullet displays in its place.

Sub SetBulletedListProperties()
    Dim objList As FPHTMLULListElement
    Dim strList As String

    strList = "<ul id="newlist">" & _

    ActiveDocument.activeElement.insertAdjacentHTML _
    where:="beforebegin", HTML:=strList

    Set objList = ActiveDocument.all.tags("ul").Item("newlist")

    With objList.Style
        .listStyleImage = "sqbullet.gif"
        .listStylePosition = "inside"
        .listStyleType = "circle"
    End With
End Sub
**listStyleType Property**

Returns or sets a **String** that represents the type of bullet that is displayed for an ordered or unordered list.

`expression.listStyleType`

*expression*   Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value for the **listStyleType** property can be one of the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disc</td>
<td>Displays solid circles. Default.</td>
</tr>
<tr>
<td>circle</td>
<td>Displays outlined circles.</td>
</tr>
<tr>
<td>square</td>
<td>Displays solid squares.</td>
</tr>
<tr>
<td>decimal</td>
<td>Displays Arabic numerals; 1, 2, 3, 4, and so on.</td>
</tr>
<tr>
<td>lower-roman</td>
<td>Displays lower case Roman numerals; i, ii, iii, iv, and so on.</td>
</tr>
<tr>
<td>upper-roman</td>
<td>Displays upper case Roman numerals; I, II, III, IV, and so on.</td>
</tr>
<tr>
<td>lower-alpha</td>
<td>Displays lower case Latin alphabet; a, b, c, d, and so on.</td>
</tr>
<tr>
<td>upper-alpha</td>
<td>Displays upper case Latin alphabet; A, B, C, D, and so on.</td>
</tr>
<tr>
<td>none</td>
<td>Displays no marker.</td>
</tr>
</tbody>
</table>

**Note**  If you specify a value for the **listStyleImage** and the **listStyleType** properties, the **listStyleImage** property takes precedence when a page is displayed in a browser, unless the specified image can't be found, in which case the list would display as specified in the **listStyleType** property.
Example

The following example inserts an unordered list into the active document before the currently selected element, and then specifies an image to display for the bullet, specifies that the bullet is displayed even with the wrapped lines, and then specifies that if the image file cannot be found that a circle bullet displays in its place.

Sub SetBulletedListProperties()
    Dim objList As FPHTMLULListElement
    Dim strList As String

    strList = "<ul id="newlist">" & _

    ActiveDocument.activeElement.insertAdjacentHTML _
        where:="beforebegin", HTML:=strList

    Set objList = ActiveDocument.all.tags("ul").Item("newlist")

    With objList.Style
        .listStyleImage = "sqbullet.gif"
        .listStylePosition = "inside"
        .listStyleType = "circle"
    End With
End Sub
location Property

Returns an FPHTMLLocation object that represents the properties of the URL of the an HTML document.

expression.location

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example reloads the specified document.

Note When you use the `reload` method on a saved document that has been changed but not resaved, a message appears asking if the user wants to revert to the previously saved version.

Dim objLocation As FPHTMLLocation

Set objLocation = ActiveDocument.Location

objLocation.reload
lockLayout Property

Sets or returns a **Boolean** that represents whether Web Parts within a zone can be added, deleted, resized, or moved when designing a Web Part Page inside of the browser; however, non-layout properties on Web Parts can still be changed. **True** indicates they cannot be added, deleted, resized, or moved.

**Note** The lockLayout property applies only to pages within a Web site based on Microsoft Windows SharePoint Services.

`expression.lockLayout`

`expression` Required. An expression that returns one of the objects in the Applies To list.
loop Property

Sets or returns an **Integer** that represents the number of times a marquee or a sound or video clip will loop when activated.

\[ expression.loop \]

*expression*   Required. An expression that returns one of the objects in the Applies To list.
Remarks

The default value of the `loop` property is 1. Setting the `loop` property to -1 will cause the specified object to loop indefinitely.
Example

The following example adds a MARQUEE element to the active document and then sets the behavior, direction, number of times to loop, height, width, and border and font formatting of the new element.

Sub SetMarqueeValues()
    Dim objMarquee As FPHTMLMarqueeElement
    ActiveDocument.body.insertAdjacentHTML where:="afterbegin", _
    HTML:="<marquee id="newmarquee"></marquee>"
    Set objMarquee = ActiveDocument.all.tags("marquee").Item("newmarq
With objMarquee
    .behavior = "slide"
    .direction = "up"
    .loop = 5
    .Height = "100%"
    .Width = "10%"
    With .Style
        .verticalAlign = "middle"
        .fontStyle = "italic"
        .Border = "dashed thick red"
    End With
    .innerText = "This is a scrolling Marquee."
End With
End Sub
lowsrc Property

Sets or returns a **String** that represents a lower resolution image to display. Corresponds to the **lowsrc** attribute.

```
expression.lowsrc
```

*expression* Required. An expression that returns one of the objects in the Applies To list.
margin Property

Returns or sets a String that represents the width of the top, bottom, left, and right margins for the specified object. Corresponds to the margin property of an inline style attribute.

expression.margin

expression Required. An expression that returns one of the objects in the Applies To list.
## Remarks

The **String** for the **margin** property can be one or more of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>Value of all margins is the same as the default setting.</td>
</tr>
</tbody>
</table>

The following code sets all margins in the active document to the same default setting.

```javascript
ActiveDocument.body.Style.margin = "auto"
```

Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex).

You can set all margins to the same measurement. For example, the following code sets the all margins **margin** property of the active document to 50 pixels. This setting affects all margins.

```javascript
ActiveDocument.body.Style.margin = "50px"
```

You can set each margin separately by specifying a measurement for each of the four margins. Using a space to separate the individual measurements, specify the margin settings starting with the top margin and working clockwise. For example, the following code sets the top margin to 50 pixels, the right margin to 10 pixels, the bottom margin to 100 pixels, and the left margin to 0 pixels.

```javascript
ActiveDocument.body.Style.margin = "50px 10px 100px 0px"
```

Integer, followed by a %. The value is a percentage of the width (for left and right margins) or height (for top and bottom margins) of the parent object.

The following code sets all margins to 10 percent of the width or height of the browser window.

```javascript
ActiveDocument.body.Style.margin = "10%"
```
The following code sets the top margin to 10 percent of the height of the browser window, the right margin to 20 percent of the width of the browser window, the bottom margin to 30 percent of the height of the browser window, and the left margin to 40 percent of the width of the browser window.

`ActiveDocument.body.Style.margin = "10% 20% 30%, 40%"

In addition to specifying all margin settings at the same time, you can mix and match the above measurements as necessary. For example, the following code sets the top margin to 10 pixels, the right margin to five percent of the width of the browser window, the bottom margin to the default setting for the document, and the left margin to one and a half inches.

`ActiveDocument.body.Style.margin = "10px 5% auto 1.5in"`
marginBottom Property

Returns or sets a String that represents the width of the bottom margin for the specified object. Corresponds to the marginBottom property of an inline style attribute.

(expression).MarginBottom

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** for the **marginBottom** property can be one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>Bottom margin measurement is the default setting.</td>
</tr>
</tbody>
</table>
| width    | The following code sets the bottom margin in the active document to the default setting.  

```
ActiveDocument.body.Style.marginBottom = "auto"
```

Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex).

```
ActiveDocument.body.Style.marginBottom = "50px"
```

Integer, followed by a %. The value is a percentage of the height of the parent object.

```
ActiveDocument.body.Style.marginBottom = "10%"
```

*You can also use the **margin** property to set all margins at the same time. For example, the following code does the same as the example below but uses the **margin** property.*

```
ActiveDocument.body.Style.margin = "auto 25px 2px 50px"
```
**Example**

The following example sets the top, right, bottom, and left margins for the active document.

```vba
With ActiveDocument.body.Style
    MarginTop = "auto"
    MarginRight = "25px"
    **MarginBottom** = "1%"
    MarginLeft = "50px"
End With
```
marginHeight Property

Returns or sets a Variant that represents the height of the top and bottom margins for an IFRAME element.

expression.MarginHeight

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The value of the `marginHeight` property can be a `String` that contains a number that represents the width of the margins followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). Alternatively, you can specify an `Integer` that represents the height of the margins in pixels.

For example, the following two code samples both set the height of the top and bottom margins of an IFRAME element to 10 pixels.

```javascript
marginHeight = "10px"
marginHeight = 10
```
Example

The following example sets the margin height and width and the source file for the specified IFRAME element.

Dim objIFrame As FPHTMLIFrame
Set objIFrame = ActiveDocument.body.all.tags("iframe").Item(0)

With objIFrame
    .marginHeight = "10px"
    .marginWidth = "10px"
    .src = strSource
End With
**marginRight Property**

Returns or sets a **String** that represents the width of the right margin for the specified object. Corresponds to the `marginRight` property of an inline **style** attribute.

`expression.marginRight`

`expression` Required. An expression that returns one of the objects in the Applies To list.
## Remarks

The **String** for the **marginRight** property can be one or more of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>Right margin measurement is the default setting.</td>
</tr>
<tr>
<td></td>
<td>The following code sets the right margin in the active document to</td>
</tr>
<tr>
<td></td>
<td>the default setting.</td>
</tr>
<tr>
<td></td>
<td>ActiveDocument.body.Style.marginRight = &quot;auto&quot;</td>
</tr>
<tr>
<td>width</td>
<td>Floating-point number, followed by an absolute units designator (cm, mm, in,</td>
</tr>
<tr>
<td></td>
<td>pt, pc, or px) or a relative units designator (em or ex).</td>
</tr>
<tr>
<td></td>
<td>The following code sets the right margin of the active document to</td>
</tr>
<tr>
<td></td>
<td>50 pixels.</td>
</tr>
<tr>
<td></td>
<td>ActiveDocument.body.Style.marginRight = &quot;50px&quot;</td>
</tr>
<tr>
<td>percentage</td>
<td>Integer, followed by a %. The value is a percentage of the width of</td>
</tr>
<tr>
<td></td>
<td>the parent object.</td>
</tr>
<tr>
<td></td>
<td>The following code sets the right margin to 10 percent of the width of</td>
</tr>
<tr>
<td></td>
<td>the browser window.</td>
</tr>
<tr>
<td></td>
<td>ActiveDocument.body.Style.marginRight = &quot;10%&quot;</td>
</tr>
</tbody>
</table>

You can also use the **margin** property to set all margins at the same time. For example, the following code does the same as the example below but uses the **margin** property.

ActiveDocument.body.Style.margin = "auto 25px 2px 50px"
Example

The following example sets the top, right, bottom, and left margins for the active document.

With ActiveDocument.body.Style
    .MarginTop = "auto"
    .MarginRight = "25px"
    .MarginBottom = "1%"
    .MarginLeft = "50px"
End With
marginTop Property

Returns or sets a **String** that represents the width of the top margin for the specified object. Corresponds to the **marginTop** property of an inline **style** attribute.

`expression.MarginTop`

`expression` Required. An expression that returns one of the objects in the Applies To list.
## Remarks

The **String** for the **marginTop** property can be one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>auto</code></td>
<td>Bottom margin measurement is the default setting.</td>
</tr>
<tr>
<td><code>width</code></td>
<td>The following code sets the top margin in the active document to the default setting.</td>
</tr>
<tr>
<td></td>
<td><code>ActiveDocument.body.Style.marginTop = &quot;auto&quot;</code></td>
</tr>
<tr>
<td></td>
<td>Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex).</td>
</tr>
<tr>
<td></td>
<td>The following code sets the top margin of the active document to 50 pixels.</td>
</tr>
<tr>
<td></td>
<td><code>ActiveDocument.body.Style.marginTop = &quot;50px&quot;</code></td>
</tr>
<tr>
<td><code>percentage</code></td>
<td>Integer, followed by a %. The value is a percentage of the height of the parent object.</td>
</tr>
<tr>
<td></td>
<td>The following code sets the top margin to 10 percent of the height of the browser window.</td>
</tr>
<tr>
<td></td>
<td><code>ActiveDocument.body.Style.marginTop = &quot;10%&quot;</code></td>
</tr>
</tbody>
</table>

You can also use the **margin** property to set all margins at the same time. For example, the following code does the same as the example below but uses the **margin** property.

`ActiveDocument.body.Style.margin = "auto 25px 2px 50px"`
**Example**

The following example sets the top, right, bottom, and left margins for the active document.

```vbnet
With ActiveDocument.body.Style
    .MarginTop = "auto"
    .MarginRight = "25px"
    .MarginBottom = "1%"
    .MarginLeft = "50px"
End With
```
marginWidth Property

Returns or sets a Variant that represents the width of the left and right margins for an IFRAME element.

expression.marginWidth

description Required. An expression that returns one of the objects in the Applies To list.
Remarks

The value of the `marginWidth` property can be a `String` that contains a number that represents the width of the margins followed by an absolute units designator (`cm`, `mm`, `in`, `pt`, `pc`, or `px`) or a relative units designator (`em` or `ex`). Alternatively, you can specify an `Integer` that represents the width of the margins in pixels.

For example, the following two code samples both set the width of the left and right margins of a IFRAME element to 10 pixels.

```javascript
marginWidth = "10px"
marginWidth = 10
```
Example

The following example sets the margin height and width and the source file for the specified IFRAME element.

Dim objIFrame As FPHTMLIFrame
Set objIFrame = ActiveDocument.body.all.tags("iframe").Item(0)

With objIFrame
    .marginHeight = "10px"
    .marginWidth = "10px"
    .src = strSource
End With
MaxLength Property (Page Object Model)

Returns or sets a Long that represents the maximum number of characters that a user can enter into an input control.

expression.MaxLength

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

The `maxLength` property limits the number of characters a user can enter into a text box and has a default value of no limit.
Example

The following example specifies the maximum number of characters for the first INPUT element in the active document. This assumes that there is at least one INPUT element in the active document and that the first INPUT element is a text box.

Dim objInput As FPHTMLInputTextElement
Set objInput = ActiveDocument.all.tags("input").Item(0)

objInput.MaxLength = 50
**media Property**

Returns or sets a string that represents the type of media in which the content will be displayed. Corresponds to the `media` attribute for the STYLE and LINK elements.

`expression.media`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value for the **media** property can be one of the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>screen</td>
<td>Output is intended for computer screens.</td>
</tr>
<tr>
<td>print</td>
<td>Output is intended for printed material and for documents on screen viewed in Print Preview mode.</td>
</tr>
<tr>
<td>all</td>
<td>Applies to all devices.</td>
</tr>
</tbody>
</table>
Remarks

If the **media** property has no value, the style sheet applies to all views of the document.
**Example**

The following example specifies that the styles contained in the first STYLE element in the active document apply to when the page is printed or viewed in print preview mode.

```
Dim objHead As IHTMLElement
Dim objStyle As FPHTMLStyleElement

Set objHead = ActiveDocument.all.tags("head").Item(0)
Set objStyle = objHead.all.tags("style").Item("printmedia")

objStyle.media = "print"
```
The `menuArguments` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**method Property**

Returns or sets a **String** that represents how to send the data in a form to the server.

*expression.method*

*expression* Required. An expression that returns one of the objects in the Applies To list.
## Remarks

The **String** value for the **method** property can be one of the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>get</td>
<td>Append the arguments to the action URL and open it as if it were an anchor.</td>
</tr>
<tr>
<td>post</td>
<td>Send the data through an HTTP post transaction.</td>
</tr>
</tbody>
</table>
**Example**

The following example inserts a FORM element into the active document and sets the `method` property.

```vbscript
Sub AddForm()
    Dim objForm As FPHTMLFormElement
    Dim objTextBox As FPHTMLInputTextElement
    Dim intCounter As Integer
    Dim strForm As String

    strForm = "<form id="NewCustomer"><input type="text" id="FavoriteIceCream"><input type="submit"></form>"

    ActiveDocument.body.insertAdjacentHTML where:="afterbegin", HTML:=strForm

    Set objForm = ActiveDocument.forms("NewCustomer")

    objForm.method = "post"
End Sub
```
Methods Property

Returns or sets a **String** that represents the list of HTTP methods supported by a hyperlink. Corresponds to the **methods** attribute for an **A** element.

`expression.Methods`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

Setting the **Methods** property sets the value of the **methods** attribute for a specified hyperlink.
Example

The following example creates an array of the items listed in the *Methods* property.

```vba
Dim objLink As FPHTMLAnchorElement
Dim strMethods() As Variant

Set objLink = ActiveDocument.Links.Item(0)
strMethods = Split(objLink.Methods, "",""
```
**mimeType Property**

The `mimeType` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
mimeTypes Property

The `mimeTypes` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
mseMode Property

Returns a **Boolean** that indicates if the current document is in MSE mode. MSE mode corresponds to the Microsoft Script Editor. If **True**, the document is open in Microsoft Script Editor. If **False**, the document is not open in Microsoft Script Editor.

`expression.mseMode`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
**Example**

The following example displays a message to the user indicating if the active document is open in Microsoft Script Editor.

```
Sub DisplayMsemode()
    If ActiveDocument.mseMode = False Then
        MsgBox "The current document is not in open in Microsoft Scr"
    Else
        MsgBox "The current document is open in Microsoft Script Edi"
    End If
End Sub
```
multiple Property

Returns or sets a Boolean that represents whether multiple items can be selected from a list.

\texttt{expression.multiple}

\textit{expression} Required. An expression that returns one of the objects in the Applies To list.
Example

The following example inserts a list box into the active document and then specifies that users can select multiple items from the list, the number of items to display in the list at a time, and the script to run when the selection changes.

Sub AddListBox()
    Dim objListBox As FPHTMLSelectElement
    Dim strHTML As String

    strHTML = "<SELECT ID=""pets"">" & "<OPTION VALUE=""1"">Cat" & _
          vbCrLf & "<OPTION VALUE=""2"">Dog" & vbCrLf & _
          "<OPTION VALUE=""3"">Snake" & vbCrLf & "</SELECT>"

    ActiveDocument.body.insertAdjacentHTML _
        where:="beforeend", HTML:=strHTML

    Set objListBox = ActiveDocument.all.tags("select").Item("pets")

    With objListBox
        .multiple = True
        .Size = "6"
    End With
End Sub
n Property

The n property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
name Property

Returns or sets a String that represents the name of a bookmark in an HTML document.

expression.name

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

Bookmarks are A elements with the `name` attribute and are members of the collection returned by using the `anchors` property. Hyperlinks are A elements with the `href` attribute and are members of the collection returned by using the `links` property.
Example

The following example inserts a bookmark into the active document.

Dim objBookmark As FPHTMLAnchorElement
ActiveDocument.body.insertAdjacentHTML "beforeend", _
"<a id="newbookmark"">Bookmark</a>"
Set objBookmark = ActiveDocument.anchors.Item("newbookmark")
objBookmark.Name = "newBookmark"
nameProp Property

Returns a String that represents the file name of the specified document.

expression.nameProp

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example inserts the file name of the active document into the document.

Sub InsertDocumentName()
    With ActiveDocument
        .body.insertAdjacentHTML where:="beforeend", _
        HTML:="<p id="newpara2">" & .nameProp & "</p>"
    End With
End Sub
namespace Property

Returns a **String** that represents the namespace attribute of the Register directive (i.e., "<%@ Register … %>") for the Web Part.

*expression*.namespace

*expression*  Required. An expression that returns one of the objects in the Applies To list.
**navigator Property**

Returns an `FPHTMLNavigator` object that represents the browser in which a document is displayed.

`expression.navigator`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example displays the browser application name for the active document, which in this case is "Microsoft FrontPage".

Dim objNav As FPHTMLNavigator

Set objNav = ActiveDocument.parentWindow.navigator

MsgBox objNav.appName
**ncssBdo Property**

Sets or returns a **Boolean** that represents whether the range will be enclosed within a BDO element.

*expression.ncssBdo*

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example wraps a BDO element around the text within the first P element in the active document.

Dim objPara As IHTMLElement
Dim objSS As IFPStyleState

Set objPara = ActiveDocument.all.tags("p").Item(0)
Set objSS = Application.ActiveDocument.createStyleState

With objSS
    .gatherFromElement objPara
    .ncssBdo = True
    .Apply
End With
ncssBookmark Property

Returns or sets a String that specifies an HTML bookmark for a given element. Bookmarks are used to provide a navigation or reference point within an HTML document.

expression.ncssBookmark

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example specifies new text for the active document, creates a new \texttt{IFPStyleState} object, and then sets a bookmark for the new text.

Sub SetBookMark()

\hspace{1em} Dim objSs As IFPStyleState
\hspace{1em} Dim objDoc As FPHTMLDocument
\hspace{1em} Dim objRng As IHTMLTxtRange
\hspace{1em} Dim objSl As IFPStyleLength

\hspace{1em} Set objDoc = Application.ActiveDocument
\hspace{1em} objDoc.body.innerHTML = "<b>Introduction</b>"
\hspace{1em} Set objSs = objDoc.createStyleState
\hspace{1em} Set objRng = objDoc.body.createTextRange

\hspace{1em} objSs.gather objRng
\hspace{1em} objSs.\texttt{ncssBookmark} = "#Intro"
\hspace{1em} objSs.apply

End Sub
ncssCite Property

Returns or sets a **Boolean** that specifies that the range will be enclosed within a CITE element. Cited text normally appears in italics.

`expression.ncssCite`

*expression* Required. An expression that returns one of the objects in the Applies To list.
Example

The following example creates a range of text and uses the `ncssCite` property to cite the text. The text will now be rendered as bold and italic.

Sub SetCitedText()
'Sets text as cited

    Dim objSs As IFPStyleState
    Dim objLine1 As IHTMLElement
    Dim strHTML As String

    strHTML = "This is sample text. <b>This is bold and will be cited</b>"

    Application.ActiveDocument.body.innerHTML = strHTML

    Set objLine1 = Application.ActiveDocument.all.tags("B").Item(0)
    Set objSs = Application.ActiveDocument.createStyleState
    objSs.gatherFromElement objLine1
    objSs.\texttt{ncssCite} = True
    objSs.apply

End Sub
ncssCode Property

Returns or sets a Boolean that determines if a text range will appear as code. The CODE element is used to denote code in an HTML document.

expression.ncssCode

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example encloses the first instance of bold text within a CODE element. The text will now appear as code.

Sub SetCodeWrap()
    Dim objSs As IFPStyleState
    Dim objLine1 As IHTMLElement
    Dim strHTML As String

    strHTML = "This is sample text. <br> <b>This is bold and will appear as code.</b>

    Application.ActiveDocument.body.innerHTML = strHTML

    Set objLine1 = Application.ActiveDocument.all.tags("B").Item(0)

    Set objSs = Application.ActiveDocument.createStyleState
    objSs.gatherFromElement objLine1

    objSs.ncssCode = True
    objSs.apply

End Sub
ncssDfn Property

Returns or sets a **Boolean** that indicates if the text range will be enclosed in a DFN element, indicating the defining instance of a text range.

`expression.ncssDfn`

*expression* Required. An expression that returns one of the objects in the Applies To list.
Example

The following example uses the `ncssDFN` property to enclose the first instance of italic text in the document within a DFN element.

```
Sub SetNcssDfn()
    Dim objSs As IFPStyleState
    Dim objLine1 As IHTMLElement
    Dim strHTML As String

    strHTML = "This is sample text. <br> <i>This is italicized and w"
    Application.ActiveDocument.body.innerHTML = strHTML

    Set objLine1 = Application.ActiveDocument.all.tags("I").Item(0)
    Set objSs = Application.ActiveDocument.createStyleState
    objSs.gatherFromElement objLine1
    objSs.ncssDfn = True
    objSs.apply

End Sub
```
ncssDynanim Property

The `ncssDynanim` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
ncssEm Property

Returns or sets a **Boolean** that specifies that a text range will be enclosed within EM elements, indicating the emphasis style.

`expression.ncssEm`

**expression**  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example marks all text within the first B element in the active document with the emphasis style.

Sub SetNcssEm()

Dim objSs As IFPStyleState
Dim objLine1 As IHTMLElement
Dim strHTML As String

strHTML = "This is sample text. <br> <b>This is italicized and w"

Application.ActiveDocument.body.innerHTML = strHTML

Set objLine1 = Application.ActiveDocument.all.tags("b").Item(0)

Set objSs = Application.ActiveDocument.createStyleState
objSs.gatherFromElement objLine1

objSs.ncssEm = True
objSs.apply

End Sub
ncssHyperlink Property

Returns or sets a **String** that specifies the URL of the hyperlink. Setting the **ncssHyperLink** property of a text range turns it into a hyperlink.

`expression.ncssHyperlink`

- **expression** Required. An expression that returns one of the objects in the Applies To list.
Example

The following example creates hyperlinks from all text ranges enclosed within CITE elements. The user is prompted for the URL for each citation in the active document.

Sub SetHyperlink()
    Dim objSs As IFPStyleState
    Dim objLine1 As IHTMLElement
    Dim strHTML As String
    Dim strURL As String

    strHTML = "This is sample text. <br> <cite>This is italicized and will appear as a hyperlink.</cite>"
    Application.ActiveDocument.body.innerHTML = strHTML

    For Each objLine1 In Application.ActiveDocument.all.tags("cite")
        Set objSs = Application.ActiveDocument.createStyleState
        objSs.gatherFromElement objLine1
        strURL = InputBox("Enter a URL for the hyperlink:" & _
                         vbCrLf & objLine1.innerText)
        objSs.ncssHyperlink = Trim(strURL)
        objSs.apply
    Next objLine1
End Sub
ncssKbd Property

Returns or sets a Boolean that determines if the text in a given text range will be rendered with a fixed-width font. Read/write.

expression.ncssKbd

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets all text ranges within CITE elements to be rendered with a fixed-width font.

Sub SetWidth()
    'Wraps a text range in kbd tags
    Dim objSs As IFPStyleState
    Dim objLine1 As IHTMLElement
    Dim strHTML As String

    strHTML = "This is sample text. <br> <cite>This is cited and will appear with a fixed-width font.</cite>"
    Application.ActiveDocument.body.innerHTML = strHTML

    For Each objLine1 In Application.ActiveDocument.all.tags("cite")
        Set objSs = Application.ActiveDocument.createStyleState
        objSs.gatherFromElement objLine1
        objSs.ncssKbd = True
        objSs.apply
    Next objLine1

End Sub
**ncssLabelfor Property**

Returns or sets a **String** that specifies a label for a given text range.

`expression.ncssLabelfor`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
**Example**

The following example associates the label "Citation" with all text ranges in the active document within CITE elements.

```vba
Sub SetLabel()
    Dim objSs As IFPStyleState
    Dim objLine1 As IHTMLElement
    Dim strHTML As String

    strHTML = "This is sample text. <br> <cite>This is cited and will within the labeled text.</cite>
    Application.ActiveDocument.body.innerHTML = strHTML

    For Each objLine1 In Application.ActiveDocument.all.tags("cite")
        Set objSs = Application.ActiveDocument.createStyleState
        objSs.gatherFromElement objLine1
        objSs.ncssLabelfor = "Citation"
        objSs.apply
    Next objLine1

End Sub
```
ncssLang Property

Sets or returns an **Integer** that specifies the language format of the current page.

`expression.ncssLang`

*expression* Required. An expression that returns one of the objects in the Applies To list.
ncssNobr Property

Returns a **Boolean** that specifies that an associated range will be rendered without line breaks.
Example

The following example encloses all text ranges currently within H1 elements within NOBR elements. All ranges with the heading 1 style will no longer wrap.

Sub NoBREnding()
'Wraps all H1 tagged ranges in NoBr tags

    Dim objSs As IFPStyleState
    Dim objLine1 As IHTMLElement
    Dim strHTML As String

    strHTML = "<h1>This is a sample heading that will not be wrapped"

    Application.ActiveDocument.body.innerHTML = strHTML

    For Each objLine1 In Application.ActiveDocument.all.tags("h1")
        Set objSs = Application.ActiveDocument.createStyleState
        objSs.gatherFromElement objLine1
        objSs.ncssNobr = True
        objSs.apply
    Next objLine1

End Sub
ncssSamp Property

Returns or sets a **Boolean** that indicates whether a given text range is enclosed within a SAMP element.

`expression.ncssSamp`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example encloses all text ranges currently within CODE elements within SAMP elements. The ranges will now appear with the formatting associated with samples.

Sub NcssSampleTag()
    Dim objSs As IFPStyleState
    Dim objLine1 As IHTMLElement
    Dim strHTML As String

    strHTML = "<code>This is a sample that will appear with the asso"

    Application.ActiveDocument.body.innerHTML = strHTML

    For Each objLine1 In Application.ActiveDocument.all.tags("code")
        Set objSs = Application.ActiveDocument.createStyleState
        objSs.gatherFromElement objLine1
        objSs.ncssSamp = True
        objSs.apply
    Next objLine1
End Sub
ncssSpanclass Property

Sets or returns a **String** that represents the class of the specified **SPAN** element.

*expression.ncssSpanclass*

*expression*  Required. An expression that returns one of the objects in the Applies To list.
ncssSpandir Property

Sets or returns a **String** that specifies the direction of the text within the specified **SPAN** element.

`expression.ncssSpandir`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
ncssStrong Property

Returns a **Boolean** that encloses a given text range within a STRONG element.

*expression*.ncssStrong

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example encloses all `B` elements in the current document within `STRONG` elements.

Sub NcssStrongTag()
    Dim objSs As IFPStyleState
    Dim objLine1 As IHTMLElement
    Dim strHTML As String

    strHTML = "<b>This is a sample that will be wrapped in a STRONG

    Application.ActiveDocument.body.innerHTML = strHTML

    For Each objLine1 In Application.ActiveDocument.all.tags("b")
        Set objSs = Application.ActiveDocument.createStyleState
        objSs.gatherFromElement objLine1
        objSs.**ncssStrong** = True
        objSs.apply
    Next objLine1

End Sub
ncssSub Property

Returns a **Boolean** that specifies if the text will appear as subscript.

*expression.ncssSub*

*expression* Required. An expression that returns one of the objects in the Applies To list.
Example

The following example formats all text within STRONG elements as subscript.

Sub NcssSubTag()
    Dim objSs As IFPStyleState
    Dim objLine1 As IHTMLElement
    Dim strHTML As String

    strHTML = "<strong>This is a sample that will appear as subscript.</strong>"
    Application.ActiveDocument.body.innerHTML = strHTML

    For Each objLine1 In Application.ActiveDocument.all.tags("strong")
        Set objSs = Application.ActiveDocument.createStyleState
        objSs.gatherFromElement objLine1
        objSs.\textit{ncssSub} = True
        objSs.apply
    Next objLine1
End Sub
ncssSup Property

Returns or sets a Boolean that specifies whether text will appear as superscript.

expression.ncssSup

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets all text enclosed in STRONG elements to appear as superscript.

Sub NcssSupTag()
    Dim objSs As IFPStyleState
    Dim objLine1 As IHTMLElement
    Dim strHTML As String

    strHTML = "<strong>This is a sample that will appear as superscr"
    Application.ActiveDocument.body.innerHTML = strHTML

    For Each objLine1 In Application.ActiveDocument.all.tags("strong"
        Set objSs = Application.ActiveDocument.createStyleState
        objSs.gatherFromElement objLine1
        objSs.NcssSup = True
        objSs.apply
    Next objLine1

End Sub
**ncssTableCellspacing Property**

Sets or returns an **Integer** that specifies the cell spacing of a given table.

*expression*.ncssTableCellspacing

*expression* Required. An expression that returns one of the objects in the Applies To list.
ncssTblAlign Property

Returns or sets a String that represents the alignment of a given table.

expression.ncssTblAlign

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the alignment of a given table to right. The table will now appear aligned to the right-hand side of the page.

Sub SetTableAlign()
    Dim objSs As IFPStyleState
    Dim objDoc As FPHTMLDocument
    Dim objRng As IHTMLTxtRange
    Set objDoc = Application.ActiveDocument
    objDoc.body.innerHTML = "<table><tr><td>Cell 1</td><td>Cell 2</td></tr></table>
    Set objSs = objDoc.createStyleState
    Set objRng = objDoc.body.createTextRange
    objSs.gather objRng
    objSs.borderCollapse = True
    objSs.ncssTblAlign = "right"
    objSs.apply
End Sub
**ncssTextValign Property**

Returns or sets a `String` that represents the vertical alignment of the text in a given table.

`expression.ncssTextValign`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the vertical alignment of the text in the given table to bottom. The text will now appear at the bottom of each table cell.

Sub SetTableAlign()
'Sets the text alignment of a given table

    Dim objSs As IFPStyleState
    Dim objDoc As FPHTMLDocument
    Dim objRng As IHTMLTxtRange

    Set objDoc = Application.ActiveDocument
    objDoc.body.innerHTML = "<table height=107><tr><td>Cell 1</td><td>Cell 2</td></tr></table>"
    Set objSs = objDoc.createStyleState
    Set objRng = objDoc.body.createTextRange

    objSs.gather objRng
    objSs.borderCollapse = True
    objSs.ncssTextValign = "bottom"
    objSs.apply

End Sub
**ncssVar Property**

Returns or sets a **Boolean** that determines if the text will be enclosed in a **VAR** element. The **VAR** element is used to specify a programming variable and is typically rendered as italic.

*expression*.**ncssVar**

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example encloses all text currently within **STRONG** elements within **VAR** tags. The text will now appear bold and italic.

Sub NcssVarTag()

    Dim objSs As IFPStyleState
    Dim objLine1 As IHTMLElement
    Dim strHTML As String

    strHTML = "<strong>This is a sample that will appear within a VA

    Application.ActiveDocument.body.innerHTML = strHTML

    For Each objLine1 In Application.ActiveDocument.all.tags("strong
        Set objSs = Application.ActiveDocument.createStyleState
        objSs.gatherFromElement objLine1
        objSs.
ncssVar = True
        objSs.apply
    Next objLine1

End Sub
ncssXime Property

The `ncssXime` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**noHref Property**

Returns or sets a **Boolean** that represents whether an AREA element causes a click event. **False** indicates that the AREA element does not cause a click event. Corresponds to the **nohref** attribute.

`expression.noHref`

*expression* Required. An expression that returns one of the objects in the Applies To list.
Example

The following example removes the `href` attribute for the first AREA element in the first MAP element in the active document and then inserts the `nohref` attribute.

```vbscript
Dim objMap As FPHTMLMapElement
Dim objArea As FPHTMLAreaElement

Set objMap = ActiveDocument.all.tags("map").Item(0)
Set objArea = objMap.areas.Item(0)

objArea.removeAttribute "href"
objArea.noHref = True
```
**noResize Property**

Returns or sets a **Boolean** that represents whether a user can resize a border to a frame when it is displayed in the browser. **False** indicates the frame size is fixed and the user cannot resize it. Corresponds to the **noresize** attribute for the FRAME element.

`expression.noResize`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example disables resizing the first frame in the active document when it is displayed in a browser.

Dim objFrame As IHTMLFrameElement
Set objFrame = ActiveDocument.all.tags("frame").Item(0)
objFrame.noResize = True
noShade Property

Returns or sets a **Boolean** that represents whether the shading line is added to a horizontal line. **True** indicates that the horizontal line is a solid one-color line. **False** indicates that the horizontal line is a traditional two-color line. Default is **False**. Corresponds to the **noshade** attribute for the HR element.

`expression.noShade`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the first horizontal line in the active document to a solid one-color line.

Dim objLine As FPHTMLHRElement
Set objLine = ActiveDocument.all.tags("hr").Item(0)
objLine.noShade = True
**noWrap Property**

Returns or sets a **Boolean** that represents whether text in an element wraps onto subsequent lines. **True** disables word wrapping so that text does not wrap onto following lines. **False** enables word wrapping so that text wraps onto following lines. Default is **False**.

`expression.noWrap`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example specifies that text in the BODY element in the active document will not wrap text onto following lines.

Dim objBody As FPHTMLBody
Set objBody = ActiveDocument.body
objBody.noWrap = True
offscreenBuffering Property

The offscreenBuffering property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
offsetHeight Property

Retrieves the height of the specified object relative to the layout or coordinate parent, as indicated in the offsetParent property.

expression.offsetHeight

expression  Required. An expression that returns one of the objects in the Applies To list.
offsetLeft Property

Retrieves the left edge of the specified object relative to the layout or coordinate parent, as indicated in the offsetParent property.

expression.offsetLeft

expression Required. An expression that returns one of the objects in the Applies To list.
offsetParent Property

Retrurns an IHTMLElement object that represents the parent element that defines the offsetTop and offsetLeft properties.

expression.offsetParent

expression  Required. An expression that returns one of the objects in the Applies To list.
**offsetTop Property**

Retrieves the top edge of the specified object relative to the layout or coordinate parent, as indicated in the `offsetParent` property.

`expression.offsetTop`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
offsetWidth Property

Retrives the width of the specified object relative to the layout or coordinate parent, as indicated in the `offsetParent` property.

`expression.offsetWidth`

`expression` Required. An expression that returns one of the objects in the Applies To list.
offsetY Property

Retrieves the vertical position of the specified object relative to the layout or coordinate parent, as indicated in the offsetY property.

expression.offsetY

expression  Required. An expression that returns one of the objects in the Applies To list.
onabort Property

Sets or returns a **Variant** that represents the value of the **onabort** attribute that is the script when the **onabort** event fires.

`expression.onabort`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the **onabort** attribute for the first IMG element in the active document.

```vbscript
Dim objImage As FPHTMLImg
Set objImage = ActiveDocument.body.all.tags("img").Item(0)
objImage.onabort = "script()"
```
onafterupdate Property

Sets or returns a Variant that represents the value of the onafterupdate attribute that is the name of the script to run when the onafterupdate event fires.

expression.onafterupdate

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `onafterupdate` attribute for the BODY element in the active document.

```javascript
ActiveDocument.body.onafterupdate = "script()"
```
onbeforeunload Property

Returns or sets a Variant that represents the value of the onbeforeunload attribute that is the name of the script to run when the onbeforeunload event fires.

expression.onbeforeunload

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `onbeforeunload` attribute for the BODY element in the active document.

```javascript
ActiveDocument.body.onbeforeunload = "script()
```
onbeforeupdate Property

Returns or sets a **Variant** that represents the value of the **onbeforeupdate** attribute that is the name of the script to run when the **onbeforeupdate** event fires.

`expression.onbeforeupdate`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `onbeforeupdate` attribute for the BODY element in the active document.

`ActiveDocument.body.onbeforeupdate = "script()"`
onblur Property

Sets or returns a Variant that represents the onblur attribute that is the script to run when the onblur event fires.

expression.onblur

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the **onblur** attribute for the first TEXTAREA element in the active document.

Dim objTextbox As FPHTMLTextAreaElement
Set objTextbox = ActiveDocument.body.all.tags("textarea").Item(0)
objTextbox.onblur = "script()"

```
onbounce Property

Sets or returns a **Variant** that represents the **onbounce** attribute that is the script to run when the **onbounce** event fires.

`expression.onbounce`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The `behavior` attribute of the `MARQUEE` element must be set to "alternate" for the `onbounce` event to fire.
Example

The following example inserts a SCRIPT element into the head and replaces the text in the active document with a marquee, and then specifies the behavior and sets the value of the onbounce attribute for the MARQUEE element.

Dim objMarquee As FPHTMLMarqueeElement

With ActiveDocument
   .body.innerHTML = "<marquee id="newmarquee"></marquee>"
   .all.tags("head").Item(0).innerHTML = "<script language=vbscript></script>"
End With

Set objMarquee = ActiveDocument.all.tags("marquee").Item("newmarquee"

With objMarquee
   .behavior = "alternate"
   .direction = "up"
   .loop = 5
   .height = "100%"
   .width = "10%"
   .onbounce = "script()"
   .innerText = "Use Dynamic HTML to create compelling Web sites."
End With
onchange Property

Returns or sets a **Variant** that represents the value of the **onchange** attribute that is the script to run when the **onchange** event fires.

`expression.onchange`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the **onchange** attribute for the first TEXTAREA element in the active document.

Dim objTextbox As FPHTMLTextAreaElement
Set objTextbox = ActiveDocument.all.tags("textarea").Item(0)
objTextbox.onchange = "script()"
onclick Property

Sets or returns a Variant that represents the value of the onclick attribute that is the name of the script to run when the onclick event fires.

expression.onclick

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example specifies the name of the script to run when a user clicks on the active document when it is displayed in a browser.

ActiveDocument.body.onclick = "script()"
ondataavailable Property

Returns or sets a **Variant** that represents the value of the **ondataavailable** attribute that is the name of the script to run when the **ondataavailable** event fires.

```plaintext
expression.ondataavailable
```

*expression* Required. An expression that returns one of the objects in the Applies To list.
**Example**

The following example sets the `ondataavailable` attribute for the BODY element in the active document.

`ActiveDocument.body.ondataavailable = "script()"`
ondatasetchanged Property

Returns or sets a Variant that represents the value of the ondatasetchanged attribute that is the name of the script to run when the ondatasetchanged event fires.

expression.ondatasetchanged

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `ondatasetchanged` attribute for the BODY element in the active document.

`ActiveDocument.body.ondatasetchanged = "script()"`
ondatasetcomplete Property

Returns or sets a Variant that represents the value of the ondatasetcomplete attribute that is the name of the script to run when the ondatasetcomplete event fires.

expression.ondatasetcomplete

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the ondatasetcomplete attribute for the BODY element in the active document.

ActiveDocument.body.ondatasetcomplete = "script()"
ondblclick Property

Returns or sets a Variant that represents the value of the ondblclick attribute that is the name of the script to run when the ondblclick event fires.

expression.ondblclick

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `ondblclick` attribute for the BODY element in the active document.

`ActiveDocument.body.ondblclick = "script()"`
**ondragstart Property**

Returns or sets a **Variant** that represents the value of the **ondragstart** attribute that is the name of the script to run when the **ondragstart** event fires.

`expression.ondragstart`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
**Example**

The following example sets the `ondragstart` attribute for the BODY element in the active document.

```
ActiveDocument.body.ondragstart = "script()"
```
onerror Property

Returns or sets a **Variant** that represents the value of the **onerror** attribute that is the name of the script to run when the **onerror** event fires.

*expression*.onerror

*expression*   Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `onerror` attribute for the first IMG element in the active document.

Dim objImage As FHTMLImg
Set objImage = ActiveDocument.images.Item(0)
objImage.OnError = "script()"
onerrorupdate Property

Returns or sets a Variant that represents the value of the onerrorupdate attribute that is the name of the script to run when the onerrorupdate event fires.

expression.onerrorupdate

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `onerrorupdate` attribute for the BODY element in the active document.

```javascript
ActiveDocument.body.onerrorupdate = "script()"
```
onfilterchange Property

Returns or sets a Variant that represents the value of the onfilterchange attribute that is the name of the script to run when the onfilterchange event fires.

expression.onfilterchange

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `onfilterchange` attribute for the BODY element in the active document.

```
ActiveDocument.body.onfilterchange = "script()"
```
onfinish Property

Returns or sets a Variant that represents the value of the onfinish attribute that is the name of the script to run when the onfinish event fires.

expression.onfinish

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the **onfinish** attribute for the first MARQUEE element in the active document.

```vba
Dim objMarquee As FPHTMLMarqueeElement
Set objMarquee = ActiveDocument.all.tags("marquee").Item(0)
objMarquee.onfinish = "script()"
```
onfocus Property

Returns or sets a Variant that represents the value of the onfocus attribute that is the script to run when the onfocus event fires.

expression.onfocus

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the **onfocus** attribute for the first IMG element in the active document.

```vba
Dim objImage As FPHTMLImg
Set objImage = ActiveDocument.all.tags("img").Item(0)
objImage.onfocus = "script()"
```
onhelp Property

Returns or sets a **Variant** that represents the value of the **onhelp** attribute that is the name of the script to run when the **onhelp** event fires.

*expression.onhelp*

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `onhelp` attribute for the BODY element in the active document.

`ActiveDocument.body.onhelp = "script()"`
onkeydown Property

Returns or sets a **Variant** that represents the value of the **onkeydown** attribute that is the name of the script to run when the **onkeydown** event fires.

**expression.onkeydown**

**expression**  Required. An expression that returns one of the objects in the Applies To list.
Remarks

See also the onkeypress and onkeyup properties.
Example

The following example sets the `onkeydown` attribute for the BODY element in the active document.

`ActiveDocument.body.onkeydown = "script()"`
onkeypress Property

Returns or sets a Variant that represents the value of the onkeypress attribute that is the name of the script to run when the onkeypress event fires.

expression.onkeypress

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

See also the onkeydown and onkeyup properties.
Example

The following example sets the `onkeypress` attribute for the BODY element in the active document.

`ActiveDocument.body.onkeypress = "script()"`
onkeyup Property

Returns or sets a Variant that represents the value of the onkeyup attribute that is the name of the script to run when the onkeyup event fires.

expression.onkeyup

expression Required. An expression that returns one of the objects in the Applies to list.
Remarks

See also the `onkeydown` and `onkeypress` properties.
Example

The following example sets the `onkeyup` attribute for the BODY element in the active document.

ActiveDocument.body.onkeyup = "script()"
onLine Property

The onLine property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**onload Property**

Returns or sets a **Variant** that represents the value of the **onload** attribute that is the name of the script to run when the **onload** event fires.

`expression.onload`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the **onload** attribute for the BODY element in the active document.

```javascript
ActiveDocument.body.onload = "script()"
```
onmousedown Property

Returns or sets a **Variant** that represents the value of the **onmousedown** attribute that is the name of the script to run when the **onmousedown** event fires.

```expression.onmousedown```

**expression**  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `onmousedown` attribute for the BODY element in the active document.

`ActiveDocument.body.onmousedown = "script()"`
onmousemove Property

Returns or sets a Variant that represents the value of the onmousemove attribute that is the name of the script to run when the onmousemove event fires.

expression.onmousemove

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `onmousemove` attribute for the BODY element in the active document.

`ActiveDocument.body.onmousemove = "script()"`
onmouseout Property

Returns or sets a **Variant** that represents the value of the **onmouseout** attribute that is the name of the script to run when the **onmouseout** event fires.

`expression.onmouseout`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the onmouseout attribute for the BODY element in the active document.

ActiveDocument.body.onmouseout = "script()"
onmouseover Property

Returns or sets a Variant that represents the value of the onmouseover attribute that is the name of the script to run when the onmouseover event fires.

expression.onmouseover

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `onmouseout` attribute for the BODY element in the active document.

```
ActiveDocument.body.onmouseout = "script()"
```
onmouseup Property

Returns or sets a Variant that represents the value of the onmouseup attribute that is the name of the script to run when the onmouseup event fires.

expression.onmouseup

expression  Required. An expression that returns one of the objects in the Applies To list.
The following example sets the `onmouseup` attribute for the BODY element in the active document.

`ActiveDocument.body.onmouseup = "script()"`
onreadystatechange Property

Returns or sets a **Variant** that represents the value of the **onreadystatechange** attribute that is the name of the script to run when the **onreadystatechange** event fires.

*expression*.**onreadystatechange**

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `onreadystatechange` attribute for the first LINK element in the active document.

```vba
Dim objLink As FPHTMLLinkElement
Set objLink = ActiveDocument.all.tags("link").Item(0)
objLink.onreadystatechange = "script()"
```
onreset Property

Returns or sets a **Variant** that represents the value of the **onreset** attribute that is the name of the script to run when the **onreset** event fires.

*expression*.onreset

*expression*  Required. An expression that returns one of the objects in the Applies To list.
**Example**

The following example sets the `onreset` attribute for the first FORM element in the active document.

```vba
Dim objForm As FPHTMLFormElement
Set objForm = ActiveDocument.forms.Item(0)
objForm.onreset = "script()"
```
onresize Property

Returns or sets a **Variant** that represents the value of the **onresize** attribute that is the name of the script to run when the **onresize** event fires.

*expression*.onresize

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `onresize` attribute for the BODY element in the active document.

```javascript
ActiveDocument.body.onresize = "script()"
```
onrowenter Property

Returns or sets a Variant that represents the value of the onrowenter attribute that is the name of the script to run when the onrowenter event fires.

expression.onrowenter

expression  Required. An expression that returns one of the objects in the Applies To list.
**Example**

The following example sets the `onrowenter` attribute for the first row in the first table in the active document.

```vba
Dim objTable As FPHTMLTable
Dim objRow As FPHTMLTableRow

Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objRow = objTable.rows.Item(0)

objRow.onrowenter = "script()"
```
onrowexit Property

Returns or sets a Variant that represents the value of the onrowexit attribute that is the name of the script to run when the onrowexit event fires.

expression.onrowexit

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the **onrowexit** attribute for the first row in the first table in the active document.

```vba
Dim objTable As FPHtmlTable
Dim objRow As FPHtmlTableRow

Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objRow = objTable.rows.Item(0)

objRow.onrowexit = "script()"
```
onscroll Property

Returns or sets a Variant that represents the value of the onscroll attribute that is the name of the script to run when the onscroll event fires.

expression.onscroll

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the onscroll attribute for the BODY element in the active document.

ActiveDocument.body.onscroll = "script()"
onselect Property

Returns or sets a **Variant** that represents the value of the **onselect** attribute that is the name of the script to run when the **onselect** event fires.

*expression*.onselect

*expression* Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `onselect` attribute for the BODY element in the active document.

`ActiveDocument.body.onselect = "script()"`
onselectstart Property

Returns or sets a Variant that represents the value of the onselectstart attribute that is the name of the script to run when the onselectstart event fires.

expression.onselectstart

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `onselectstart` attribute for the BODY element in the active document.

`ActiveDocument.body.onselectstart = "script()"`
onstart Property

Returns or sets a Variant that represents the value of the onstart attribute that is the name of the script to run when the onstart event fires.

expression.onstart

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `onstart` attribute for the first MARQUEE element in the active document.

```vba
Dim objMarquee As FHTMLMarqueeElement
Set objMarquee = ActiveDocument.all.tags("marquee").Item(0)
objMarquee.onstart = "script()"
```
onsubmit Property

Returns or sets a **Variant** that represents the value of the **onsubmit** attribute that is the name of the script to run when the **onsubmit** event fires.

*expression*.onsubmit

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `onsubmit` attribute for the first FORM element in the active document.

```vba
Dim objForm As FHTMLFormElement
Set objForm = ActiveDocument.forms.Item(0)
objForm.onsubmit = "script()"
```
onunload Property

Returns or sets a **Variant** that represents the value of the **onunload** attribute that is the name of the script to run when the **onunload** event fires.

*expression*.onunload

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the `onunload` attribute for the BODY element in the active document.

```javascript
ActiveDocument.body.onunload = "script()"
```
opener Property

The opener property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
opsProfile Property

The **opsProfile** property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**Option Property**

The **Option** property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
options Property

As it applies to the SearchInfo object.

Returns an FpSearchOptions that represents the type of search to perform.

FpSearchOptions can be one or more of the following FpSearchOptions constants.

- fpSearchIgnoreWhitespace
- fpSearchMatchAlef
- fpSearchMatchCase
- fpSearchMatchDiacritics
- fpSearchMatchKashida
- fpSearchRegExp
- fpSearchReviseHangul
- fpSearchWholeWord

expression.options

expression Required. An expression that returns a SearchInfo object.

As it applies to all other objects in the Applies To list.

Returns an IHTMLElementCollection collection that represents the OPTION elements for a SELECT element.

expression.options

expression Required. An expression that returns one of the objects in the Applies To list.
Example

As it applies to the `SearchInfo` object.

The following example searches for all occurrences of lower case "the" and capitalizes the first letter.

```vba
Dim objSearch As SearchInfo
Set objSearch = Application.CreateSearchInfo
objSearch.Action = fpSearchReplaceAllText
objSearch.Find = "the"
objSearch.Options = fpSearchMatchCase
objSearch.ReplaceWith = "The"
ActiveDocument.Find objSearch
```

As it applies to the `IHTMLSelectElement` object.

The following example sets the `value` attribute and the displayed list text for the first OPTION element in the first SELECT element in the active document.

```vba
Dim objSelect As IHTMLSelectElement
Dim objOption As IHTMLOptionElement
Set objSelect = ActiveDocument.all.tags("select").Item(0)
Set objOption = objSelect.Options(0)
objOption.Value = "cat"
objOption.Text = "cat"
```
orientation Property

Sets or returns a **String** that represents the direction in which Web Parts will be placed within a zone. Possible values are "vertical" for stacked top-to-bottom and "horizontal" for side-by-side left-to-right.

`expression.orientation`

*expression* Required. An expression that returns one of the objects in the Applies To list.
outerHTML Property

Returns or sets a `String` that represents the HTML for the specified object and all text and HTML that it contains, including the surrounding start and end tags.

`expression.outerHTML`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

See also the `innerHTML`, `innerText`, and `outerText` properties.
Example

The following example replaces the active element with a MARQUEE element and inserts the text and HTML inside the active element into the new MARQUEE element.

```javascript
With ActiveDocument.activeElement
    .outerHTML = "<marquee>" & .innerHTML & "</marquee>"
End With
```
outerText Property

Returns or sets a String that represents the text, without HTML, of the specified object.

expression.outerText

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

See also the `innerHTML`, `innerText`, and `outerHTML` properties.
Example

The following example replaces the active element with a BLOCKQUOTE element and places the existing text for the active element, without any associative HTML, into the new BLOCKQUOTE element.

Note This example will remove any custom formatting, such as bold and italics, and if the active element is the BODY element, it would also remove paragraph-level formatting, such as tables and paragraphs.

With ActiveDocument.activeElement .outerHTML = "<blockquote>" & _ .outerText & "</blockquote>"
End With
overflow Property

Sets or returns a **String** that represents how to manage the content of the object when the content exceeds the height and/or width of the object.

`expression.overflow`

*expression* Required. An expression that returns one of the objects in the Applies To list.
## Remarks

The `overflow` property can be one of the following `String` values.

String that specifies one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>visible</td>
<td>Content is not clipped, and scroll bars are not added.</td>
</tr>
<tr>
<td>scroll</td>
<td>Content is clipped, and scroll bars are added even if the content does not exceed the dimensions of the object.</td>
</tr>
<tr>
<td>hidden</td>
<td>Content that exceeds the dimensions of the object is not shown.</td>
</tr>
<tr>
<td>auto</td>
<td>Content is clipped, and scrolling is added only when necessary.</td>
</tr>
</tbody>
</table>
owningElement Property

Returns an IHTMLElement object that represents the element where a stylesheet is referenced.

expression.owningElement

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example creates a style sheet reference in the active document and then sets the onload attribute that specifies the script to run when the style sheet loads.

```vbscript
Dim objStyleSheet As FPHTMLStyleSheet
Set objStyleSheet = ActiveDocument.createStyleSheet("web.css")
objStyleSheet.owningElement.onload = "script()"
```
padding Property

Returns a String that represents the amount of space between the element and its margin or, if there is a border, between the element and its border.

expression.padding

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **padding** property can be one of the following:

- **length**
  Sets the horizontal or vertical position of the background image to an exact location. Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex).

- **percentage**
  Integer, followed by a percent sign (%). The value is a percentage of the width or height of the object.
paddingBottom Property

Returns a String that represents the amount of space between the bottom edge of the element and its margin or, if there is a border, between the bottom edge of the element and its border.

expression.paddingBottom

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The `paddingBottom` property can be one of the following:

- **length**: Sets the horizontal or vertical position of the background image to an exact location. Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex).

- **percentage**: Integer, followed by a percent sign (%). The value is a percentage of the width or height of the object.
paddingLeft Property

Returns a **String** that represents the amount of space between the left edge of the element and its margin or, if there is a border, between the left edge of the element and its border.

`expression.paddingLeft`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **paddingLeft** property can be one of the following:

| **length** | Sets the horizontal or vertical position of the background image to an exact location. Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). |
| **percentage** | Integer, followed by a percent sign (%). The value is a percentage of the width or height of the object. |
paddingRight Property

Returns a String that represents the amount of space between the right edge of the element and its margin or, if there is a border, between the right edge of the element and its border.

expression.paddingRight

expression    Required. An expression that returns one of the objects in the Applies To list.
**Remarks**

The **paddingRight** property can be one of the following:

<table>
<thead>
<tr>
<th>Length</th>
<th>Sets the horizontal or vertical position of the background image to an exact location. Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percentage</td>
<td>Integer, followed by a percent sign (%). The value is a percentage of the width or height of the object.</td>
</tr>
</tbody>
</table>
paddingTop Property

Returns a String that represents the amount of space between the top edge of the element and its margin or, if there is a border, between the top edge of the element and its border.

expression.paddingTop

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **paddingTop** property can be one of the following:

| **length** | Sets the horizontal or vertical position of the background image to an exact location. Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). |
| **percentage** | Integer, followed by a percent sign (%). The value is a percentage of the width or height of the object. |
pageBreakAfter Property

Sets or returns a String that represents whether a page break occurs after the element when the document is printed. Corresponds to the page-break-after attribute for a cascading style sheet.

expression.pageBreakAfter

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The `pageBreakAfter` property can be one of the following `String` values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>always</td>
<td>Always insert a page break after the element.</td>
</tr>
<tr>
<td>auto</td>
<td>Neither force nor forbid a page break after the element.</td>
</tr>
<tr>
<td>[blank]</td>
<td>Empty string does not insert a page break.</td>
</tr>
</tbody>
</table>
Example

The following example places a page break after the first paragraph in the active document every time the document is printed.

Dim objPara As FPHTMLParaElement
Set objPara = ActiveDocument.all.tags("p").Item(0)
objPara.Style.pageBreakAfter = "always"
pageBreakBefore Property

Sets or returns a `String` that represents whether a page break occurs before the element when the document is printed. Corresponds to the `page-break-before` attribute for a cascading style sheet.

`expression.pageBreakBefore`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Remarks

The `pageBreakBefore` property can be one of the following `String` values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>always</td>
<td>Always insert a page break before the element.</td>
</tr>
<tr>
<td>auto</td>
<td>Neither force nor forbid a page break after the element.</td>
</tr>
<tr>
<td>blank</td>
<td>Empty string does not insert a page break.</td>
</tr>
</tbody>
</table>
Example

The following example places a page break before the first paragraph in the active document every time the document is printed.

Dim objPara As FPHTMLParaElement
Set objPara = ActiveDocument.all.tags("p").Item(0)
objPara.Style.pageBreakBefore = "always"
pageContentElement Property

Returns an IHTMLElement object that represents the HTML element containing the contents of the specified document.

expression.pageContentElement

expression  Required. An expression that returns one of the objects in the Applies to list.
Remarks

Generally, the element returned by using the `pageContentElement` property corresponds to the BODY element. Therefore, you can assign the `pageContentElement` property to an `FPHTMLBody` or `IHTMLBodyElement` object without receiving a Type Mismatch error, as shown in the following code:

```vbnet
Dim objBody As IHTMLBodyElement
Set objBody = ActiveDocument.pageContentElement
```
Example

The following example changes the background color of the active document to aqua.

Dim objBody As FPHTMLBody
Set objBody = ActiveDocument.pageContentElement

objBody.bgColor = "aqua"
**palette Property**

Sets or returns a **String** that represents the palette used for an embedded object in an HTML document.

`expression.palette`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
parent Property

As it applies to the FPHTMLWindow2, FPHTMLWindowProxy, and IHTMLWindow2 objects.

Returns an FPHTMLWindow2 object that represents the parent window.

expression.parent

expression Required. An expression that returns one of the objects in the Applies To list.

As it applies to the FPLayoutRegion and IFPLayoutRegion objects.

Returns an FPLayoutRegion object that represents the parent cell formatting table of a layout region.

expression.parent

expression Required. An expression that returns one of the objects in the Applies To list.
**parentElement Property**

Returns an **IHTMLElement** that represents the element that is one level up in the HTML element hierarchy.

`expression.parentElement`

**expression**  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example accesses the element that is one level up from the BODY element, which is the HTML element.

Dim objElement As IHTMLElement
Set objElement = ActiveDocument.body.parentElement
parentStyleSheet Property

Returns an **IHTMLStyleSheet** object that represents the style sheet that was used for importing style sheets.

*expression*.parentStyleSheet

*expression* Required. An expression that returns an **FPHTMLStyleSheet** object.
**parentTextEdit Property**

Returns an `IHTMLElement` object that represents the parent element in the document hierarchy that can be used to create a text range containing the original element.

`expression.parentTextEdit`

- `expression` Required. An expression that returns one of the objects in the Applies To list.
parentWindow Property

Returns an FPHTMLWindow2 object that represents the parent window of the current document.

expression.parentWindow

expression  Required. An expression that returns one of the objects in the Applies to list.
Example
The following example displays the name of the current document's parent
window if it exists. If the name property is not specified, a message is displayed
to the user.
Sub ReturnParent()
'Returns the parent window of the active document
Dim objApp As FrontPage.Application
Dim objDoc As DispFPHTMLDocument
Dim wdwParent As FPHTMLWindow2
Set objApp = FrontPage.Application
Set objDoc = objApp.ActiveDocument
'Create reference to active document's parent window
Set wdwParent = objDoc.parentWindow
'If parent's name exists
If Not wdwParent.Name = "" Then
'Display names to user
MsgBox objDoc.nameProp & "'s parent window is " _
& wdwParent.Name & "."
Else
'Display message to user
MsgBox objDoc.nameProp & _
"'s parent window does not have a name or does not exist."
End If
End Sub


pathname Property

Returns a String that represents the file name or path for a specified element.

`expression.pathname`

`expression` Required. An expression that returns one of the objects in the Applies To list.
platform Property

The platform property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
plugins Property

Returns an IHTMLElementCollection object that represents the EMBED elements contained in the specified page.

expression.plugins

expression  Required. An expression that returns one of the above objects.
Remarks

The **plugins** property returns the same collection as the **embeds** property.
**Example**

The following example sets an object reference to the first item in the collection of embedded objects in the active document.

```vba
Dim objPlugIn As FPHTMLEmbed
Set objPlugIn = ActiveDocument.plugins.Item(0)
```
pluginspage Property

Returns a String that represents the URL of the plug-in used to view an embedded document.

expression.pluginspage

expression  Required. An expression that returns one of the objects in the Applies To list.
posHeight Property

Sets or returns a **Single** that represents the height of an element in the units specified by the cascading style sheets **height** attribute.

*expression*.posHeight

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the height and width of the cascading style sheet attributes for the first image in the active document.

Dim objImage As FPHTMLImg
Set objImage = ActiveDocument.all.tags("img").Item(0)

With objImage.Style
  .posHeight = 25
  .posWidth = 100
End With
position Property

Returns a **String** that represents the type of positioning used for the specified object.

```javascript
expression.position
```

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value for the **position** property can be one of the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>empty</td>
<td>Object has no special positioning; it follows the layout rules of HTML.</td>
</tr>
<tr>
<td>static</td>
<td>Object has no special positioning; it follows the layout rules of HTML.</td>
</tr>
<tr>
<td>absolute</td>
<td>Object is positioned relative to parent element's position—or to the BODY object if its parent element is not positioned—using the top and left properties.</td>
</tr>
<tr>
<td>relative</td>
<td>Object is positioned according to the normal flow, and then offset by the top and left properties.</td>
</tr>
</tbody>
</table>

For a script to move an element, the element must be defined to have relative or absolute positioning when it is created.

The **position** property is read-only. Use the **setAttribute** method to set the **position** property of the **style** attribute to one of the above values.
Example

The following example sets the \texttt{style} attribute's \texttt{position} property for the \texttt{BODY} element of the active document.

\begin{verbatim}
If ActiveDocument.body.Style.\texttt{Position} = "" Then
    ActiveDocument.body.Style.setAttribute "position", "absolute"
End If
\end{verbatim}
**posLeft Property**

Sets or returns a **Single** that represents the left position of an element in the units specified by the cascading style sheets **left** attribute.

`expression.posLeft`

*expression* Required. An expression that returns one of the objects in the **Applies To** list.
**posTop Property**

Sets or returns a **Single** that represents the top position of an element in the units specified by the cascading style sheets **top** attribute.

`expression.posTop`

*expression* Required. An expression that returns one of the objects in the Applies To list.
posWidth Property

Sets or returns a **Single** that represents the width of an element in the units specified by the cascading style sheets **width** attribute.

`expression.posWidth`

`expression` Required. An expression that returns one of the objects in the Applies To list.
**Example**

The following example sets the height and width of the cascading style sheet attributes for the first image in the active document.

```vba
Dim objImage As FPHTMLImg
Set objImage = ActiveDocument.all.tags("img").Item(0)

With objImage.Style
    .posHeight = 25
    .posWidth = 100
End With
```
prompt Property

Sets or returns a **String** that represents the **prompt** attribute for an ISINDEX element.

`expression.prompt`

*expression* Required. An expression that returns one of the objects in the Applies To list.
**propertyCount Property**

Returns an **Integer** that represents the number of style properties associated with the current element or text range.

`expression.propertyCount`

`expression`  
Required. An expression that returns one of the objects in the Applies To list.
Example

The following example displays the number of properties associated with a given text range.

Sub DisplayPropertyNumber()
    Dim objSs As IFPStyleState
    Dim objDoc As FPHTMLDocument
    Dim objRng As IHTMLTxtRange

    Set objDoc = Application.ActiveDocument

    objDoc.body.innerHTML = "<i><b>Heading 1</b></i>"
    Set objSs = objDoc.createStyleState
    Set objRng = objDoc.body.createTextRange

    objSs.gather objRng
    objSs.setProperty "background-color", vbBlue
    MsgBox "The total number of properties available is: ", _
        & objSs.propertyCount
    objSs.apply

End Sub
protocol Property

Returns a String that represents how a document is accessed. The protocol property returns a substring of the document's URL up to and including the first colon. For example, a page from a disk-based Web site would return a protocol property String of "file:". A page from a server-based Web site might return a protocol property String of "http:".

expression.protocol

expression  Required. An expression that returns one of the objects in the Applies to list.
Remarks

While the Object Browser in the Visual Basic Editor shows the `protocol` property as Read/Write, if you try to set the `protocol` property you will receive an error.
Example

The following example displays the protocol type used to access the active document.

MsgBox ActiveDocument.protocol
**protocolLong Property**

The `protocolLong` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
qualifier Property

The qualifier property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
QueryContents Property

Sets or returns a `String` that is an XML representation of the query.

`expression.QueryContents`

`expression` Required. An expression that returns a `SearchInfo` object.
Example

The following example searches for TD elements in the current selection and adds the **align** attribute with a value of "center."

```vba
Dim objSearch As SearchInfo
Dim objRange As IHTMLTxtRange
Dim blnMatches As Boolean
Dim strQuery As String

strQuery = "<?xml version="1.0"?><fpquery version="1.0">
    <find tag="td"><rule type="insideTag" tag="table"/>
    </find><replace type="setAttribute" attribute="align" value="center"/>
</fpquery>"

Set objRange = ActiveDocument.selection.createRange
Set objSearch = Application.CreateSearchInfo

objSearch.QueryContents = strQuery
blnMatches = Application.ActiveDocument.Find(objSearch, Nothing, objRange)
If blnMatches = True Then objRange.Select
```
readOnly Property

Sets or returns a Boolean that represents whether the content of an object is read-only.

`expression.readOnly`

`expression` Required. An expression that returns one of the objects in the Applies To list.
ReadyState Property

ReadyState property as it applies to the `FPHTMLObjectElement` and `IHTMLObjectElement` objects.

Returns a `Long` that represents the state of the object.

`expression.ReadyState`

`expression` Required. An expression that returns one of the above objects.

ReadyState property as it applies to the `FPHTMLDocument`, `FPHTMLImg`, `FPHTMLInputImage`, `FPHTMLLinkElement`, `FPHTMLScriptElement`, `FPHTMLStyleElement`, `FPHTMLTable`, `IHTMLDocument2`, `IHTMLImgElement`, `IHTMLInputImage`, `IHTMLLinkElement`, `IHTMLScriptElement`, `IHTMLStyleElement`, and `IHTMLTable` objects.

Returns a `String` that represents the state of the specified object.

`expression.ReadyState`

`expression` Required. An expression that returns one of the above objects.
Remarks

The `ReadyState` property is especially useful with long documents that contain a lot of pictures or markup tags that take time to load. Generally, the `ReadyState` property will return the `String"complete."` However, with long documents, you may get the following `String` values:

<table>
<thead>
<tr>
<th>String</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>complete</td>
<td>Object is completely initialized.</td>
</tr>
<tr>
<td>interactive</td>
<td>User can interact with the object even though it is not fully loaded.</td>
</tr>
<tr>
<td>loaded</td>
<td>Object has finished loading its data.</td>
</tr>
<tr>
<td>loading</td>
<td>Object is loading its data.</td>
</tr>
<tr>
<td>uninitialized</td>
<td>Object is not initialized with data.</td>
</tr>
</tbody>
</table>
Example

As it applies to the `FPHTMLDocument` object.

The following example monitors the state of the document in the active page window and allows FrontPage to do other things until the large document has completely loaded.

```vbnet
Do While ActivePageWindow.Document.ReadyState <> "complete"
    DoEvents
Loop
```
reason Property

The `reason` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**recordNumber Property**

The `recordNumber` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
referrer Property

The **referrer** property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
rel Property

Sets or returns a String that represents the relationship(s) between the element and the destination of the link. Corresponds to the rel attribute.

expression.rel

expression Required. An expression that returns one of the objects in the Applies To list.
## Remarks

The **rel** property can be one or more of the following comma-delimited **String** values.

<table>
<thead>
<tr>
<th>Rel</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternate</td>
<td>Substitute version of the file that contains the link.</td>
</tr>
<tr>
<td>Appendix</td>
<td>Page that is an appendix for the set of pages.</td>
</tr>
<tr>
<td>Bookmark</td>
<td>Bookmark.</td>
</tr>
<tr>
<td>Chapter</td>
<td>Page is a chapter for a set of pages.</td>
</tr>
<tr>
<td>Contents</td>
<td>Table of contents document.</td>
</tr>
<tr>
<td>Copyright</td>
<td>Copyright notice for the current page.</td>
</tr>
<tr>
<td>Glossary</td>
<td>Glossary for the current page.</td>
</tr>
<tr>
<td>Help</td>
<td>Help document.</td>
</tr>
<tr>
<td>Index</td>
<td>Index document for the current page.</td>
</tr>
<tr>
<td>Next</td>
<td>Next document in a sequence.</td>
</tr>
<tr>
<td>Offline</td>
<td>Href containing a path to the CDF file to be used for an offline favorite.</td>
</tr>
<tr>
<td>Prev</td>
<td>Previous document in a sequence.</td>
</tr>
<tr>
<td>Section</td>
<td>Page is a section for a set of pages.</td>
</tr>
<tr>
<td>Shortcut Icon</td>
<td>Href containing a path to an icon file to be used for the favorite or link.</td>
</tr>
<tr>
<td>Start</td>
<td>First document of a set.</td>
</tr>
<tr>
<td>Stylesheet</td>
<td>Style sheet.</td>
</tr>
<tr>
<td>Subsection</td>
<td>Page is a subsection for a set of pages.</td>
</tr>
</tbody>
</table>
ReplaceWith Property

Sets or returns a **String** that represents the text to replace the text specified in the **Find** property.

*expression*.ReplaceWith

*expression*  Required. An expression that returns a **SearchInfo** object.
Example

The following example searches for all occurrences of lower case "the" and capitalizes the first letter.

Dim objSearch As SearchInfo
Set objSearch = Application.CreateSearchInfo

objSearch.Action = fpSearchReplaceAllText
objSearch.Find = "the"
objSearch.Options = fpSearchMatchCase
objSearch.ReplaceWith = "The"

ActiveDocument.Find objSearch
**returnValue Property**

Sets or returns a **Variant** that represents the return value from an event.

`expression.returnValue`

**expression** Required. An expression that returns an **IHTMLEventObj** object.
**rev Property**

Sets or returns a **String** that represents the relationship(s) between the hyperlink and the destination page.

`expression.rev`

*expression*     Required. An expression that returns one of the objects in the Applies To list.
Remarks

The `rev` property can be one or more of the following space-delimited values.

- **Alternate**: Substitute version of the file containing the link.
- **Appendix**: Page that is an appendix for the set of pages.
- **Bookmark**: Bookmark.
- **Chapter**: Page is a chapter for a set of pages.
- **Contents**: Table of contents document.
- **Copyright**: Copyright notice for the current page.
- **Glossary**: Glossary for the current page.
- **Help**: Help document.
- **Index**: Index document for the current page.
- **Next**: Next document in a sequence.
- **Prev**: Previous document in a sequence.
- **Section**: Page is a section for a set of pages.
- **Start**: First document of a set.
- **Stylesheet**: Style sheet.
- **Subsection**: Page is a subsection for a set of pages.
Right Property

Sets or returns an IFPStyleLength object that represents the right side of an element.

expression.Right

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example creates a new IFPStyleState object and sets the value and units properties of its associated right IFPStyleLength object.

Sub SetStyleLength()
    Dim objSs As IFPStyleState
    Dim objDoc As FPHTMLDocument
    Dim objRng As IHTMLTxtRange
    Dim objS1 As IFPStyleLength

    Set objDoc = Application.ActiveDocument

    objDoc.body.innerHTML = "<h1>Heading 1</h1>"
    Set objSs = objDoc.createStyleState
    Set objRng = objDoc.body.createTextRange

    objSs.gather objRng

    Set objS1 = objSs.Right

    objS1.units = fpLengthUnitsPT
    objS1.Value = 4
    objSs.Right = objS1
    objSs.Apply
End Sub
rightMargin Property

Sets or returns a String that represents the number of pixels for the right margin of the page.

expression.rightMargin

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the bottom, top, left, and right margins to 50 pixels.

Sub SetMargins()
    Dim objBody As FPHTMLBody
    Set objBody = ActiveDocument.body
    With objBody
        .bottomMargin = "50"
        .leftMargin = "50"
        .rightMargin = "50"
        .topMargin = "50"
    End With
End Sub
rowIndex Property

Returns a `Long` that represents the ordinal position of a TR element in the rows collection for a TABLE element.

`expression.rowIndex`

`expression` Required. An expression that returns one of the objects in the Applies To list.
rows Property

rows property as it applies to the FPHTMLTable, FPHTMLTableSection, IHTMLTable, and IHTMLTableSection objects.

Returns an IHTMLElementCollection object that represents the number of rows in a table.

expression.rows

description Required. An expression that returns one of the above objects.

rows property as it applies to the FPHTMLTextAreaElement and IHTMLTextAreaElement objects.

Sets or returns a Long that represents the height in number of rows of a TEXTAREA element, which corresponds to the value of the rows attribute.

expression.rows

description Required. An expression that returns one of the above objects.

rows property as it applies to the FPHTMLFrameSetSite and IHTMLFrameSetElement objects.

Sets or returns a String that represents the number and height of horizontal frames in a FRAMESET element, which corresponds to the value of the rows attribute.

expression.rows

description Required. An expression that returns one of the above objects.
Remarks

The **String** value for the **rows** property can be one or more of the following comma-delimited values:

- **width**  
  Integer that specifies the frame height, in pixels.

- **percentage**  
  Integer, followed by a %. The value is a percentage of total available height.
  
  Integer that specifies the frame height as a relative value. After allocating pixel or percentage values, the remaining space is divided among all relative-sized frames.

The number of comma-separated items is equal to the number of horizontal frames contained within the FRAMESET, while the value of each item determines the frame height.
Example

As it applies to the **FPHTMLFrameSetSite** object.

The following example replaces the active document's current HTML with a frameset, and then specifies the number of rows or columns contained in each frameset.

Sub CreateFrameSet()
    Dim objFrames As FPHTMLFrameSetSite
    ActiveDocument.body.innerHTML = "<frameset id=""topframe"">" & vbCrLf & _
        vbTab & "<frame id=""top"">" & vbCrLf & _
        vbTab & "<frameset id=""mainframe"">" & vbCrLf & _
        vbTab & vbTab & "<frame id=""left"">" & vbCrLf & _
        vbTab & vbTab & "<frame id=""right"">" & vbCrLf & _
        vbTab & "</frameset>" & vbCrLf & _
    Set objFrames = ActiveDocument.body.all.tags("frameset").Item("t rows") = "75,*"
    Set objFrames = ActiveDocument.body.all.tags("frameset").Item("m cols") = "145,*"
End Sub

As it applies to the **FPHTMLTable** object.

The following example replaces the text in the active document with a table and then changes the background color of the second cell of the second row.

Sub SetCellColor()
    Dim objTable As FPHTMLTable
    ActiveDocument.body.innerHTML = "<table id=""table1"">" & vbCrLf & _
        "<tr><td>r1,c1</td><td>r1,c2</td><td>r1,c3</td></tr>" & vbCrLf & _
        "<tr><td>r2,c1</td><td>r2,c2</td><td>r2,c3</td></tr>" & vbCrLf & _
        "<tr><td>r3,c1</td><td>r3,c2</td><td>r3,c3</td></tr>" & vbCrLf & _
    Set objTable = ActiveDocument.body.all.tags("table").Item("table rows") = 1
    objTable.rows(1).cells(1).bgColor = "#FF00FF"
End Sub
**rowSpan Property**

Returns or sets an **Integer** that represents the number of rows the cell in a TABLE should span.

*expression*.rowSpan

*expression* Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the first cell in the first row of the first table to span two rows. This example assumes that you have at least one TABLE element in the active document.

Dim objCell As FPHTMLTableCell

Set objCell = ActiveDocument.body.all.tags("table")
    .Item(0).rows.Item(0).cells.Item(0)

objCell.rowSpan = 2
rules Property

As it applies to the **FPHTMLStyleSheet** object.

Returns an **IHTMLStyleSheetRulesCollection** collection that represents the style sheet rules for a cascading style sheet.

`expression.rules`

`expression` Required. An expression that returns one of the objects in the Applies To list.

As it applies to the **FPHTMLTable** and **IHTMLTable** objects.

Returns a **String** that represents which dividing lines (inner borders) are displayed in a table. Corresponds to the **rules** attribute.

`expression.rules`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **rules** property can be one of the following **String** values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Displays borders on all rows and columns.</td>
</tr>
<tr>
<td>cols</td>
<td>Displays borders between all table columns. Displays horizontal borders between all THEAD, TBODY, and TFOOT elements; and displays vertical borders between all COLGROUP elements.</td>
</tr>
<tr>
<td>groups</td>
<td>Displays no interior table borders.</td>
</tr>
<tr>
<td>rows</td>
<td>Displays horizontal borders between all table rows.</td>
</tr>
</tbody>
</table>
screen Property

The screen property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**screenX Property**

Returns a Long that represents the horizontal position of the mouse pointer relative to the user's screen.

```
expression.screenX
```

*expression* Required. An expression that returns an **IHTMLEventObj** object.
**screenY Property**

Returns a **Long** that represents the vertical position of the mouse pointer relative to the user's screen.

`expression.screenY`

*expression* Required. An expression that returns an **IHTMLEventObj** object.
scripts Property

Returns an IHTMLElementCollection collection object that represents the collection of all SCRIPT elements in a Web page.

expression.scripts

expression  Required. An expression that returns one of the objects in the Applies to list.
Remarks

You can return the individual **FPHTMLScriptElement** or **IHTMLScriptElement** objects by using the **Item** method and the element identifier, as indicated in the **id** attribute. If there are SCRIPT elements with duplicate **id** attribute values, a collection of the SCRIPT elements with the same **id** attribute values is returned. Collections of SCRIPT elements with the same **id** attribute values must be referenced by ordinal position.
Example

The following example sets the language property for the first SCRIPT element in the active document. This example assumes there is at least one SCRIPT element in the active document.

Dim objScript As FPHTMLScriptElement

Set objScript = ActiveDocument.Scripts.Item(0)
objScript.Language = "javascript"
scroll Property

Sets or returns a String that represents whether the browser displays scroll bars for the element. Corresponds to the scroll attribute.

expression.scroll

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **scroll** property can be one of the following **String** values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>Scroll bars are displayed.</td>
</tr>
<tr>
<td>no</td>
<td>Scroll bars are not displayed.</td>
</tr>
</tbody>
</table>
Example

The following example turns off display of the scroll bars when the active document is displayed in a browser.

Dim objBody As FPHTMLBody
Set objBody = ActiveDocument.body
objBody.scroll = "no"
scrollAmount Property

Sets or returns a **Long** that represents the number of pixels the text scrolls between each subsequent drawing of a MARQUEE element.

```
expression.scrollAmount
```

*expression*  Required. An expression that returns one of the objects in the Applies To list.
scrollDelay Property

Sets or returns a Long that represents the speed, in milliseconds, that the text within a MARQUEE element scrolls.

expression.scrollDelay

expression  Required. An expression that returns one of the objects in the Applies To list.
scrollHeight Property

Returns a Long that represents the scrolling height of an element.

expression.scrollHeight

expression Required. An expression that returns one of the objects in the Applies To list.
scrolling Property

Sets or returns a String that represents whether a frame can be scrolled.

expression.scrolling

expression Required. An expression that returns one of the objects in the Applies To list.
**Remarks**

The **scrolling** property can be one of the following **String** values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>Browser determines whether scroll bars are necessary.</td>
</tr>
<tr>
<td>no</td>
<td>Frame cannot be scrolled.</td>
</tr>
<tr>
<td>yes</td>
<td>Frame can be scrolled</td>
</tr>
</tbody>
</table>
scrollLeft Property

Sets or returns a Long that represents the distance between the left edge of the element and the left-most portion of the content currently visible in the window.

expression.scrollLeft

expression  Required. An expression that returns one of the objects in the Applies To list.
scrollTop Property

Sets or returns a **Long** that represents the distance between the top edge of the element and the top-most portion of the content currently visible in the window.

`expression.scrollTop`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
scrollWidth Property

Returns a Long that represents the scrolling width of an element.

expression.scrollWidth

expression Required. An expression that returns one of the objects in the Applies To list.
search Property

Sets or returns a **String** that represents the value of an **href** attribute that follows a question mark.

`expression.search`

**expression**  Required. An expression that returns one of the objects in the Applies To list.
sectionRowIndex Property

Returns a Long that represents the ordinal position of a TBODY, THEAD, TFOOT, or TR element within a TABLE element.

expression.sectionRowIndex

expression  Required. An expression that returns one of the objects in the Applies To list.
security Property

The security property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**selected Property**

Sets or returns a **Boolean** that represents whether the option in the list box is the default option selected.

**expression.selected**

**expression**  Required. An expression that returns one of the objects in the Applies To list.
## Remarks

The selected property can be one of the following String values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>Item is not selected as the default option.</td>
</tr>
<tr>
<td>true</td>
<td>Item is selected as the default option.</td>
</tr>
</tbody>
</table>
selectedIndex Property

The `selectedIndex` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**selection Property**

Returns an **IHTMLSelectionObject** object that represents the currently selected text in Microsoft FrontPage.

*expression.selection*

*expression*  Required. An expression that returns one of the objects in the Applies to list.
Example

The following example creates a text range with the current selection and then collapses the range to paste a new empty paragraph into the document.

```
Dim objRange As IHTMLTxtRange
Set objRange = ActiveDocument.Selection.createRange
objRange.collapse False
objRange.pasteHTML "<p>&nbsp;</p>"
```
selectorText Property

Sets or returns a **String** that represents the selection string for a specified style sheet rule.

*expression*.selectorText

*expression* Required. An expression that returns an **IHTMLStyleSheetRule** object.
Example

This example retrieves the selection string from the first style sheet rule in the first style sheet attached to the active document. This example assumes that you have at least one style sheet attached to the active document.

Private Sub GetSelectorText()
    Dim objRule As IHTMLStyleSheetRule

    Set objRule = ActiveDocument.styleSheets(0).rules(0)

    ActiveDocument.body.insertAdjacentHTML "beforeend", _
"<P>SELECTOR TEXT: " & objRule.selectorText & "</P>"
End Sub
**self Property**

Returns an **IHTMLWindow2** object that represents a reference to the current window or frame.

`expression.self`

*expression* Required. An expression that returns one of the objects in the Applies To list.
shape Property

Sets or returns a String that represents the shape of a hyperlink in an AREA element within a MAP element.

expression.shape

expression    Required. An expression that returns one of the objects in the Applies To list.
Remarks

The String value of the shape property, which corresponds to the value of the shape attribute, can be one of the following:

- circ or circle: The shape of the hyperlink is a circle.
- poly or polygon: The shape of the hyperlink is a polygon.
- rect or rectangle: The shape of the hyperlink is a rectangle.
Example

The following example replaces the text in the active document with a graphic, and then inserts an image map with one AREA element and specifies its shape, coordinates, and hyperlink URL.

Sub SetImageCoords()
    Dim objImage As FPHTMLImg
    Dim objArea As FPHTMLAreaElement

    ActiveDocument.body.innerHTML = _
        "<img src="graphics/chelan.jpg" id="chelan"">" & vbCrLf

    Set objImage = ActiveDocument.all.tags("img").Item("chelan")
    objImage.useMap = "#ImageMap"

    ActiveDocument.body.insertAdjacentHTML where:="beforeend", _
        HTML:="<map name="ImageMap">" & vbCrLf & _
        "<area id="Area1">" & vbCrLf & "</map>" & vbCrLf
    Set objArea = ActiveDocument.body.all.tags("area").Item("Area1")

    With objArea
        .Shape = "rect"
        .coords = "5, 16, 151, 286"
        .href = "http://www.microsoft.com"
    End With

End Sub
shiftKey Property

Returns a **Boolean** that represents whether the SHIFT key on the keyboard is pressed. **True** if the SHIFT key is pressed. **False** if it is not.

`expression.shiftKey`

`expression` Required. An expression that returns an [IHTMLEventObj](#) object.
Example

The following example displays a message every time the user clicks on the active document telling whether the user is pressing the SHIFT key. This example must be placed in a class module, and it assumes that you have declared in the General Declarations section of a class module an `FPHTMLDocument` object variable called "doc" using the `WithEvents` keyword.

Private Function objDoc_onclick() As Boolean
    Dim objEvent As IHTMLEventObj
    Set objEvent = objWindow.event

    Select Case objEvent.shiftKey
        Case False
            MsgBox "You are not pressing your SHIFT key."
        Case True
            MsgBox "You are pressing your SHIFT key."
    End Select
End Function
ShowHostPropertyPages Property

The ShowHostPropertyPages property is a member of the FrontPageHostedControl interfaces. For more information about how to use these when creating a Microsoft ActiveX control to use in Microsoft FrontPage, see the FrontPage Software Development Kit (SDK) on the Microsoft Developer Network (MSDN) Web site.
**size Property**

Sets or returns a **String** that represents either the size of the font (when applied to the FONT element) or the height of a horizontal line (when applied to the HR element).

*expression*.

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example inserts a horizontal line at the beginning of the active document and sets the height of the line.

Dim objLine As FPHTMLHRElement

ActiveDocument.body.insertAdjacentHTML "afterbegin", "<HR>
Set objLine = ActiveDocument.all.tags("hr").Item(0)

objLine.Size = "5"
SkipOnQuery Property

Returns a **Boolean** that specifies how to handle files that contain regions that have not been specified using the **SetBodyMapping** or **SetHeadMapping** methods and do not have an equivalently named region in the new template. In this circumstance, Microsoft FrontPage normally displays a dialog box asking for user input. **True** skips the file. **False** terminates the process and returns an error. The default value is **False**.

```
expression.SkipOnQuery
```

**expression** Required. An expression that returns a **DynamicTemplateState** object.
Example

The following example updates or attaches the specified Dynamic Web Template to all files in the active Web site and creates a log.

Dim objTemplate As DynamicTemplateState
Dim objFile As WebFile
Dim strLog As String
Dim strLogFile As String

Set objTemplate = Application.CreateDynamicTemplateState

objTemplate.SetBodyMapping ",", "template.dwt", "(body)", "A1"
objTemplate.SetHeadMapping ",", "template.dwt", "", "metadata"
objTemplate.SkipOnQuery = True

For Each objFile In ActiveWeb.AllFiles
    If objFile.Extension = "htm" Then
        If objFile.DynamicTemplate <> "" Then
            objFile.UpdateDynamicTemplate objTemplate, strLog

            strLogFile = strLogFile & strLog & vbCrLf
            strLog = ""
        End If
    Next

sourceIndex Property

Returns a **Long** that represents the ordinal position of the element, in source order, as the element appears in the **IHTMLElementCollection** collection accessed by using the **all** property.

```expression.sourceIndex```

*expression*. **sourceIndex**

**expression** Required. An expression that returns one of the objects in the Applies To list.
span Property

Sets or returns a Long that represents the number of columns a table cell spans. Corresponds to the span attribute for a COL element.

expression.span

expression  Required. An expression that returns one of the objects in the Applies To list.
**src Property**

Returns or sets a **String** that represents the source file for a specified object. Corresponds with the `src` attribute.

*expression.src*

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example specifies a source file for the first IMG element in the active document and removes the **height** and **width** attributes since the new image may have a different height and width.

```vbscript
Set objImage = ActiveDocument.images.Item(0)
With objImage
    .src = "graphics/chelan.jpg"
    .removeAttribute "height"
    .removeAttribute "width"
End With
```
srcElement Property

Returns an IHTMLElement that represents the element that raised an event.

expression.srcElement

expression Required. An expression that returns an IHTMLEventObj object.
Remarks

See also the `fromElement` and `toElement` properties.
Example

The following example changes the background color of the element that the users clicks. This example must be placed in a class module, and it assumes that you have declared in the General Declarations section of a class module an **FPHTMLDocument** object variable called "objDoc" using the **WithEvents** keyword.

```vba
Private Function objDoc_onclick() As Boolean
    Dim objEvent As IHTMLEventObj
    Dim objElement As IHTMLElement

    Set objEvent = objWindow.event
    Set objElement = objEvent.srcElement

    objElement.Style.backgroundColor = "aqua"

End Function
```
srcFilter Property

The srcFilter property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**start Property**

As it applies to the **FPHTMLOListElement** and **IHTMLOListElement** objects.

Sets or returns a **Long** that represents the starting number for an ordered list. Corresponds to the **start** attribute.

**expression.start**

**expression**  Required. An expression that returns one of the objects in the Applies To list.

As it applies to all other objects in the Applies To list.

Sets or returns a **String** that represents when a video clip file should begin playing.

**expression.start**

**expression**  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **start** property can be one of the following **String** values.

- **fileopen**: Video begins as soon as it finishes loading.
- **mouseover**: Video begins when the user moves the mouse over the animation
**status Property**

The `status` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
style Property

Returns an FPHTMLStyle object that represents the inline style properties for an element. Corresponds to the style attribute.

expression.style

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

Inline styles are style sheets for a single instance of an element and are specified in the element's opening tag. Inline styles are defined using the style attribute, and the data for the attribute is specified using the cascading style sheet language. Use the style property to access and set cascading style sheet properties. For more information about cascading style sheets, see CSS Reference.
Example

The following example sets the background color of the active document.

```
ActiveDocument.body.Style.backgroundColor = "blue"
```

The following example sets the background color for the first P element in the document.

```
ActiveDocument.all.tags("p").Item(0)._Style.cssText = "background-color:yellow"
```

The following example sets the margin for the active document.

```
ActiveDocument.body.Style.margin = "10px"
```
styleFloat Property

Returns or sets a String that represents the side of the page on which the object is positioned. Text will flow around the object on the opposite side.

expression.styleFloat

expression Required. An expression that returns one of the objects in the Applies To list.
## Remarks

The **String** value for the **styleFloat** property can be one of the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>Object displays where it appears in the text.</td>
</tr>
<tr>
<td>left</td>
<td>Object is positioned on the left side of the page. Text flows to the right of the object.</td>
</tr>
<tr>
<td>right</td>
<td>Object is positioned on the right side of the page. Text flows to the left of the object.</td>
</tr>
</tbody>
</table>
Example

The following example specifies that the first image in the active document is positioned on the right side of the page and any text that follows flows around the object on the left side.

ActiveDocument.images.Item(0).Style.styleFloat = "right"
**styleSheet Property**

Returns an IHTMLStyleSheet object that represents the cascading style sheet associated with a LINK or STYLE element.

```plaintext
expression.styleSheet
```

*expression* Required. An expression that returns one of the objects in the Applies To list.
**styleSheets Property**

Returns an **FPHTMLStyleSheetsCollection** collection that represents the collection of all style sheets in a document.

`expression.styleSheets`

`expression` Required. An expression that returns one of the objects in the Applies to list.
Remarks

Use the `imports` property to access imported style sheets, specified by using a `STYLE` element with an `@Import` statement.
Example
The following example creates a reference to the current document's style sheets
collection and displays the title (if it exists) for each stylesheet in the collection.
Sub ReturnStyleSheets()
'Returns a collection of a style sheets in the document
Dim objApp As FrontPage.Application
Dim objStyleSheets As FPHTMLStyleSheetsCollection
Dim objStyleSheet As FPHTMLStyleSheet
Set objApp = FrontPage.Application
Set objStyleSheets = objApp.ActiveDocument.styleSheets
'For each style sheet in the document
For Each objStyleSheet In objStyleSheets
'if it has a title, display it
If objStyle.Title <> "" Then
MsgBox objStyleSheet.Title
End If
Next objStyleSheet
End Sub


systemLanguage Property

The `systemLanguage` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**tabIndex Property**

Sets or returns an **Integer** that represents the position within the tab selection order for an element.

`expression.tabIndex`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
tagName Property

Returns a String that represents the name of an element.

expression.tagName

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the background color of the active element if the active element is a P element.

Dim objElement As IHTMLElement
Dim objPara As FPHTMLParaElement
Dim objStyle As IFPStyleState

Set objElement = ActiveDocument.activeElement

If LCase(objElement.tagName) = LCase("p") Then
    Set objPara = objElement
    Set objStyle = ActiveDocument.createStyleState

    objStyle.gatherFromElement objPara

    objPara.Style.backgroundColor = "blue"
End If
target Property

Sets or returns a **String** that represents the window or frame in which to display the contents of a hyperlink. Corresponds to the **target** attribute.

`expression.target`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
**Remarks**

The **target** property can be one of the following **String** values

- `_blank` Loads the linked document into a new blank window. This window is not named.
- `_parent` Loads the linked document into the immediate parent of the document the link is in.
- `_search` Loads the linked document into the browser's search pane. Available in Internet Explorer 5 or later.
- `_self` Loads the linked document into the same window the link was clicked in (the active window).
- `_top` Loads the linked document into the topmost window.
- `name` Loads the linked document into the named frame or window.
Example

The following example loads the page referenced in the first hyperlink in the active document in a new browser window.

Dim objLink As FPHTMLAnchorElement
Set objLink = ActiveDocument.all.tags("a").Item(0)
objLink.target = "_blank"
tBodies Property

Returns an IHTMLElementCollection collection that represents the collection of TBODY elements inside a TABLE element.

expression.tBodies

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

Use the **FPHTMLTableSection** object to return a single TBODY element.
Example

The following example sets the background color for the first TBODY element inside the first TABLE element in the active document.

Dim objTable As FPHTMLTable
Dim objTBodies As IHTMLElementCollection
Dim objBody As FPHTMLTableSection

Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objTBodies = objTable.tBodies
Set objBody = objTBodies.Item(0)

objBody.bgColor = "red"
templateRegions Property

Returns an IHTMLElementCollection collection that represents the Dynamic Web Template regions in a document.

`expression.templateRegions`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Example

The following example accesses the first region for the attached Dynamic Web Template and sets the \texttt{id} attribute equal to the value of the region's name.

\begin{verbatim}
Dim objRegions As IHTMLElementCollection
Dim objRegion As FHTMLTemplateRegionElement

Set objRegions = ActiveDocument.\texttt{templateRegions}
Set objRegion = objRegions.Item(0)

objRegion.Id = objRegion.Name
\end{verbatim}
text Property

Returns or sets a **Variant** (in the case of the **FPHTMLBody** and **IHTMLBodyElement** objects) or a **String** (in the case of all other objects in the Applies To list) that represents the color of text in the specified object.

*expression*.text

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

Color values can be specified as color names or Web-safe red-green-blue (RGB) color values. For more information on available color names and RGB color values, see the HTML Color Table.
Example

The following example sets the color of the text in the active document to blue.

ActiveDocument.body.Text = "blue"
**textAlign Property**

Returns and sets a *String* that represents how text is aligned in an element. Corresponds to the *text-align* property for a cascading style sheet.

`expression.textAlign`

*expression*    Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value for the **textAlign** property can be one of the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>left</td>
<td>Text is aligned to the left. Default.</td>
</tr>
<tr>
<td>right</td>
<td>Text is aligned to the right.</td>
</tr>
<tr>
<td>center</td>
<td>Text is centered.</td>
</tr>
<tr>
<td>justify</td>
<td>Text is justified.</td>
</tr>
</tbody>
</table>
Example

The following example sets the text in the body to align on the right side of the page.

ActiveDocument.body.style.textAlign = "right"

The following example centers text in the first paragraph in the active document.

ActiveDocument.all.tags("p").Item(0).Style.textAlign = "center"
textDecoration Property

Returns a String that represents whether the text in the object has strike-through, overline, or underline text decorations.

`expression.textDecoration`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Remarks

The `textDecoration` property can be one of the following `String` values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>Text has no decoration.</td>
</tr>
<tr>
<td>underline</td>
<td>Text is underlined.</td>
</tr>
<tr>
<td>overline</td>
<td>Text has a line over it.</td>
</tr>
<tr>
<td>line-through</td>
<td>Text has a line drawn through it.</td>
</tr>
<tr>
<td>blink</td>
<td>Not implemented.</td>
</tr>
</tbody>
</table>
Example

The following example causes all text in the active document to be underlined.

Dim objStyle As FPHTMLStyle

Set objStyle = ActiveDocument.body.Style
objStyle.textDecoration = "underline"
textDecorationBlink Property

Sets or returns a **Boolean** that represents whether text is set to blink.

**Note** The blink text decoration has not been implemented in the Internet Explorer browser.

`expression.textDecorationBlink`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
**textDecorationLineThrough Property**

Sets or returns a **Boolean** that represents whether text is displayed with a line through it.

`expression.textDecorationLineThrough`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the text in the body of the active document to display with a line through it.

Dim objStyle As FPHTMLStyle
Set objStyle = ActiveDocument.body.Style
objStyle.textDecorationLineThrough = True
textDecorationNone Property

Sets or returns a **Boolean** that represents whether to remove text decoration formatting from text.

`expression.textDecorationNone`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Example

The following example removes text decoration from all text in the body of the active document. Note that this applies only to text that was previously affected by the `textdecoration` property of the BODY element's `style` attribute.

Dim objStyle As FPHTMLStyle

Set objStyle = ActiveDocument.body.Style
objStyle.textDecorationNone = True
**textDecorationOverline Property**

Sets or returns a **Boolean** that represents whether there is a line above the text in an element.

*expression*.textDecorationOverline

*expression* Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the text in the body of the active document to display with a line over it.

Dim objStyle As FPHTMLStyle

Set objStyle = ActiveDocument.body.Style
objStyle.textDecorationOverline = True
**textDecorationUnderline Property**

Sets or returns a **Boolean** that represents whether text is underlined.

`expression.textDecorationUnderline`

`expression`   Required. An expression that returns one of the objects in the Applies To list.
Example

The following example underlines the text in the body of the active document.

Dim objStyle As FPHTMLStyle

Set objStyle = ActiveDocument.body.Style
objStyle.textDecorationUnderline = True
textIndent Property

Sets or returns the indentation of text in an element.

`expression.textIndent`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **textIndent** property can be one of the following:

- **length**
  Sets the horizontal or vertical position of the background image to an exact location. Floating-point number, followed by an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex).

- **percentage**
  Integer, followed by a percent sign (%). The value is a percentage of the width or height of the object.
Example

The following example indents the left edge of text in the active document to 25 pixels from the left edge of the document.

Dim objStyle As FPHTMLStyle
Set objStyle = ActiveDocument.body.Style
objStyle.textIndent = "25px"
textTransform Property

Sets or returns a String that represents whether text is displayed by using uppercase, lowercase, or mixed case.

expression\_textTransform

expression  Required. An expression that returns one of the objects in the Applies To list.
## Remarks

The **textTransform** property can be one of the following **String** values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>Text is not transformed.</td>
</tr>
<tr>
<td>capitalize</td>
<td>Transforms the first character of each word to uppercase.</td>
</tr>
<tr>
<td>uppercase</td>
<td>Transforms all the characters to uppercase.</td>
</tr>
<tr>
<td>lowercase</td>
<td>Transforms all the characters to lowercase.</td>
</tr>
</tbody>
</table>
Example

The following example displays the text in the body of the active document in uppercase characters.

Dim objStyle As FPHTMLStyle

Set objStyle = ActiveDocument.body.Style
objStyle.textTransform = "uppercase"
**tFoot Property**

Returns an [IHTMLTableSection](#) object that represents the TFOOT element in a TABLE element.

`expression.tFoot`

*expression*   Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the background color for the TFOOT element in the first table in the active document.

Dim objTable As FPHTMLTable
Dim objTFoot As FPHTMLTableSection

Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objTFoot = objTable.tFoot

objTFoot.bgColor = "blue"
tHead Property

Returns an IHTMLTableSection object that represents the THEAD element in a TABLE element.

expression.tHead

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the background color for the THEAD element in the first table in the active document.

Dim objTable As FPHTMLTable
Dim objTHead As FPHTMLTableSection

Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objTHead = objTable.tHead

objTHead.bgColor = "blue"
title Property

Sets or returns a String that represents the value of the TITLE element or a ScreenTip for an element.

expression.title

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the value of the TITLE element in the active document.

```
ActiveDocument.Title = "Home Page"
```

The following example sets the ScreenTip text for the first paragraph in the active document.

```
Dim objPara As FPHTMLParaElement

Set objPara = ActiveDocument.all.tags("p").Item(0)
objPara.Title = "tooltip text"
```
toElement Property

Returns an IHTMLElement object that represents the element to which the user is moving the mouse pointer during an onmouseover or onmouseout event.

expression.toElement

expression  Required. An expression that returns an IHTMLEventObj object.
Remarks

See also the `fromElement` and `srcElement` properties
Example

The following example follows the mouse pointer and selects the element over which the mouse pointer is positioned. This example must be placed in a class module, and it assumes that you have declared in the General Declarations section of a class module an `FPHTMLDocument` object variable called "objDoc" using the ` WithEvents` keyword.

Private Sub objDoc_onmouseover()
    Dim objEvent As IHTMLEventObj
    Dim objRange As IHTMLTxtRange
    Dim objElement As IHTMLElement

    Set objEvent = objWindow.event
    Set objElement = objEvent.toElement
    Set objRange = objDoc.body.createTextRange

    objRange.moveToElementText objElement
    objRange.Select
End Sub
top Property

As it applies to the FPHTMLWindow2 object.

Returns an FPHTMLWindows2 object that represents the topmost ancestor window, which is its own parent.

expression.top

expression Required. An expression that returns one of the objects in the Applies To list.

As it applies to the FPHTMLStyle object.

Sets or returns a Variant that represents the position of an element relative to the top of the next positioned element in the document hierarchy. Corresponds to the top attribute of a cascading style sheet.

expression.top

expression Required. An expression that returns one of the objects in the Applies To list.

As it applies to the IFPStyleState object.

Returns an IFPStyleLength object that represents the measurements associated with the position of an element.

expression.top

expression Required. An expression that returns one of the objects in the Applies To list.
**topMargin Property**

Sets or returns a **String** that represents the number of pixels for the top margin of the page.

`expression.topMargin`

*expression* Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the bottom, top, left, and right margins to 50 pixels.

Sub SetMargins()
    Dim objBody As FPHTMLBody
    Set objBody = ActiveDocument.body

    With objBody
        .bottomMargin = "50"
        .leftMargin = "50"
        .rightMargin = "50"
        .topMargin = "50"
    End With
End Sub
trueSpeed Property

Sets or returns a **Boolean** that specifies whether scrolling of text in a MARQUEE element is calculated using the **scrollDelay** and **scrollAmount** properties and the actual time elapsed from the last clock tick. Corresponds to the **trueSpeed** attribute of a MARQUEE element.

`expression.trueSpeed`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The `trueSpeed` property can be one of the following `String` values.

- **false**: Bases scroll movement on 60-millisecond ticks of the clock. This ignores `scrollDelay` attribute values under 60, and the text in the MARQUEE element advances the amount of the `scrollAmount` attribute each 60 milliseconds. For example, if the `scrollDelay` attribute is 6 and the `scrollAmount` attribute is 10, the text in the MARQUEE element advances 10 pixels every 60 milliseconds.

- **true**: Advances the pixel value of the `scrollAmount` attribute by the number of milliseconds set for the `scrollDelay` attribute. For example, the text in the MARQUEE element would scroll 10 pixels for every 6 milliseconds if the `scrollDelay` attribute is 6, the `scrollAmount` attribute is 10, and the `trueSpeed` attribute is "true".
As it applies to the **FPHTMLButtonElement** and **IHTMLButtonElement** objects

Returns a **String** that represents the classification and default behavior of the button.

*expression*.type

*expression*   Required. An expression that returns one of the above objects.
Remarks

The **type** property may be one of the following **String** values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>button</td>
<td>Creates a Command button.</td>
</tr>
<tr>
<td>reset</td>
<td>Creates a Reset button. If it's in a form, this button resets the fields in the form to their initial values.</td>
</tr>
<tr>
<td>submit</td>
<td>Creates a Submit button. If it's in a form, this button submits the form.</td>
</tr>
</tbody>
</table>

As it applies to the [FPHTMLInputElement](https://www.w3.org/TR/html5/input.html#input-control-attributes), [FPHTMLInputFileElement](https://www.w3.org/TR/html5/input.html#input-control-attributes), [FPHTMLInputHiddenElement](https://www.w3.org/TR/html5/input.html#input-control-attributes), [FPHTMLInputImage](https://www.w3.org/TR/html5/input.html#input-control-attributes), [FPHTMLInputTextElement](https://www.w3.org/TR/html5/input.html#input-control-attributes), [FPHTMLSelectElement](https://www.w3.org/TR/html5/input.html#input-control-attributes), [FPHTMLTextAreaElement](https://www.w3.org/TR/html5/input.html#input-control-attributes), [IHTMLInputButtonElement](https://www.w3.org/TR/html5/input.html#input-control-attributes), [IHTMLInputFileElement](https://www.w3.org/TR/html5/input.html#input-control-attributes), [IHTMLInputHiddenElement](https://www.w3.org/TR/html5/input.html#input-control-attributes), [IHTMLInputImage](https://www.w3.org/TR/html5/input.html#input-control-attributes), [IHTMLInputTextElement](https://www.w3.org/TR/html5/input.html#input-control-attributes), [IHTMLSelectElement](https://www.w3.org/TR/html5/input.html#input-control-attributes), and [IHTMLTextAreaElement](https://www.w3.org/TR/html5/input.html#input-control-attributes) objects

Returns a **String** that represents the type of control.

`expression.type`

`expression` Required. An expression that returns one of the above objects.
### Remarks

The *type* property may be one of the following *String* values:

<table>
<thead>
<tr>
<th>Object</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPHTMLInputElement and IHTMLInputElement</td>
<td>button</td>
<td>Creates a button control.</td>
</tr>
<tr>
<td>FPHTMLInputElement and IHTMLInputElement</td>
<td>checkbox</td>
<td>Creates a checkbox control.</td>
</tr>
<tr>
<td>FPHTMLInputElement and IHTMLInputElement</td>
<td>file</td>
<td>Creates a file upload object.</td>
</tr>
<tr>
<td>FPHTMLInputElement and IHTMLInputElement</td>
<td>hidden</td>
<td>Creates a control hidden from a user.</td>
</tr>
<tr>
<td>FPHTMLInputElement and IHTMLInputElement</td>
<td>image</td>
<td>Creates an image control that can be clicked, causing a form to be immediately submitted.</td>
</tr>
<tr>
<td>FPHTMLInputElement and IHTMLInputElement</td>
<td>password</td>
<td>Creates a control similar to the text control, except that text is not displayed as the user enters it.</td>
</tr>
<tr>
<td>FPHTMLInputElement and IHTMLInputElement</td>
<td>radio</td>
<td>Creates radio buttons used for mutually exclusive sets of values.</td>
</tr>
<tr>
<td>FPHTMLInputElement and IHTMLInputElement</td>
<td>reset</td>
<td>Creates a button that resets a form's controls to their specified initial values.</td>
</tr>
<tr>
<td>FPHTMLInputElement and IHTMLInputElement</td>
<td>submit</td>
<td>Creates a button that submits a form.</td>
</tr>
<tr>
<td>FPHTMLInputElement and IHTMLInputElement</td>
<td>text</td>
<td>Creates a single-line text-entry control.</td>
</tr>
<tr>
<td>FPHTMLSelectElement and IHTMLSelectElement</td>
<td>select-multiple</td>
<td>Creates a list box from which a user can select multiple items.</td>
</tr>
</tbody>
</table>
FPHTMLSelectElement and IHTMLSelectElement

select-one

Creates a list box from which a user can select one item.

FPHTMLTextAreaElement and IHTMLTextAreaElement

textarea

Creates a multiple-line text-entry control.

As it applies to the FPHTMLLIElement, FPHTMLOListElement, FPHTMLULListElement, IHTMLLIElement, IHTMLOListElement, and IHTMLULListElement objects

Sets or returns a String that represents the style of bullet for a numbered or bulleted list.

expression.type

expression Required. An expression that returns one of the above objects.
Remarks

The **type** property may be one of the following **String** values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Displays Arabic numerals for items in a numbered list.</td>
</tr>
<tr>
<td>a</td>
<td>Displays lowercase letters for items in a numbered list.</td>
</tr>
<tr>
<td>A</td>
<td>Displays uppercase letters for items in a numbered list.</td>
</tr>
<tr>
<td>i</td>
<td>Displays lowercase Roman numerals for items in a numbered list.</td>
</tr>
<tr>
<td>I</td>
<td>Displays uppercase Roman numerals for items in a numbered list.</td>
</tr>
<tr>
<td>disc</td>
<td>Displays a solid disc for items in a bulleted list.</td>
</tr>
<tr>
<td>circle</td>
<td>Displays a hollow circle for items in a bulleted list.</td>
</tr>
<tr>
<td>square</td>
<td>Displays a solid square for items in a bulleted list.</td>
</tr>
</tbody>
</table>

As it applies to the **FPHTMLObjectElement** and **IHTMLObjectElement** objects

Sets or returns a **String** that represents the MIME type of an OBJECT element.

*expression.type*

*expression*  Required. An expression that returns one of the above objects.
Remarks

MIME is an abbreviation for Multipurpose Internet Mail Extension and is a set of enhancements used with Simple Mail Transfer Protocol (SMTP). Used together, MIME and SMTP allow Internet messages to include a mixture of audio, images, video, and text components, and accommodate a variety of international character sets.

As it applies to the FPHTMLScriptElement and IHTMLScriptElement objects

Sets or returns a String that represents the language associated with a script.

expression.type

expression  Required. An expression that returns one of the above objects.
Remarks

The type property may be one of the following String values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text/ecmascript</td>
<td>ECMAScript.</td>
</tr>
<tr>
<td>text/Jscript</td>
<td>JScript (compatible with ECMA 262 language specification).</td>
</tr>
<tr>
<td>text/javascript</td>
<td>JScript.</td>
</tr>
<tr>
<td>text/vbs</td>
<td>VBScript.</td>
</tr>
<tr>
<td>text/vbscript</td>
<td>VBScript (same as text/vbs).</td>
</tr>
<tr>
<td>text/xml</td>
<td>XML.</td>
</tr>
</tbody>
</table>

As it applies to the FPHTMLStyle and IHTMLStyleElement objects

Sets or returns a String that represents the Cascading Style Sheet (CSS) language in which the style sheet is written.

expression.type

eexpression    Required. An expression that returns one of the above objects.

As it applies to the FPHTMLStyleSheet object

Returns a String that represents the Cascading Style Sheet (CSS) language in which a style sheet is written.

eexpression.type

eexpression    Required. An expression that returns one of the above objects.

As it applies to the IHTMLEventObj object

Returns a String that represents the name of the event.

eexpression.type
expression Required. An expression that returns one of the above objects.

As it applies to the IHTMLSelectionObject object

Returns a String that represents the type of selection.

dot expression.type

dot expression Required. An expression that returns an IHTMLSelectionObject object.
## Remarks

The **type** property can be one of the following **String** values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>No selection/insertion point.</td>
</tr>
<tr>
<td>text</td>
<td>Text selection.</td>
</tr>
<tr>
<td>control</td>
<td>Control selection.</td>
</tr>
</tbody>
</table>
units Property

As it applies to the **FPHTML\Embed** and **IHTML\Embed\Element** objects.

Sets or returns a **String** that represents the measurement type for the height and width of an embedded object.

```
espression.units
```

```
espression  Required. An expression that returns one of the objects in the Applies To list.
```
Remarks

The **units** property can be one of the following **String** values.

### **px**
Height and width are measured in pixels.

### **em**
Height and width are measured relative to the height and width of the element's font.

As it applies to the **IFPStyleLength** object.

Returns an **FpLengthUnits** constant that represents the measurement type for the property settings in the specified **IFPStyleLength** object.

**FpLengthUnits** can be one of the following **FpLengthUnits** constants.

- **fpLengthUnitsAtom**
- **fpLengthUnitsCM**
- **fpLengthUnitsEM**
- **fpLengthUnitsEX**
- **fpLengthUnitsHalfPoints**
- **fpLengthUnitsHTMLFontUnits**
- **fpLengthUnitsIN**
- **fpLengthUnitsInvalid**
- **fpLengthUnitsMM**
- **fpLengthUnitsNone**
- **fpLengthUnitsNoValue**
- **fpLengthUnitsPC**
- **fpLengthUnitsPercent**
- **fpLengthUnitsPT**
- **fpLengthUnitsPX**
- **fpLengthUnitsRelative**

*expression.units*
**expression**  Required. An expression that returns one of the objects in the Applies To list.
**updateInterval Property**

The `updateInterval` property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
URL Property

Returns a **String** that represents the Uniform Resource Locator (URL) for the document.

`expression.URL`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Remarks

The Microsoft Visual Basic Object Browser specifies the URL property as read/write. However, setting this property will raise a run-time error.
**urn Property**

Sets or returns a **String** that represents a Uniform Resource Name (URN) for a target document in a hyperlink.

*expression.urn*

*expression*  Required. An expression that returns one of the objects in the Applies To list.
**useMap Property**

Sets or returns a **String** that represents the name, often with a bookmark extension (`#name`), to use as a client-side image map for an image.

`expression.useMap`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Example

The following example replaces the text in the active document with a graphic, and then inserts an image map with one AREA element and specifies its share, coordinates, and hyperlink URL.

Sub SetImageCoords()
    Dim objImage As FPHTMLImg
    Dim objArea As FPHTMLAreaElement

    ActiveDocument.body.innerHTML = _
        "<img src="graphics/chelan.jpg" id="chelan"">" & vbCrLf

    Set objImage = ActiveDocument.all.tags("img").Item("chelan")
    objImage.useMap = "#ImageMap"

    ActiveDocument.body.insertAdjacentHTML where:="beforeend", _
        HTML:="<map name="ImageMap">" & vbCrLf & _
        "<area id="Area1">" & vbCrLf & "</map>" & vbCrLf
    Set objArea = ActiveDocument.body.all.tags("area").Item("Area1")

    With objArea
        .Shape = "rect"
        .coords = "5, 16, 151, 286"
        .href = "http://www.microsoft.com"
    End With

End Sub
userAgent Property

The userAgent property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
userLanguage Property

The userLanguage property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
userProfile Property

The userProfile property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
vAlign Property

Sets or returns a **String** that represents the vertical position of a table caption or the vertical position of text within an element.

`expression.vAlign`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The `vAlign` property for the `FPHTMLTableCaption` and `ITHMLTableCaption` objects can be one of the following `String` values.

- **top** Places the caption at the top of the table. (Default.)
- **bottom** Places the caption at the bottom of the table.

The `vAlign` property for all other objects in the Applies To list.

- **middle** Aligns the text in the middle of the object. (Default.)
- **center** Aligns the text in the middle of the object.
- **baseline** Aligns the base line of the first line of text with the base lines in adjacent objects.
- **bottom** Aligns the text at the bottom of the object.
- **top** Aligns the text at the top of the object.
Example

The following example places the caption in the first table in the active document below the table.

Dim objTable As FPHTMLTable
Dim objCaption As FPHTMLTableCaption

Set objTable = ActiveDocument.all.tags("table").Item(0)
Set objCaption = objTable.Caption

objCaption.vAlign = "bottom"
value Property

As it applies to the FPHTMLLIElement and IHTMLLIElement objects.

Sets or returns a Long that represents the count of an item in an ordered list. Corresponds to the value attribute of an LI element.

expression.value

expression Required. An expression that returns one of the above objects.

As it applies to the FPHTMLTextAreaElement and IHTMLTextAreaElement objects.

Sets or returns a String that represents the text displayed a TEXTAREA element. Corresponds to the value attribute of a TEXTAREA element.

expression.value

expression Required. An expression that returns one of the above objects.

As it applies to all other objects in the Applies To list.

Sets or returns a String that represents value of the the value attribute for an INPUT or OPTION element.

expression.value

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

The purpose of the **value** property for the INPUT element depends on the value of the **type** property. The following list describes the purpose of the **value** property for each of the different types of INPUT elements.

- **checkbox**: The selected value. The control submits this value only if the user has selected the control. Otherwise, the control submits no value.
- **file**: The value, a file name, typed by the user into the control. Unlike other controls, this value is read-only.
- **hidden**: The control submits this value when the form is submitted.
- **password**: The default value. The control displays this value when it is first created and when the user clicks the reset button.
- **radio**: The button label. If not set, the label defaults to "Reset".
- **reset**: The selected value. The control submits this value only if the user has selected the control. Otherwise, the control submits no value.
- **submit**: The button label. If not set, the label defaults to "Submit Query."
- **text**: The default value. The control displays this value when it is first created and when the user clicks the reset button.
verticalAlign Property

As it applies to the FPHTMLStyle object.

Sets or returns a Variant that represents the vertical positioning of an element or the text within an element.

\textit{expression}.verticalAlign

\textit{expression} Required. An expression that returns one of the objects in the Applies To list.
Remarks

The `verticalAlign` property can be one of the following `String` values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>Aligns the contents of an element to the base line.</td>
</tr>
<tr>
<td>sub</td>
<td>Vertically aligns the text to subscript.</td>
</tr>
<tr>
<td>super</td>
<td>Vertically aligns the text to superscript.</td>
</tr>
<tr>
<td>top</td>
<td>Vertically aligns the contents of an element to the top of the object.</td>
</tr>
<tr>
<td>middle</td>
<td>Vertically aligns the contents of an element to the middle of the object.</td>
</tr>
<tr>
<td>bottom</td>
<td>Vertically aligns the contents of an element to the bottom of the object.</td>
</tr>
<tr>
<td>text-top</td>
<td>Vertically aligns the text of an element to the top of the object.</td>
</tr>
<tr>
<td>text-bottom</td>
<td>Vertically aligns the text of of an element to the bottom of the object.</td>
</tr>
</tbody>
</table>

As it applies to the `IFPStyleState` object.

Returns an `IFPStyleLength` object that represents the measurement settings for the vertical alignment of an element.

`expression垂直Align`  

`expression` Required. An expression that returns one of the objects in the Applies To list.
**ViewMode Property (Page Object Model)**

Returns a **Long** that represents whether the page has been edited in Code view. A 1 indicates that the page has not been edited; a 2 indicates that the page has been edited.

`expression.ViewMode`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Remarks

In previous versions of Microsoft FrontPage, it was impossible to run Microsoft Visual Basic for Applications (VBA) code that modified the HTML in a page — for example, adding elements or changing attributes — while viewing the HTML in FrontPage. Doing so displayed a "Permission Denied" error.

With Microsoft Office FrontPage 2003, you can now make changes to the HTML in a page by using VBA code while a page is displayed in the FrontPage Code view. However, the FrontPage Page Object model becomes unavailable when edits are made in Code view. Use the ViewMode property to determine if changes have been made to a Web page in Code view. If the ViewMode property returns a 2, use the parseCodeChanges method to reparse the page without having to change the view and to eliminate receiving the "Permission Denied" error.
Example

The following example reparses the code in the active document if the document has been changed in Code view.

Sub ReparseCodeInPage()
    If ActiveDocument.ViewMode = 2 Then
        ActiveDocument.parseCodeChanges
    End If
End Sub
visibility Property

Sets or returns a String that represents whether or not the contents of an element is visible.

expression.visibility

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **visibility** property can be one of the following **String** values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inherit</td>
<td>Inherits the visibility of the nearest parent element.</td>
</tr>
<tr>
<td>visible</td>
<td>Text in the element is visible.</td>
</tr>
<tr>
<td>hidden</td>
<td>Text in the element is hidden.</td>
</tr>
</tbody>
</table>
Example

The following example hides the first table in the active document so that it doesn't render when the page is displayed in the browser.

Dim objTable As FPHTMLTable
Set objTable = ActiveDocument.all.tags("table").Item(0)

objTable.Style.**visibility** = "hidden"
vLink Property

Returns or sets a **Variant** that specifies the color of all visited links in the current document. Corresponds to the **vlink** attribute of the BODY element.

*expression*.vLink

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

Values can be specified as color names or Web-safe red-green-blue (RGB) color values. For more information on available color names and RGB color values, see the HTML Color Table.

Both the vLink and vlinkColor properties correspond the vLink attribute of a BODY element. The difference between the two properties is that the vLink property is a member of the body object and the vlinkColor property is a member of the document object.
Example

The following example sets the color of all visited links in the active document to green.

ActiveDocument.body.vLink = "#00FF00"
vlinkColor Property

Returns or sets a Variant that specifies the color of all visited links in the current
document. Corresponds to the vlink attribute of the BODY element.

`expression.vlinkColor`

`expression`  Required. An expression that returns one of the objects in the
Applies to list.
Remarks

Values can be specified as color names or Web-safe red-green-blue (RGB) color values. For more information on available color names and RGB color values, see the [HTML Color Table](#).

Both the `vLink` and `vlinkColor` properties correspond the `vLink` attribute of a BODY element. The difference between the two properties is that the `vLink` property is a member of the body object and the `vlinkColor` property is a member of the document object.
Example

The following example sets the color of all visited links in the active document to red.

ActiveDocument.vlinkColor = "#FF0000"
**volume Property**

Sets or returns a **Variant** that represents the volume for a background sound for a Web page. Corresponds to the **volume** attribute for a BGSOUND element.

```
expression.volume
```

**expression** Required. An expression that returns one of the objects in the Applies To list.
Example

The following example sets the properties for the background sound for the active document.

Dim objHead As IHTMLElement
Dim objSound As FPHTMLBGsound

Set objSound = ActiveDocument.all.tags("bgsound").Item(0)

With objSound
    .src = "sounds/applause.wav"
    .balance = 0
    .loop = 1
    .volume = 0
End With
**vrml Property**

Sets or returns a **String** that represents the URL of the virtual reality modeling language (VRML) world to be displayed in the window.

`expression.vrml`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
vspace Property

Sets or returns a Long that represents the vertical margin for an element.

expression.vspace

expression  Required. An expression that returns one of the objects in the Applies To list.
web Property

Returns a WebEx object that represents the Web site in which the specified page exists.

expression.web

expression Required. An expression that returns one of the objects in the Applies To list.
**webbots Property**

Returns an `IHTMLElementCollection` collection that represents the Microsoft FrontPage components inserted into the document.

```javascript
expression.webbots
```

*expression* Required. An expression that returns one of the objects in the Applies to list.
Example

The following example creates an object variable for the FrontPage components in the active document, and then uses the `Item` method to access the component named "calendar". This example assumes you have at least one FrontPage component in the active document with an `id` attribute of "calendar".

Dim objFPComponents As IHTMLElementCollection
Dim objCalendar As IHTMLElement

Set objFPComponents = ActiveDocument.webbots
Set objCalendar = objFPComponents.Item("calendar")
webParts Property

Returns an IHTMLElementCollection collection that represents the Web Parts in a document.

expression.webParts

expression  Required. An expression that returns one of the objects in the Applies To list.
Example

The following example accesses the first Web Part in the active document.

Dim objPart As FPHTMLWebPartElement
Set objPart = ActiveDocument.webParts.Item(0)
webPartZones Property

Returns an IHTMLElementCollection collection that represents the Web Part zones in a document.

expression.webPartZones

expression Required. An expression that returns one of the objects in the Applies To list.
Example

The following example accesses the first Web Part zone in the active document.

Dim objZone As IHTMLWebPartZoneElement

Set objZone = ActiveDocument.webPartZones.Item(0)
**whiteSpace Property**

Sets or returns a *String* that represents the *whiteSpace* attribute for inline HTML styles.

*expression*.**whiteSpace**

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **String** value for the **whiteSpace** property can be one of the following:

- **normal**: Default. Lines of text break automatically. Content wraps to the next line if it exceeds the width of the object.
- **nowrap**: Line breaks are suppressed. Content does not wrap to the next line. Line breaks and other white space are preserved. This possible value is supported in Internet Explorer 6 and later when the !DOCTYPE declaration specifies standards-compliant mode. When the !DOCTYPE declaration does not specify standards-compliant mode, you can retrieve this value, but it does not affect rendering— it functions like the normal value.
Example

The following example sets the **whiteSpace** attribute for the body of the active document to not wrap text.

Sub SetWhiteSpaceToNoWrap()
    ActiveDocument.body.Style.whiteSpace = "nowrap"
End Sub
width Property

As it applies to the FPHTMLImg, FPHTMLInputImage, IHTMLImgElement, and IHTMLInputImage objects.

Sets or returns a **Long** that represents the width of an image. Corresponds to the **width** attribute for an IMG element or an INPUT element of type "image".

`expression.width`

`expression` Required. An expression that returns one of the objects in the Applies To list.

As it applies to the IFPStyleState object.

Returns an **IFPStyleLength** object that represents the measurement settings for the width attribute for the element specified in the **IFPStyleState** object.

`expression.width`

`expression` Required. An expression that returns one of the objects in the Applies To list.

As it applies to all other objects in the Applies To list.

Sets or returns a **Variant** that represents the value of the **width** attribute of an element. Corresponds to the **width** attribute.

`expression.width`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Window Property

Returns an FPHTMLWindow2 object that represents the browser window.

expression.Window

expression Required. An expression that returns one of the objects in the Applies To list.
The **wordSpacing** property is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**wrap Property**

Sets or returns a **String** that represents how to handle word wrapping within a TEXTAREA element.

`expression.wrap`

`expression` Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **wrap** property can be one of the following **String** values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>soft</td>
<td>Wraps words and submits it without carriage returns and line feeds.</td>
</tr>
<tr>
<td>hard</td>
<td>Wraps words and submits it with soft returns and line feeds.</td>
</tr>
<tr>
<td>off</td>
<td>Disables word wrapping and displays the lines exactly as the user types them.</td>
</tr>
</tbody>
</table>
x Property

Returns a Long that represents the horizontal position of the mouse pointer relative to the parent element.

expression.x

expression Required. An expression that returns an IHTMLEventObj object.
y Property

Returns a Long that represents the vertical position of the mouse pointer relative to the parent element.

expression.y

expression  Required. An expression that returns an IHTMLEventObj object.
zIndex Property

As it applies to the FPHTMLStyle object.

Sets or returns a Variant that represents the stacking order of positioned objects. Corresponds to the z-index property of a cascading style sheet.

expression.zIndex

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

The `zIndex` property can be one of the following values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>String that specifies the stacking order of the positioned objects based on the order in which the elements appear in the HTML source.</td>
</tr>
<tr>
<td>order</td>
<td>Integer that specifies the position of the element in the stacking order.</td>
</tr>
</tbody>
</table>

As it applies to the `IFPStyleState` object.

Returns an `IFPStyleLength` object that represents the measurement settings for the `z-index` property of a cascading style sheet for the element specified in the `IFPStyleState` object.

`expression.zIndex`

`expression` Required. An expression that returns one of the objects in the Applies To list.
**zoneId Property**

Sets or returns a **String** that represents the identifier of the a Web Part zone.

*expression.zoneId*

**expression**  Required. An expression that returns one of the objects in the Applies To list.
**zoneTitle Property**

Sets or returns a **String** that represents a friendly name that is displayed when designing a Web Part Page inside of the browser.

The **zoneTitle** property can also specify a localization string. To specify one of the localized strings, use "loc:string" as your title, where "string" is one of the localization values within the Microsoft .Net Framework.

**Note** The **zoneTitle** property applies only to pages within a Web site based on Microsoft Windows SharePoint Services.

```csharp
expression.zoneTitle
```

*expression* Required. An expression that returns one of the objects in the Applies To list.
onabort Event

The onabort event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onafterupdate Event

The onafterupdate event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onbeforeunload Event

Occurs before unloading a page in Microsoft FrontPage.

expression.onbeforeunload

expression  Required. An expression that returns one of the objects in the Applies To list, which has been declared using the WithEvents keyword in a class module.
Example

The following example displays a message if the active document has been modified since it was last saved, and then saves the document at the user's request. This example must be placed in a class module, and it assumes that you have declared in the General Declarations section of a class module an **FPHTMLWindow2** object variable called "objWindow" using the ** WithEvents** keyword.

```vba
Private Sub objWindow_onbeforeunload()
    Dim intResponse As Integer
    Dim objDoc As FPHTMLDocument

    Set objDoc = objWindow.Document

    If objDoc.IsDirty = True Then
        intResponse = MsgBox("Do you want to save " & _
        "the page before you close it?", vbYesNo)

        If intResponse = vbYes Then
            objDoc.Save False
        End If
    End If
End Sub
```
onbeforeupdate Event

The onbeforeupdate event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onblur Event

The onblur event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onbounce Event

The **onbounce** event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onchange Event

The `onchange` event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onclick Event

Occurs when a user clicks on a page or an element within a page in Design view in Microsoft FrontPage.

expression.onclick

expression    Required. An expression that returns one of the objects in the Applies To list, which has been declared using the WithEvents keyword in a class module.
Example

The following example selects the active element when the user clicks on the document. This example must be placed in a class module, and it assumes that you have declared in the General Declarations section of a class module an `FPHTMLDocument` object variable called "objDoc" using the ` WithEvents` keyword.

Private Function objDoc_onclick() As Boolean
    Dim objRange As IHTMLTxtRange
    Dim objElement As IHTMLElement
    Dim objEvent As IHTMLEventObj

    Set objEvent = objDoc.parentWindow.event
    Set objElement = objEvent.srcElement
    Set objRange = ActiveDocument.body.createTextRange

    objRange.moveToElementText objElement
    objRange.Select
End Function
The `ondataavailable` event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
ondatasetchanged Event

The ondatasetchanged event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onatasetcomplete Event

The onatasetcomplete event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
**ondblclick Event**

Occurs when a user double-clicks on an element within a page in Design view in Microsoft FrontPage.

`expression.ondblclick`

`expression`  Required. An expression that returns one of the objects in the Applies To list, which has been declared using the **WithEvents** keyword in a [class module](#).
Example

The following example changes the background color of the source element when the user double-clicks in the document. This example must be placed in a class module, and it assumes that you have declared in the General Declarations section of a class module an `FPHTMLDocument` object variable called "objDoc" using the ` WithEvents` keyword.

```vba
Private Function objDoc_ondblclick() As Boolean
    Dim objElement As IHTMLElement
    Dim objEvent As IHTMLEventObj

    Set objEvent = objDoc.parentWindow.event
    Set objElement = objEvent.srcElement

    objElement.Style.backgroundColor = "yellow"
End Function
```
**ondragstart Event**

Occurs when the user starts to drag a text selection or selected object on a page in Design view in Microsoft FrontPage.

*expression.ondragstart*

*expression*  Required. An expression that returns one of the objects in the Applies To list, which has been declared using the ` WithEvents` keyword in a *class module*. 
onerror Event

The onerror event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onerrorupdate Event

The onerrorupdate event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onfilterchange Event

The **onfilterchange** event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onfinish Event

The onfinish event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onfocus Event

The **onfocus** event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onhelp Event

The onhelp event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onkeydown Event

Occurs when a user presses a key on the keyboard when a page is displayed in Design view in Microsoft FrontPage.

expression.onkeydown

expression  Required. An expression that returns one of the objects in the Applies To list, which has been declared using the `WithEvents` keyword in a class module.
Example

The following example sets the **letterspacing** attribute for the active element when the user presses the **Ctrl-Alt-X** key combination. This example must be placed in a class module, and it assumes that you have declared in the General Declarations section of a class module an **FPHTMLBody** object variable called "objBody" using the ** WithEvents** keyword.

Private Sub objBody_onkeydown()
    Dim objEvent As IHTMLEventObj
    Dim objElement As IHTMLElement

    Set objEvent = objBody.Document.parentWindow.event

    If objEvent.ctrlKey = True And objEvent.altKey = True And _
        objEvent.keyCode = 88 Then
        objEvent.srcElement.Style.letterSpacing = "10px"
    End If
End Sub
onkeypress Event

The onkeypress event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onkeyup Event

Occurs when a user releases a key when a page is displayed in Design view in Microsoft FrontPage.

expression.onkeyup

expression Required. An expression that returns one of the objects in the Applies To list, which has been declared using the WithEvents keyword in a class module.
onload Event

The `onload` event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onmousedown Event

Occurs when a user clicks an element with either mouse button when a page is displayed in Design view in Microsoft FrontPage.

expression.onmousedown

expression  Required. An expression that returns one of the objects in the Applies To list, which has been declared using the WithEvents keyword in a class module.
onmousemove Event

Occurs when the user moves the mouse pointer over an element when a page is displayed in Design view in Microsoft FrontPage.

expression.onmousemove

expression    Required. An expression that returns one of the objects in the Applies To list, which has been declared using the WithEvents keyword in a class module.
**Example**

The following example follows the mouse pointer and selects the element over which the mouse pointer is positioned. This example must be placed in a class module, and it assumes that you have declared in the General Declarations section of a class module an `FPHTMLDocument` object variable called "objDoc" using the ` WithEvents ` keyword.

```vba
Private Sub objDoc_onmousemove()
    Dim objEvent As IHTMLEventObj
    Dim objRange As IHTMLTxtRange
    Dim objElement As IHTMLElement

    Set objEvent = objDoc.parentWindow.event
    Set objElement = objEvent.srcElement
    Set objRange = ActiveDocument.body.createTextRange

    objRange.moveToElementText objElement
    objRange.Select
End Sub
```
onmouseout Event

Occurs when a user moves the mouse pointer outside the boundaries of an element when a page is displayed in Design view in Microsoft FrontPage.

expression.onmouseout

expression Required. An expression that returns one of the objects in the Applies To list, which has been declared using the WithEvents keyword in a class module.
Example

The following example trails after the mouse pointer and selects the element from which the mouse pointer is moving away. This example must be placed in a class module, and it assumes that you have declared in the General Declarations section of a class module an FPHTMLBody object variable called "objBody" using the WithEvents keyword.

Private Sub objBody_onmouseout()
    Dim objEvent As IHTMLEventObj
    Dim objRange As IHTMLTxtRange
    Dim objElement As IHTMLElement

    Set objEvent = objBody.Document.parentWindow.event
    Set objElement = objEvent.srcElement
    Set objRange = ActiveDocument.body.createTextRange

    objRange.moveToElementText objElement
    objRange.Select
End Sub
onmouseover Event

Occurs when a user drags the mouse pointer over an element when a page is displayed in Design view in Microsoft FrontPage.

expression.onmouseover

expression Required. An expression that returns one of the objects in the Applies To list, which has been declared using the WithEvents keyword in a class module.
Example

The following example follows the mouse pointer and selects the element over which the mouse pointer is positioned. This example must be placed in a class module, and it assumes that you have declared in the General Declarations section of a class module an **FPHTMLDocument** object variable called "objDoc" using the ** WithEvents** keyword.

Private Sub objDoc_onmouseover()
    Dim objEvent As IHTMLEventObj
    Dim objRange As IHTMLTxtRange
    Dim objElement As IHTMLElement

    Set objEvent = objDoc.parentWindow.event
    Set objElement = objEvent.toElement
    Set objRange = ActiveDocument.body.createTextRange

    objRange.moveToElementText objElement
    objRange.Select
End Sub
onmouseup Event

Occurs when a user releases a mouse button while the mouse is over a page or an element when a page is displayed in Design view in Microsoft FrontPage.

expression.onmouseup

expression  Required. An expression that returns one of the objects in the Applies To list, which has been declared using the WithEvents keyword in a class module.
**Example**

The following example changes the background color of the active element when the user releases the mouse button. This example must be placed in a class module, and it assumes that you have declared in the General Declarations section of a class module an **FPHTMLBody** object variable called "objBody" using the ** WithEvents** keyword.

Private Sub objBody_onmouseup()
    Dim objElement As IHTMLElement
    Dim objEvent As IHTMLEventObj
    Set objEvent = objBody.Document.parentWindow.event
    Set objElement = objEvent.srcElement
    objElement.Style.backgroundColor = "yellow"
End Sub
onreadystatechange Event

The onreadystatechange event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onreset Event

The onreset event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onresize Event

Occurs before a window or page is resized in the Microsoft FrontPage Page window.

expression.onresize

expression  Required. An expression that returns one of the objects in the Applies To list, which has been declared using the WithEvents keyword in a class module.
onrowenter Event

The onrowenter event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onrowexit Event

The onrowexit event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
Show All
onscroll Event

Occurs when a user repositions the scroll box in either the horizontal or vertical scroll bar in a frame or window.

expression.onscroll

expression  Required. An expression that returns one of the objects in the Applies To list, which has been declared using the WithEvents keyword in a class module.
The *onselect* event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onselectstart Event

Occurs when a user starts selecting text or elements when a page is displayed in Design view in Microsoft FrontPage.

**expression.onselectstart**

**expression**  Required. An expression that returns one of the objects in the Applies To list, which has been declared using the **WithEvents** keyword in a [class module](http://example.com).
onstart Event

The **onstart** event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onsubmit Event

The onsubmit event is not supported in Microsoft FrontPage and is therefore unavailable for programmatic access.
onunload Event

Occurs after a user closes a page window in Microsoft FrontPage.

expression.onunload

expression  Required. An expression that returns one of the objects in the Applies To list.
**Example**

The following example displays information about the document that the user is closing.

```vbnet
Private Sub objWindow_onunload()
    Dim strLogInfo As String

    strLogInfo = "User: " & Application.UserName & vbCrLf & _
                "Page Name: " & objWindow.Document.Url & vbCrLf & _
                "Modified On: " & objWindow.Document.fileModifiedDate

End Sub
```
# FrontPage Page Object Model

## Enumerated Constants

This topic provides a list of all enumerated constants in the Microsoft FrontPage Page object model.

### FpCheckOutOption

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpCheckOut</td>
<td>1</td>
</tr>
<tr>
<td>fpCheckOutCancel</td>
<td>3</td>
</tr>
<tr>
<td>fpCheckOutPromptUser</td>
<td>0</td>
</tr>
<tr>
<td>fpCheckOutReadOnly</td>
<td>2</td>
</tr>
</tbody>
</table>

### FpChoiceFieldFormat

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpChoiceFieldCheckboxes</td>
<td>2</td>
</tr>
<tr>
<td>fpChoiceFieldDropdown</td>
<td>0</td>
</tr>
<tr>
<td>fpChoiceFieldRadioButtons</td>
<td>1</td>
</tr>
</tbody>
</table>

### FpConnectionSpeed

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpConnect1000K</td>
<td>9</td>
</tr>
<tr>
<td>fpConnect128K</td>
<td>3</td>
</tr>
<tr>
<td>fpConnect144</td>
<td>0</td>
</tr>
<tr>
<td>fpConnect1500K</td>
<td>4</td>
</tr>
<tr>
<td>fpConnect256K</td>
<td>6</td>
</tr>
<tr>
<td>fpConnect288</td>
<td>1</td>
</tr>
<tr>
<td>fpConnect384K</td>
<td>7</td>
</tr>
<tr>
<td>fpConnect512K</td>
<td>8</td>
</tr>
<tr>
<td>Constant</td>
<td>Value</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>fpCurrencyFieldAlgerian</td>
<td>5121</td>
</tr>
<tr>
<td>fpCurrencyFieldArgentina</td>
<td>11274</td>
</tr>
<tr>
<td>fpCurrencyFieldAustralia</td>
<td>3081</td>
</tr>
<tr>
<td>fpCurrencyFieldAustria</td>
<td>3079</td>
</tr>
<tr>
<td>fpCurrencyFieldBahraini</td>
<td>15361</td>
</tr>
<tr>
<td>fpCurrencyFieldBelgiumBF</td>
<td>2067</td>
</tr>
<tr>
<td>fpCurrencyFieldBelgiumFB</td>
<td>2060</td>
</tr>
<tr>
<td>fpCurrencyFieldBolivia</td>
<td>16394</td>
</tr>
<tr>
<td>fpCurrencyFieldBrazil</td>
<td>1046</td>
</tr>
<tr>
<td>fpCurrencyFieldCanada</td>
<td>4105</td>
</tr>
<tr>
<td>fpCurrencyFieldCanadaF</td>
<td>3084</td>
</tr>
<tr>
<td>fpCurrencyFieldChile</td>
<td>13322</td>
</tr>
<tr>
<td>fpCurrencyFieldColombia</td>
<td>9226</td>
</tr>
<tr>
<td>fpCurrencyFieldCostaRica</td>
<td>5130</td>
</tr>
<tr>
<td>fpCurrencyFieldCzech</td>
<td>1029</td>
</tr>
<tr>
<td>fpCurrencyFieldDenmark</td>
<td>1030</td>
</tr>
<tr>
<td>fpCurrencyFieldDominicanRepublic</td>
<td>7178</td>
</tr>
<tr>
<td>fpCurrencyFieldEcuador</td>
<td>12298</td>
</tr>
<tr>
<td>fpCurrencyFieldEgyptian</td>
<td>3073</td>
</tr>
<tr>
<td>fpCurrencyFieldElSalvador</td>
<td>17418</td>
</tr>
<tr>
<td>fpCurrencyFieldEuro</td>
<td>-1</td>
</tr>
<tr>
<td>fpCurrencyFieldEuroPostfix</td>
<td>-2</td>
</tr>
<tr>
<td>fpCurrencyFieldFinland</td>
<td>1035</td>
</tr>
<tr>
<td>fpCurrencyFieldFrance</td>
<td>1036</td>
</tr>
<tr>
<td>fpCurrencyFieldGermany</td>
<td>1031</td>
</tr>
<tr>
<td>Currency Field</td>
<td>Code</td>
</tr>
<tr>
<td>------------------------</td>
<td>------</td>
</tr>
<tr>
<td>fpCurrencyFieldGreece</td>
<td>1032</td>
</tr>
<tr>
<td>fpCurrencyFieldGuatemala</td>
<td>4106</td>
</tr>
<tr>
<td>fpCurrencyFieldHonduras</td>
<td>18442</td>
</tr>
<tr>
<td>fpCurrencyFieldHongKong</td>
<td>3076</td>
</tr>
<tr>
<td>fpCurrencyFieldHungary</td>
<td>1038</td>
</tr>
<tr>
<td>fpCurrencyFieldIndian</td>
<td>1081</td>
</tr>
<tr>
<td>fpCurrencyFieldIranian</td>
<td>1065</td>
</tr>
<tr>
<td>fpCurrencyFieldIraqi</td>
<td>2049</td>
</tr>
<tr>
<td>fpCurrencyFieldIreland</td>
<td>6153</td>
</tr>
<tr>
<td>fpCurrencyFieldIsraeli</td>
<td>1037</td>
</tr>
<tr>
<td>fpCurrencyFieldItaly</td>
<td>1040</td>
</tr>
<tr>
<td>fpCurrencyFieldJapan</td>
<td>1041</td>
</tr>
<tr>
<td>fpCurrencyFieldJordanian</td>
<td>11265</td>
</tr>
<tr>
<td>fpCurrencyFieldKorea</td>
<td>1042</td>
</tr>
<tr>
<td>fpCurrencyFieldKuwaiti</td>
<td>13313</td>
</tr>
<tr>
<td>fpCurrencyFieldLebanese</td>
<td>12289</td>
</tr>
<tr>
<td>fpCurrencyFieldLibyan</td>
<td>4097</td>
</tr>
<tr>
<td>fpCurrencyFieldMexico</td>
<td>2058</td>
</tr>
<tr>
<td>fpCurrencyFieldMoroccan</td>
<td>6145</td>
</tr>
<tr>
<td>fpCurrencyFieldNetherlands</td>
<td>1043</td>
</tr>
<tr>
<td>fpCurrencyFieldNewZealand</td>
<td>5129</td>
</tr>
<tr>
<td>fpCurrencyFieldNicaragua</td>
<td>19466</td>
</tr>
<tr>
<td>fpCurrencyFieldNorway</td>
<td>1044</td>
</tr>
<tr>
<td>fpCurrencyFieldOmani</td>
<td>8193</td>
</tr>
<tr>
<td>fpCurrencyFieldPakistani</td>
<td>1056</td>
</tr>
<tr>
<td>fpCurrencyFieldPanama</td>
<td>6154</td>
</tr>
<tr>
<td>fpCurrencyFieldParaguay</td>
<td>15370</td>
</tr>
<tr>
<td>fpCurrencyFieldPeru</td>
<td>10250</td>
</tr>
<tr>
<td>fpCurrencyFieldPoland</td>
<td>1045</td>
</tr>
<tr>
<td>fpCurrencyFieldPortugal</td>
<td>2070</td>
</tr>
<tr>
<td>fpCurrencyFieldPRChina</td>
<td>2052</td>
</tr>
<tr>
<td>fpCurrencyFieldQatari</td>
<td>16385</td>
</tr>
<tr>
<td>fpCurrencyFieldRussia</td>
<td>1049</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>fpCurrencyField</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saudi Arabian</td>
<td>1025</td>
</tr>
<tr>
<td>Singapore</td>
<td>4100</td>
</tr>
<tr>
<td>Slovakia</td>
<td>1051</td>
</tr>
<tr>
<td>Slovenia</td>
<td>1060</td>
</tr>
<tr>
<td>South Africa</td>
<td>7177</td>
</tr>
<tr>
<td>Spain</td>
<td>1034</td>
</tr>
<tr>
<td>Sweden</td>
<td>1053</td>
</tr>
<tr>
<td>Switzerland</td>
<td>2055</td>
</tr>
<tr>
<td>Syrian</td>
<td>10241</td>
</tr>
<tr>
<td>Taiwan</td>
<td>1028</td>
</tr>
<tr>
<td>Thai</td>
<td>1054</td>
</tr>
<tr>
<td>Tunisian</td>
<td>7169</td>
</tr>
<tr>
<td>Turkey</td>
<td>1055</td>
</tr>
<tr>
<td>UAE</td>
<td>14337</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>2057</td>
</tr>
<tr>
<td>United States</td>
<td>1033</td>
</tr>
<tr>
<td>Uruguay</td>
<td>14346</td>
</tr>
<tr>
<td>Venezuela</td>
<td>8202</td>
</tr>
<tr>
<td>Vietnamese</td>
<td>1066</td>
</tr>
<tr>
<td>Yemeni</td>
<td>9217</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FpDateTimeFieldFormat</th>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpDateTimeFieldDateAndTime</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>fpDateTimeFieldDateOnly</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FpDependencyFlags</th>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpDepsDefault</td>
<td>255</td>
<td></td>
</tr>
<tr>
<td>fpDepsImages</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>fpDepsLinkbars</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>fpDepsLinks</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
fpDepsLists    4
fpDepsNavbars  32
fpDepsNone     0
fpDepsRecurse  256
fpDepsSharedBorders  128
fpDepsThemes    8
fpDepsWebParts  64

FpFieldType

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpFieldAttachments</td>
<td>13</td>
</tr>
<tr>
<td>fpFieldChoice</td>
<td>5</td>
</tr>
<tr>
<td>fpFieldComputed</td>
<td>11</td>
</tr>
<tr>
<td>fpFieldCounter</td>
<td>9</td>
</tr>
<tr>
<td>fpFieldCurrency</td>
<td>3</td>
</tr>
<tr>
<td>fpFieldDateTime</td>
<td>4</td>
</tr>
<tr>
<td>fpFieldFile</td>
<td>12</td>
</tr>
<tr>
<td>fpFieldInteger</td>
<td>10</td>
</tr>
<tr>
<td>fpFieldLookup</td>
<td>6</td>
</tr>
<tr>
<td>fpFieldMultiLine</td>
<td>1</td>
</tr>
<tr>
<td>fpFieldNumber</td>
<td>2</td>
</tr>
<tr>
<td>fpFieldRatingScale</td>
<td>14</td>
</tr>
<tr>
<td>fpFieldSingleLine</td>
<td>0</td>
</tr>
<tr>
<td>fpFieldTrueFalse</td>
<td>7</td>
</tr>
<tr>
<td>fpFieldURL</td>
<td>8</td>
</tr>
</tbody>
</table>

FpFolderType

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpFolderDHTMLSharedLib</td>
<td>1</td>
</tr>
</tbody>
</table>

FpLayoutCellAttributes
<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>border_bottom_color</td>
<td>18</td>
</tr>
<tr>
<td>border_bottom_size</td>
<td>12</td>
</tr>
<tr>
<td>border_foot_color</td>
<td>20</td>
</tr>
<tr>
<td>border_foot_size</td>
<td>14</td>
</tr>
<tr>
<td>border_head_color</td>
<td>19</td>
</tr>
<tr>
<td>border_head_size</td>
<td>13</td>
</tr>
<tr>
<td>border_left_color</td>
<td>16</td>
</tr>
<tr>
<td>border_left_size</td>
<td>10</td>
</tr>
<tr>
<td>border_right_color</td>
<td>17</td>
</tr>
<tr>
<td>border_right_size</td>
<td>11</td>
</tr>
<tr>
<td>border_top_color</td>
<td>15</td>
</tr>
<tr>
<td>border_top_size</td>
<td>9</td>
</tr>
<tr>
<td>cellFomattingID</td>
<td>0</td>
</tr>
<tr>
<td>content_color</td>
<td>45</td>
</tr>
<tr>
<td>content_html</td>
<td>47</td>
</tr>
<tr>
<td>content_padding</td>
<td>46</td>
</tr>
<tr>
<td>content_type</td>
<td>4</td>
</tr>
<tr>
<td>content_valign</td>
<td>48</td>
</tr>
<tr>
<td>corner_bottom_left_border_color</td>
<td>37</td>
</tr>
<tr>
<td>corner_bottom_left_color</td>
<td>36</td>
</tr>
<tr>
<td>corner_bottom_left_height</td>
<td>35</td>
</tr>
<tr>
<td>corner_bottom_left_imageurl</td>
<td>38</td>
</tr>
<tr>
<td>corner_bottom_left_type</td>
<td>33</td>
</tr>
<tr>
<td>corner_bottom_left_width</td>
<td>34</td>
</tr>
<tr>
<td>corner_bottom_right_border_color</td>
<td>43</td>
</tr>
<tr>
<td>corner_bottom_right_color</td>
<td>42</td>
</tr>
<tr>
<td>corner_bottom_right_height</td>
<td>41</td>
</tr>
<tr>
<td>corner_bottom_right_imageurl</td>
<td>44</td>
</tr>
<tr>
<td>corner_bottom_right_type</td>
<td>39</td>
</tr>
<tr>
<td>corner_bottom_right_width</td>
<td>40</td>
</tr>
<tr>
<td>corner_top_left_border_color</td>
<td>25</td>
</tr>
<tr>
<td>corner_top_left_color</td>
<td>24</td>
</tr>
</tbody>
</table>
### FpLengthUnits

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpLengthUnitsAtom</td>
<td>11</td>
</tr>
<tr>
<td>fpLengthUnitsCM</td>
<td>7</td>
</tr>
<tr>
<td>fpLengthUnitsEM</td>
<td>4</td>
</tr>
<tr>
<td>fpLengthUnitsEX</td>
<td>5</td>
</tr>
<tr>
<td>fpLengthUnitsHalfPoints</td>
<td>12</td>
</tr>
<tr>
<td>fpLengthUnitsHTMLFontUnits</td>
<td>13</td>
</tr>
<tr>
<td>fpLengthUnitsIN</td>
<td>6</td>
</tr>
<tr>
<td>fpLengthUnitsInvalid</td>
<td>15</td>
</tr>
<tr>
<td>fpLengthUnitsMM</td>
<td>8</td>
</tr>
<tr>
<td>fpLengthUnitsNone</td>
<td>0</td>
</tr>
<tr>
<td>fpLengthUnitsNoValue</td>
<td>14</td>
</tr>
<tr>
<td>fpLengthUnitsPC</td>
<td>10</td>
</tr>
<tr>
<td>fpLengthUnitsPercent</td>
<td>2</td>
</tr>
<tr>
<td>fpLengthUnitsPT</td>
<td>9</td>
</tr>
<tr>
<td>fpLengthUnitsPX</td>
<td>3</td>
</tr>
<tr>
<td>fpLengthUnitsRelative</td>
<td>1</td>
</tr>
</tbody>
</table>

### FpListDesignSecurity

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpListDesignSecurityCreator</td>
<td>1</td>
</tr>
<tr>
<td>fpListDesignSecurityEveryone</td>
<td>0</td>
</tr>
</tbody>
</table>

### FpListEditSecurity

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpListEditSecurityAll</td>
<td>0</td>
</tr>
<tr>
<td>fpListEditSecurityNone</td>
<td>2</td>
</tr>
<tr>
<td>fpListEditSecurityOnlyOwn</td>
<td>1</td>
</tr>
</tbody>
</table>

### FpListReadSecurity
<table>
<thead>
<tr>
<th><strong>Constant</strong></th>
<th><strong>Value</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>fpListReadSecurityAll</td>
<td>0</td>
</tr>
<tr>
<td>fpListReadSecurityOnlyOwn</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>FpListType</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>fpListTypeBasicList</td>
<td>0</td>
</tr>
<tr>
<td>fpListTypeDiscussion</td>
<td>3</td>
</tr>
<tr>
<td>fpListTypeDocumentLibrary</td>
<td>2</td>
</tr>
<tr>
<td>fpListTypeSurvey</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>FpNumberFieldFormat</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>fpNumberFieldAuto</td>
<td>-1</td>
</tr>
<tr>
<td>fpNumberFieldFiveDecimals</td>
<td>5</td>
</tr>
<tr>
<td>fpNumberFieldFourDecimals</td>
<td>4</td>
</tr>
<tr>
<td>fpNumberFieldInteger</td>
<td>0</td>
</tr>
<tr>
<td>fpNumberFieldOneDecimal</td>
<td>1</td>
</tr>
<tr>
<td>fpNumberFieldThreeDecimals</td>
<td>3</td>
</tr>
<tr>
<td>fpNumberFieldTwoDecimals</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>FpOpenAs</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>fpNeverOpen</td>
<td>4</td>
</tr>
<tr>
<td>fpOpenAsCount</td>
<td>5</td>
</tr>
<tr>
<td>fpOpenAsCSS</td>
<td>1</td>
</tr>
<tr>
<td>fpOpenAsHTML</td>
<td>0</td>
</tr>
<tr>
<td>fpOpenAsNotApplicable</td>
<td>-1</td>
</tr>
<tr>
<td>fpOpenAsText</td>
<td>2</td>
</tr>
<tr>
<td>fpOpenAsXml</td>
<td>3</td>
</tr>
</tbody>
</table>
### FpOptimizeHTMLFlags

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpHtmlOptAdjacentTags</td>
<td>64</td>
</tr>
<tr>
<td>fpHtmlOptAuthorComponents</td>
<td>8</td>
</tr>
<tr>
<td>fpHtmlOptBots</td>
<td>8</td>
</tr>
<tr>
<td>fpHtmlOptBrowseComponents</td>
<td>131072</td>
</tr>
<tr>
<td>fpHtmlOptCellFormattingAttr</td>
<td>256</td>
</tr>
<tr>
<td>fpHtmlOptDreamWeaver</td>
<td>2</td>
</tr>
<tr>
<td>fpHtmlOptDwtCmnts</td>
<td>1024</td>
</tr>
<tr>
<td>fpHtmlOptEmpty</td>
<td>32</td>
</tr>
<tr>
<td>fpHtmlOptGenerator</td>
<td>16</td>
</tr>
<tr>
<td>fpHtmlOptHTMLAllWhitespace</td>
<td>16384</td>
</tr>
<tr>
<td>fpHtmlOptHTMLCmnts</td>
<td>4096</td>
</tr>
<tr>
<td>fpHtmlOptHTMLLeadWhitespace</td>
<td>8192</td>
</tr>
<tr>
<td>fpHtmlOptHTMLMisnest</td>
<td>32768</td>
</tr>
<tr>
<td>fpHtmlOptOn</td>
<td>-2147483648</td>
</tr>
<tr>
<td>fpHtmlOptScriptCmnts</td>
<td>2048</td>
</tr>
<tr>
<td>fpHtmlOptThemes</td>
<td>4</td>
</tr>
<tr>
<td>fpHtmlOptTrcImageAttr</td>
<td>512</td>
</tr>
<tr>
<td>fpHtmlOptUnusedStyles</td>
<td>65536</td>
</tr>
<tr>
<td>fpHtmlOptVMLGraphics</td>
<td>128</td>
</tr>
<tr>
<td>fpHtmlOptWordHTML</td>
<td>1</td>
</tr>
</tbody>
</table>

### FpPageViewMode

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpPageViewDefault</td>
<td>0</td>
</tr>
<tr>
<td>fpPageViewHtml</td>
<td>2</td>
</tr>
<tr>
<td>fpPageViewInBetween</td>
<td>512</td>
</tr>
<tr>
<td>fpPageViewInvalid</td>
<td>-1</td>
</tr>
<tr>
<td>fpPageViewNoFrames</td>
<td>4</td>
</tr>
<tr>
<td>fpPageViewNormal</td>
<td>1</td>
</tr>
</tbody>
</table>
fpPageViewNoWindow  128  
fpPageViewPreview    8 
fpPageViewPrintPreview 256 
fpPageViewSplit      16     
fpPageViewText       32     
fpPageViewXml        64

### FpPkgImportConflictOpts

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpPkgFileConflictMask</td>
<td>15</td>
</tr>
<tr>
<td>fpPkgListConflictMask</td>
<td>240</td>
</tr>
<tr>
<td>fpPkgOnConflictSkip</td>
<td>17</td>
</tr>
<tr>
<td>fpPkgOnConflictStop</td>
<td>0</td>
</tr>
<tr>
<td>fpPkgOnFileConflictOverwrite</td>
<td>2</td>
</tr>
<tr>
<td>fpPkgOnFileConflictSkip</td>
<td>1</td>
</tr>
<tr>
<td>fpPkgOnFileConflictStop</td>
<td>0</td>
</tr>
<tr>
<td>fpPkgOnListConflictMergeOrRename</td>
<td>64</td>
</tr>
<tr>
<td>fpPkgOnListConflictMergeOrSkip</td>
<td>48</td>
</tr>
<tr>
<td>fpPkgOnListConflictMergeOrStop</td>
<td>32</td>
</tr>
<tr>
<td>fpPkgOnListConflictRename</td>
<td>80</td>
</tr>
<tr>
<td>fpPkgOnListConflictSkip</td>
<td>16</td>
</tr>
<tr>
<td>fpPkgOnListConflictStop</td>
<td>0</td>
</tr>
</tbody>
</table>

### FpPkgImportResult

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpPkgImportCancelled</td>
<td>2</td>
</tr>
<tr>
<td>fpPkgImportComplete</td>
<td>0</td>
</tr>
<tr>
<td>fpPkgImportErrorInPackage</td>
<td>3</td>
</tr>
<tr>
<td>fpPkgImportFailed</td>
<td>1</td>
</tr>
<tr>
<td>fpPkgImportNotTrusted</td>
<td>5</td>
</tr>
<tr>
<td>fpPkgImportServerNotSupported</td>
<td>6</td>
</tr>
<tr>
<td>fpPkgImportStopped</td>
<td>4</td>
</tr>
</tbody>
</table>
### FpPkgTrustLevel

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpPkgTrustAll</td>
<td>1</td>
</tr>
<tr>
<td>fpPkgTrustCertificateStore</td>
<td>2</td>
</tr>
</tbody>
</table>

### FpSearchAction

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpSearchFindTag</td>
<td>3</td>
</tr>
<tr>
<td>fpSearchFindText</td>
<td>0</td>
</tr>
<tr>
<td>fpSearchReplaceAllText</td>
<td>2</td>
</tr>
<tr>
<td>fpSearchReplaceText</td>
<td>1</td>
</tr>
</tbody>
</table>

### FpSearchOptions

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpSearchIgnoreWhitespace</td>
<td>4</td>
</tr>
<tr>
<td>fpSearchMatchAlef</td>
<td>128</td>
</tr>
<tr>
<td>fpSearchMatchCase</td>
<td>1</td>
</tr>
<tr>
<td>fpSearchMatchDiacritics</td>
<td>64</td>
</tr>
<tr>
<td>fpSearchMatchKashida</td>
<td>32</td>
</tr>
<tr>
<td>fpSearchRegExp</td>
<td>8</td>
</tr>
<tr>
<td>fpSearchReviseHangul</td>
<td>256</td>
</tr>
<tr>
<td>fpSearchWholeWord</td>
<td>2</td>
</tr>
</tbody>
</table>

### FpSharedBorders

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>_fpBorderNone</td>
<td>0</td>
</tr>
<tr>
<td>fpBorderAll</td>
<td>255</td>
</tr>
<tr>
<td>fpBorderBottom</td>
<td>8</td>
</tr>
<tr>
<td>fpBorderLeft</td>
<td>2</td>
</tr>
<tr>
<td>FpBorderRight</td>
<td>4</td>
</tr>
<tr>
<td>--------------</td>
<td>---</td>
</tr>
<tr>
<td>FpBorderTop</td>
<td>1</td>
</tr>
</tbody>
</table>

**FpStructModType**

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpStructBaseOnSibling</td>
<td>0</td>
</tr>
<tr>
<td>fpStructLeftmostChild</td>
<td>1</td>
</tr>
<tr>
<td>fpStructRightmostChild</td>
<td>2</td>
</tr>
</tbody>
</table>

**FpThemeProperties**

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpThemeActiveGraphics</td>
<td>16</td>
</tr>
<tr>
<td>fpThemeBackgroundColor</td>
<td>1</td>
</tr>
<tr>
<td>fpThemeCSS</td>
<td>4096</td>
</tr>
<tr>
<td>fpThemeDefaultSettings</td>
<td>16777216</td>
</tr>
<tr>
<td>fpThemeName</td>
<td>33554432</td>
</tr>
<tr>
<td>fpThemeNoBackgroundImage</td>
<td>0</td>
</tr>
<tr>
<td>fpThemeNoCSS</td>
<td>0</td>
</tr>
<tr>
<td>fpThemeNormalColors</td>
<td>0</td>
</tr>
<tr>
<td>fpThemeNormalGraphics</td>
<td>0</td>
</tr>
<tr>
<td>fpThemePropertiesAll</td>
<td>4369</td>
</tr>
<tr>
<td>fpThemePropertiesNone</td>
<td>0</td>
</tr>
<tr>
<td>fpThemeVividColors</td>
<td>256</td>
</tr>
</tbody>
</table>

**FpURLComponent**

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpURLComponentBookmark</td>
<td>256</td>
</tr>
<tr>
<td>fpURLComponentFileExtension</td>
<td>128</td>
</tr>
<tr>
<td>fpURLComponentFileName</td>
<td>64</td>
</tr>
<tr>
<td>fpURLComponentPassword</td>
<td>4</td>
</tr>
<tr>
<td>fpURLComponentPath</td>
<td>32</td>
</tr>
</tbody>
</table>
fpURLComponentPort 16
fpURLComponentQuery 512
fpURLComponentScheme 1
fpURLComponentServer 8
fpURLComponentUserName 2

### FpURLFieldFormat

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpURLFieldImage</td>
<td>1</td>
</tr>
<tr>
<td>fpURLFieldLink</td>
<td>0</td>
</tr>
</tbody>
</table>

### FpWebDeleteFlags

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpDeleteEntireWeb</td>
<td>0</td>
</tr>
<tr>
<td>fpDeleteFrontPageInfoFromWeb</td>
<td>1</td>
</tr>
</tbody>
</table>

### FpWebOpenFlags

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpOpenInWindow</td>
<td>0</td>
</tr>
<tr>
<td>fpOpenNoWindow</td>
<td>2</td>
</tr>
</tbody>
</table>

### FpWebPublishFlags

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpPublishAddToExistingWeb</td>
<td>2</td>
</tr>
<tr>
<td>fpPublishCopyAllFiles</td>
<td>64</td>
</tr>
<tr>
<td>fpPublishCopySubwebs</td>
<td>4</td>
</tr>
<tr>
<td>fpPublishIncremental</td>
<td>1</td>
</tr>
<tr>
<td>fpPublishLogInTempDir</td>
<td>8</td>
</tr>
<tr>
<td>fpPublishNoDeleteUnmatched</td>
<td>16</td>
</tr>
<tr>
<td>fpPublishNone</td>
<td>0</td>
</tr>
<tr>
<td>Constant</td>
<td>Value</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>fpPublishRemoteToLocal</td>
<td>128</td>
</tr>
<tr>
<td>fpPublishSynchronize</td>
<td>256</td>
</tr>
<tr>
<td>fpPublishUseLastPublishTime</td>
<td>32</td>
</tr>
<tr>
<td>fpPublishUsingDav</td>
<td>512</td>
</tr>
<tr>
<td>fpPublishUsingPassiveFtp</td>
<td>1024</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpWebSubViewFolders</td>
<td>1</td>
</tr>
<tr>
<td>fpWebSubViewNavigation</td>
<td>2</td>
</tr>
<tr>
<td>fpWebSubViewNone</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpWebViewAllFiles</td>
<td>4</td>
</tr>
<tr>
<td>fpWebViewBrokenLinks</td>
<td>6</td>
</tr>
<tr>
<td>fpWebViewFolders</td>
<td>1</td>
</tr>
<tr>
<td>fpWebViewLinks</td>
<td>0</td>
</tr>
<tr>
<td>fpWebViewPage</td>
<td>3</td>
</tr>
<tr>
<td>fpWebViewRemoteSite</td>
<td>8</td>
</tr>
<tr>
<td>fpWebViewSiteSummary</td>
<td>7</td>
</tr>
<tr>
<td>fpWebViewStructure</td>
<td>2</td>
</tr>
<tr>
<td>fpWebViewTodo</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpWebViewExAccessibility</td>
<td>37</td>
</tr>
<tr>
<td>fpWebViewExAllFiles</td>
<td>4</td>
</tr>
<tr>
<td>fpWebViewExAssignedTo</td>
<td>13</td>
</tr>
<tr>
<td>fpWebViewExBrokenLinks</td>
<td>6</td>
</tr>
<tr>
<td>fpWebViewExBrowserTypes</td>
<td>29</td>
</tr>
<tr>
<td>Category</td>
<td>Value</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>fpWebViewExCategories</td>
<td>17</td>
</tr>
<tr>
<td>fpWebViewExCheckoutStatus</td>
<td>16</td>
</tr>
<tr>
<td>fpWebViewExComponentErrors</td>
<td>18</td>
</tr>
<tr>
<td>fpWebViewExCSSLinks</td>
<td>33</td>
</tr>
<tr>
<td>fpWebViewExDailyPageHits</td>
<td>19</td>
</tr>
<tr>
<td>fpWebViewExDailySummary</td>
<td>22</td>
</tr>
<tr>
<td>fpWebViewExFolders</td>
<td>1</td>
</tr>
<tr>
<td>fpWebViewExLinks</td>
<td>0</td>
</tr>
<tr>
<td>fpWebViewExMasterPages</td>
<td>32</td>
</tr>
<tr>
<td>fpWebViewExMonthlyPageHits</td>
<td>21</td>
</tr>
<tr>
<td>fpWebViewExMonthlySummary</td>
<td>24</td>
</tr>
<tr>
<td>fpWebViewExNavigation</td>
<td>2</td>
</tr>
<tr>
<td>fpWebViewExOlderFiles</td>
<td>8</td>
</tr>
<tr>
<td>fpWebViewExOsTypes</td>
<td>28</td>
</tr>
<tr>
<td>fpWebViewExPage</td>
<td>3</td>
</tr>
<tr>
<td>fpWebViewExPublishStatus</td>
<td>15</td>
</tr>
<tr>
<td>fpWebViewExRecentlyAddedFiles</td>
<td>9</td>
</tr>
<tr>
<td>fpWebViewExRecentlyChangedFiles</td>
<td>10</td>
</tr>
<tr>
<td>fpWebViewExReferringDomains</td>
<td>25</td>
</tr>
<tr>
<td>fpWebViewExReferringURLs</td>
<td>26</td>
</tr>
<tr>
<td>fpWebViewExRemoteSite</td>
<td>34</td>
</tr>
<tr>
<td>fpWebViewExReviewStatus</td>
<td>14</td>
</tr>
<tr>
<td>fpWebViewExSearchStrings</td>
<td>27</td>
</tr>
<tr>
<td>fpWebViewExSharedBorders</td>
<td>36</td>
</tr>
<tr>
<td>fpWebViewExSiteSummary</td>
<td>7</td>
</tr>
<tr>
<td>fpWebViewExSlowPages</td>
<td>11</td>
</tr>
<tr>
<td>fpWebViewExThemes</td>
<td>35</td>
</tr>
<tr>
<td>fpWebViewExTodo</td>
<td>5</td>
</tr>
<tr>
<td>fpWebViewExUnlinkedFiles</td>
<td>12</td>
</tr>
<tr>
<td>fpWebViewExUsageSummary</td>
<td>31</td>
</tr>
<tr>
<td>fpWebViewExVisitingUsers</td>
<td>30</td>
</tr>
<tr>
<td>fpWebViewExWeeklyPageHits</td>
<td>20</td>
</tr>
<tr>
<td>fpWebViewExWeeklySummary</td>
<td>23</td>
</tr>
<tr>
<td>Constant</td>
<td>Value</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>READYSTATE_COMPLETE</td>
<td>4</td>
</tr>
<tr>
<td>READYSTATE_INTERACTIVE</td>
<td>3</td>
</tr>
<tr>
<td>READYSTATE_LOADED</td>
<td>2</td>
</tr>
<tr>
<td>READYSTATE_LOADING</td>
<td>1</td>
</tr>
<tr>
<td>READYSTATE_UNINITIALIZED</td>
<td>0</td>
</tr>
</tbody>
</table>
behaviorElement Property

Returns an IHTMLElement object that represents the top-most parent element of active selection. Corresponds to the element specified for Scripts on Tag as shown in the Behaviors task pane.

expression.behaviorElement

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

If the insertion point is in the middle of a P element, the `behaviorElement` property returns the P element. If a selection spans a few table rows, the `behaviorElement` property returns the parent TABLE element, since that is the top-most parent element of the selection. If a selection spans the entire document, the `behaviorElement` property would return the BODY element.
Example

The following example adds a script to the top-most parent element of the active selection in the active document.

Dim objElement As IHTMLElement
Set objElement = ActiveDocument.behaviorElement

ActiveDocument.addScript objElement, _
    "onclick", "this.innerText=prompt" & _
    "('Please enter your name: ', '')"
id Property

Returns or sets a String that represents the value of the id attribute for an HTML element.

expression.id

expression  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The value of the **id** property is used when programmatically accessing an element in a document. Therefore, the **id** attribute for an element should be unique throughout the scope of the document. For example, the following code would access the TABLE element in the active document that has an **id** attribute equal to "regions". If none is found, the object variable `objTable` is equal to `Nothing`.

```vba
Dim objTable As FPHTMLTable
Set objTable = ActiveDocument.all.tags("table").Item("regions")
```

If a document contains more than one element with the same **id** attribute, the elements are returned as an **IHTMLElementCollection** collection that can be referenced only by using the ordinal position. For example, the following example returns a collection of paragraphs with an **id** attribute equal to "intro" and then accesses the first paragraph in that collection by using its ordinal position.

```vba
Dim objParas As IHTMLElementCollection
Dim objPara As FPHTMLParaElement
Set objParas = ActiveDocument.all.tags("P").Item("intro")
Set objPara = objParas.Item(0)
```
Example

The following example adds "intropara" to the first paragraph in the active document.

ActiveDocument.all.tags("P").Item(0).Id = "intropara"
href Property

Returns or sets a String that represents the URL associated with a hyperlink. The href property corresponds to the href attribute of an A element.

expression.href

expression Required. An expression that returns one of the objects in the Applies To list.
Remarks

Hyperlinks are A elements with the `href` attribute and are members of the element collection returned by using the `links` property. Bookmarks are A elements with the `name` attribute and are members of the element collection returned by using the `anchors` property.
Example

The following example sets the `href` attribute to "http://www.microsoft.com" for the first A element in the active document.

```
ActiveDocument.body.all.tags("a").Item(0).href = "http://www.microso
```
**Body Property**

Returns an **IHTMLBody** object that represents the tags and text between the opening and closing tags of the **BODY** element in the specified document.

`expression.body`

`expression`  Required. An expression that returns one of the objects in the Applies To list.
Remarks

Use the all property to get all elements in the body of a document, regardless of hierarchy. Use the children property to get only the top-level elements directly beneath the BODY element.

Note  In the Microsoft Internet Explorer Document Object Model, the body property can represent either a BODY or FRAMESET element, if a Web page contains frames. This is not the case with the Microsoft FrontPage Page Object Model. To access FRAMESET elements with the FrontPage Page Object Model, use the tags method of the IHTMLElementCollection object to return an FPHTMLFrameSetSite or IHTMLFrameSetElement object.
Example

The following example sets the colors for active, viewed, and regular hyperlinks, and sets the background color for the active document.

Function ChangeLinkColors(ByRef objDoc As FPHTMLDocument, _
                            ByRef Optional strALink As String, ByRef Optional strVLink As String, _
                            ByRef Optional strLink As String, ByRef Optional strBGColor As String) As Boolean

    If strALink <> "" Or strVLink <> "" Or strLink <> "" Or strBGCol
        With objDoc.body
            .aLink = strALink
            .vLink = strVLink
            .link = strLink
            .bgColor = strBGColor
        End With
        ChangeLinkColors = True
    Else
        ChangeLinkColors = False
    End If
End Function

Use the following example to call the preceding function.

Sub CallChangeLinkColors()
    Call ChangeLinkColors(objDoc:=ActiveDocument, strALink:="blue", strVLink:="yellow", strLink:="green", strBGColor:="black")
End Sub
IHTMLSpanElement Object

IHTMLSpanFlow

Represents a SPAN element in an HTML document. The IHTMLSpanElement object provides access to a limited number of properties and methods related to the SPAN element. For access to all properties and methods, use the FPHTMLSpanElement object.
Using the IHTMLSpanElement object

Use the the `tags` method to return an `IHTMLElementCollection` collection that represents a collection of all the SPAN elements in a document. Use the `Item` method to return an `IHTMLSpanElement` object that accesses a specific SPAN element, referenced by ordinal number or by the value of the `id` attribute. The following example accesses the first SPAN element in the active document.

```vbscript
Dim objSelect As IHTMLSpanElement
Set objSelect = ActiveDocument.all.tags("span").Item(0)
```
**innerHTML Property**

Sets or returns a **String** that represents the text and HTML elements between the start and end tags of a specified element.

`expression.innerHTML`

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

See also the innerText, outerText, and outerHTML properties.
Example

The following example changes the HTML and text inside the first H1 element in the active document to "<i>Microsoft Home Page</i>"

Dim objTag As IHTMLElement
Set objTag = ActiveDocument.all.tags("h1").Item(0)
objTag.innerHTML = "<i>Microsoft Home Page</i>"
Colors can be specified in HTML pages in two ways—by using a color name, or by using numbers to denote a red-green-blue (RGB) color value. An RGB color value consists of three two-digit hexadecimal numbers specifying the intensity of the corresponding color.

For example, the color value #FF0000 is rendered red because the red number is set to its highest value, FF (or 255 in decimal form).

**Note** While these color names might not be recognized by other browsers, the RGB color values should display accurately across browsers. When specifying color values for Web pages targeted to work across different browsers, use the RGB color values.

The following table lists the colors supported by Microsoft Internet Explorer version 4.0 and later.

<table>
<thead>
<tr>
<th>Color</th>
<th>Code</th>
<th>Color</th>
<th>Code</th>
<th>Color</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>aliceblue</td>
<td>#F0F8FF</td>
<td>antiquewhite</td>
<td>#FAEBD7</td>
<td>aqua</td>
<td>#00FFFF</td>
</tr>
<tr>
<td>azure</td>
<td>#00FFFF</td>
<td>beige</td>
<td>#F5F5DC</td>
<td>bisque</td>
<td>#FFE4C4</td>
</tr>
<tr>
<td>blanchedalmond</td>
<td>#FFEBCD</td>
<td>blue</td>
<td>#0000FF</td>
<td>blueviolet</td>
<td>#8A2BE2</td>
</tr>
<tr>
<td>burlywood</td>
<td>#DEB887</td>
<td>cadetblue</td>
<td>#5F9EA0</td>
<td>chartreuse</td>
<td>#7FFF00</td>
</tr>
<tr>
<td>coral</td>
<td>#FF7F50</td>
<td>cornflower</td>
<td>#6495ED</td>
<td>cornsilk</td>
<td>#FFF8DC</td>
</tr>
<tr>
<td>cyan</td>
<td>#00FFFF</td>
<td>darkblue</td>
<td>#00008B</td>
<td>darkcyan</td>
<td>#008B8B</td>
</tr>
<tr>
<td>darkgray</td>
<td>#A9A9A9</td>
<td>darkgreen</td>
<td>#006400</td>
<td>darkkhaki</td>
<td>#BDB76B</td>
</tr>
<tr>
<td>darkolivegreen</td>
<td>#556B2F</td>
<td>darkorange</td>
<td>#FF8C00</td>
<td>darkorchid</td>
<td>#9932CC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>darksalmon (#E9967A)</td>
<td>darkseagreen (#8BCB8B)</td>
<td>darkslateblue (#483D8B)</td>
<td>darkslategray (#2F4F4F)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------</td>
<td>--------------------------</td>
<td>--------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>darkturquoise (#00CED1)</td>
<td>darkviolet (#9400D3)</td>
<td>deeppink (#FF1493)</td>
<td>deepskyblue (#00BFFF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dimgray (#696969)</td>
<td>dodgerblue (#1E90FF)</td>
<td>firebrick (#B22222)</td>
<td>floralwhite (#FFF0F0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>forestgreen (#228B22)</td>
<td>fuchsia (#FF00FF)</td>
<td>gainsboro (#DCDCDC)</td>
<td>ghostwhite (#F8F8FF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gold (#FFD700)</td>
<td>goldenrod (#DAA520)</td>
<td>gray (#808080)</td>
<td>green (#008000)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>greenyellow (#ADFF2F)</td>
<td>honeydew (#F0FFF0)</td>
<td>hotpink (#FF69B4)</td>
<td>indianred (#CD5C5C)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>indigo (#4B0082)</td>
<td>ivory (#FFFFFF)</td>
<td>khaki (#F0E68C)</td>
<td>lavender (#E6E6FA)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lavenderblush (#FFF0F5)</td>
<td>lawngreen (#7CFC00)</td>
<td>lemonchiffon (#FFACD)</td>
<td>lightblue (#ADD8E6)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lightcoral (#F08080)</td>
<td>lightcyan (#E0FFFF)</td>
<td>lightgoldenrodyellow (#FAFAD2)</td>
<td>lightgreen (#90EE90)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lightgray (#D3D3D3)</td>
<td>lightpink (#FFB6C1)</td>
<td>lightsalmon (#FFA07A)</td>
<td>lightseagreen (#20B2AA)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lightskyblue (#87CEFA)</td>
<td>lightslategrey (#778899)</td>
<td>lightsteelblue (#B0C4DE)</td>
<td>lightyellow (#FFFFE0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lime (#00FF00)</td>
<td>limegreen (#32CD32)</td>
<td>linen (#FAE0E6)</td>
<td>magenta (#FF00FF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>maroon (#800000)</td>
<td>mediumaquamarine (#66CDAA)</td>
<td>mediumblue (#0000CD)</td>
<td>mediumorchid (#BA55D3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mediumpurple (#9370DB)</td>
<td>mediumseagreen (#3CB371)</td>
<td>mediumslateblue (#7B68EE)</td>
<td>mediumspringgreen (#00FA9A)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mediumturquoise (#48D1CC)</td>
<td>mediumvioletred (#C71585)</td>
<td>midnightblue (#191970)</td>
<td>mintcream (#F5FFFA)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mistyrose (#FFE4E1)</td>
<td>moccasin (#FFE4B5)</td>
<td>navajowhite (#FFE4B5)</td>
<td>navy (#000080)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>oldlace (#FDF5E6)</td>
<td>olive (#808000)</td>
<td>olivedrab (#6B8E23)</td>
<td>orange (#FFA500)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
User-Defined System Colors

Microsoft Windows® maintains a set of system colors for painting various parts of the display. In addition to the colors defined in the color table, in Internet Explorer 4.0 and later these system colors may also be specified as color values in Web pages.

Users can set system colors for future Windows sessions using the Windows Control Panel. The following table illustrates how these color names correspond to the colors defined in the Control Panel.

Note The colors that do not correspond to any Control Panel colors can only be set programmatically using Windows APIs and cannot be defined by the user.

<table>
<thead>
<tr>
<th>activeborder</th>
<th>activecaption</th>
<th>appworkspace</th>
<th>background</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttonface</td>
<td>buttonhighlight</td>
<td>buttonshadow</td>
<td>buttontext</td>
</tr>
<tr>
<td>captiontext</td>
<td>graytext</td>
<td>highlight</td>
<td>highlighttext</td>
</tr>
<tr>
<td>inactiveborder</td>
<td>inactivecaption</td>
<td>inactivecaptiontext</td>
<td>infobackground</td>
</tr>
<tr>
<td>infotext</td>
<td>menu</td>
<td>menutext</td>
<td>scrollbar</td>
</tr>
<tr>
<td>threeddarkshadow</td>
<td>threedface</td>
<td>threedhighlight</td>
<td>threedlightshadow</td>
</tr>
<tr>
<td>threedshadow</td>
<td>window</td>
<td>windowframe</td>
<td>windowtext</td>
</tr>
</tbody>
</table>
**className Property**

Returns or sets a **String** that represents the value of the **class** attribute of the specified object.

```plaintext
expression.className
```

*expression*  Required. An expression that returns one of the objects in the Applies To list.
Remarks

The **class** attribute typically associates an element with a particular style rule in a style sheet.
Example

The following example sets the value of the className property of the specified tag.

Function SetClassName(ByRef strClassName As String, _
                      ByRef strTag As String) As Boolean

    Dim objDoc As FPHTMLDocument
    Dim intCounter As Integer

    Set objDoc = ActiveDocument

    For intCounter = 0 To objDoc.all.Length
        If objDoc.all(intCounter).tagName = strTag Then
            objDoc.all(intCounter).className = strClassName
            Exit For
        End If
    Next

End Function

Use the following example to call the preceding function.

Sub CallSetClassName()
    Call SetClassName(strClassName:="new", strTag:="body")
End Sub
## HTML Character Sets

Character sets determine how the bytes that represent the text of your HTML document are translated to readable characters. A Web browser interprets the bytes in your document according to the applied character set translations. It interprets numeric or hex character references ("&amp;#12345;" or "&amp;#x1234;") as ISO10646 code points, consistent with the Unicode Standard, version 2.0, and independent of the chosen character set. Named entities ("&#") are displayed independently of the chosen character set as well. The display of an arbitrary numeric character reference requires the existence of a font that is able to display that particular character on the user's system. Accordingly, the content in the first column of the following tables may not render as expected on all systems.

### ISO Latin-1 Character Set

The following table contains the complete ISO Latin-1 character set, corresponding to the first 256 entries of the Unicode character repertoire. The table provides each character, its decimal code, its named entity reference for HTML, and also a brief description.

<table>
<thead>
<tr>
<th>Character</th>
<th>Decimal code</th>
<th>Named entity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>&amp;#00;</td>
<td>---</td>
<td>Unused</td>
</tr>
<tr>
<td>---</td>
<td>&amp;#01;</td>
<td>---</td>
<td>Unused</td>
</tr>
<tr>
<td>---</td>
<td>&amp;#02;</td>
<td>---</td>
<td>Unused</td>
</tr>
<tr>
<td>---</td>
<td>&amp;#03;</td>
<td>---</td>
<td>Unused</td>
</tr>
<tr>
<td>---</td>
<td>&amp;#04;</td>
<td>---</td>
<td>Unused</td>
</tr>
<tr>
<td>---</td>
<td>&amp;#05;</td>
<td>---</td>
<td>Unused</td>
</tr>
<tr>
<td>---</td>
<td>&amp;#06;</td>
<td>---</td>
<td>Unused</td>
</tr>
<tr>
<td>---</td>
<td>&amp;#07;</td>
<td>---</td>
<td>Unused</td>
</tr>
<tr>
<td>---</td>
<td>&amp;#08;</td>
<td>---</td>
<td>Unused</td>
</tr>
<tr>
<td>---</td>
<td>&amp;#09;</td>
<td>---</td>
<td>Horizontal tab</td>
</tr>
<tr>
<td>---</td>
<td>&amp;#10;</td>
<td>---</td>
<td>Line feed</td>
</tr>
<tr>
<td>---</td>
<td>&amp;#11;</td>
<td>---</td>
<td>Unused</td>
</tr>
</tbody>
</table>
--- &amp;#12; --- Unused
--- &amp;#13; --- Carriage Return
--- &amp;#14; --- Unused
--- &amp;#15; --- Unused
--- &amp;#16; --- Unused
--- &amp;#17; --- Unused
--- &amp;#18; --- Unused
--- &amp;#19; --- Unused
--- &amp;#20; --- Unused
--- &amp;#21; --- Unused
--- &amp;#22; --- Unused
--- &amp;#23; --- Unused
--- &amp;#24; --- Unused
--- &amp;#25; --- Unused
--- &amp;#26; --- Unused
--- &amp;#27; --- Unused
--- &amp;#28; --- Unused
--- &amp;#29; --- Unused
--- &amp;#30; --- Unused
--- &amp;#31; --- Unused
--- &amp;#32; --- Space
! &amp;#33; --- Exclamation mark
" &amp;#34; &quot; Quotation mark
# &amp;#35; --- Number sign
$ &amp;#36; --- Dollar sign
% &amp;#37; --- Percent sign
&amp; &amp;#38; &amp;amp; Ampersand
' &amp;#39; --- Apostrophe
( &amp;#40; --- Left parenthesis
) &amp;#41; --- Right parenthesis
* &amp;#42; --- Asterisk
+ &amp;#43; --- Plus sign
, &amp;#44; --- Comma
-    &45;      ---        Hyphen
.    &46;      ---        Period (fullstop)
/    &47;      ---        Solidus (slash)
0    &48;      ---        Digit 0
1    &49;      ---        Digit 1
2    &50;      ---        Digit 2
3    &51;      ---        Digit 3
4    &52;      ---        Digit 4
5    &53;      ---        Digit 5
6    &54;      ---        Digit 6
7    &55;      ---        Digit 7
8    &56;      ---        Digit 8
9    &57;      ---        Digit 9
:    &58;      ---        Colon
;    &59;      ---        Semicolon
<    &60;      &lt;        Less than
=    &61;      ---        Equals sign
>    &62;      &gt;        Greater than
?    &63;      ---        Question mark
@    &64;      ---        Commercial at
A    &65;      ---        Capital A
B    &66;      ---        Capital B
C    &67;      ---        Capital C
D    &68;      ---        Capital D
E    &69;      ---        Capital E
F    &70;      ---        Capital F
G    &71;      ---        Capital G
H    &72;      ---        Capital H
I    &73;      ---        Capital I
J    &74;      ---        Capital J
K    &75;      ---        Capital K
L    &76;      ---        Capital L
M    &77;      ---        Capital M
<table>
<thead>
<tr>
<th>Character</th>
<th>ASCII Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N</td>
<td>Capital N</td>
</tr>
<tr>
<td>O</td>
<td>O</td>
<td>Capital O</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>Capital P</td>
</tr>
<tr>
<td>Q</td>
<td>Q</td>
<td>Capital Q</td>
</tr>
<tr>
<td>R</td>
<td>R</td>
<td>Capital R</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td>Capital S</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>Capital T</td>
</tr>
<tr>
<td>U</td>
<td>U</td>
<td>Capital U</td>
</tr>
<tr>
<td>V</td>
<td>V</td>
<td>Capital V</td>
</tr>
<tr>
<td>W</td>
<td>W</td>
<td>Capital W</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>Capital X</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>Capital Y</td>
</tr>
<tr>
<td>Z</td>
<td>Z</td>
<td>Capital Z</td>
</tr>
<tr>
<td>[</td>
<td>[</td>
<td>Left square bracket</td>
</tr>
<tr>
<td>\</td>
<td>\</td>
<td>Reverse solidus (backslash)</td>
</tr>
<tr>
<td>]</td>
<td>]</td>
<td>Right square bracket</td>
</tr>
<tr>
<td>^</td>
<td>^</td>
<td>Caret</td>
</tr>
<tr>
<td>_</td>
<td>_</td>
<td>Horizontal bar (underscore)</td>
</tr>
<tr>
<td>`</td>
<td>`</td>
<td>Grave accent</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>Small a</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>Small b</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td>Small c</td>
</tr>
<tr>
<td>d</td>
<td>d</td>
<td>Small d</td>
</tr>
<tr>
<td>e</td>
<td>e</td>
<td>Small e</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>Small f</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>Small g</td>
</tr>
<tr>
<td>h</td>
<td>h</td>
<td>Small h</td>
</tr>
<tr>
<td>i</td>
<td>i</td>
<td>Small i</td>
</tr>
<tr>
<td>j</td>
<td>j</td>
<td>Small j</td>
</tr>
<tr>
<td>k</td>
<td>k</td>
<td>Small k</td>
</tr>
<tr>
<td>l</td>
<td>l</td>
<td>Small l</td>
</tr>
<tr>
<td>m</td>
<td>m</td>
<td>Small m</td>
</tr>
<tr>
<td>n</td>
<td>n</td>
<td>Small n</td>
</tr>
<tr>
<td>Character</td>
<td>Hex Code</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>------------------------</td>
</tr>
<tr>
<td>o</td>
<td>o</td>
<td>Small o</td>
</tr>
<tr>
<td>p</td>
<td>p</td>
<td>Small p</td>
</tr>
<tr>
<td>q</td>
<td>q</td>
<td>Small q</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>Small r</td>
</tr>
<tr>
<td>s</td>
<td>s</td>
<td>Small s</td>
</tr>
<tr>
<td>t</td>
<td>t</td>
<td>Small t</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>Small u</td>
</tr>
<tr>
<td>v</td>
<td>v</td>
<td>Small v</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>Small w</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>Small x</td>
</tr>
<tr>
<td>y</td>
<td>y</td>
<td>Small y</td>
</tr>
<tr>
<td>z</td>
<td>z</td>
<td>Small z</td>
</tr>
<tr>
<td>{</td>
<td>{</td>
<td>Left curly brace</td>
</tr>
<tr>
<td></td>
<td></td>
<td>|</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
<td>Right curly brace</td>
</tr>
<tr>
<td>~</td>
<td>~</td>
<td>Tilde</td>
</tr>
<tr>
<td>---</td>
<td>�</td>
<td>Unused</td>
</tr>
<tr>
<td></td>
<td> </td>
<td>Nonbreaking space</td>
</tr>
<tr>
<td>i</td>
<td>¡</td>
<td>Inverted exclamation</td>
</tr>
<tr>
<td>¢</td>
<td>¢</td>
<td>Cent sign</td>
</tr>
<tr>
<td>£</td>
<td>£</td>
<td>Pound sterling</td>
</tr>
<tr>
<td>☺</td>
<td>¤</td>
<td>General currency sign</td>
</tr>
<tr>
<td>¥</td>
<td>¥</td>
<td>Yen sign</td>
</tr>
<tr>
<td></td>
<td>¦</td>
<td>Broken vertical bar</td>
</tr>
<tr>
<td>§</td>
<td>§</td>
<td>Section sign</td>
</tr>
<tr>
<td>..</td>
<td>¨</td>
<td>Diæresis / Umlaut</td>
</tr>
<tr>
<td>©</td>
<td>©</td>
<td>Copyright</td>
</tr>
<tr>
<td>a</td>
<td>ª</td>
<td>Feminine ordinal</td>
</tr>
<tr>
<td>«</td>
<td>«</td>
<td>Left angle quote, guillemot left</td>
</tr>
<tr>
<td></td>
<td>¬</td>
<td>Not sign</td>
</tr>
<tr>
<td>-</td>
<td>­</td>
<td>Soft hyphen</td>
</tr>
<tr>
<td>®</td>
<td>®</td>
<td>Registered trademark</td>
</tr>
<tr>
<td>Character</td>
<td>HTML Entity Code</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>—</td>
<td>¯</td>
<td>Macron accent</td>
</tr>
<tr>
<td>°</td>
<td>°</td>
<td>Degree sign</td>
</tr>
<tr>
<td>±</td>
<td>±</td>
<td>Plus or minus</td>
</tr>
<tr>
<td>²</td>
<td>²</td>
<td>Superscript two</td>
</tr>
<tr>
<td>³</td>
<td>³</td>
<td>Superscript three</td>
</tr>
<tr>
<td>′</td>
<td>´</td>
<td>Acute accent</td>
</tr>
<tr>
<td>µ</td>
<td>µ</td>
<td>Micro sign</td>
</tr>
<tr>
<td>¶</td>
<td></td>
<td>Paragraph sign</td>
</tr>
<tr>
<td>•</td>
<td>·</td>
<td>Middle dot</td>
</tr>
<tr>
<td>,</td>
<td>¸</td>
<td>Cedilla</td>
</tr>
<tr>
<td>¹</td>
<td>¹</td>
<td>Superscript one</td>
</tr>
<tr>
<td>²</td>
<td>º</td>
<td>Masculine ordinal</td>
</tr>
<tr>
<td>»</td>
<td>»</td>
<td>Right angle quote, guillemot right</td>
</tr>
<tr>
<td>¼</td>
<td>¼</td>
<td>Fraction one-fourth</td>
</tr>
<tr>
<td>½</td>
<td>½</td>
<td>Fraction one-half</td>
</tr>
<tr>
<td>¾</td>
<td>¾</td>
<td>Fraction three-fourths</td>
</tr>
<tr>
<td>‚</td>
<td></td>
<td>Inverted question mark</td>
</tr>
<tr>
<td>À</td>
<td>À</td>
<td>Capital A, grave accent</td>
</tr>
<tr>
<td>Á</td>
<td>Á</td>
<td>Capital A, acute accent</td>
</tr>
<tr>
<td>Â</td>
<td>Â</td>
<td>Capital A, circumflex</td>
</tr>
<tr>
<td>Á</td>
<td>Ã</td>
<td>Capital A, tilde</td>
</tr>
<tr>
<td>Ä</td>
<td>Ä</td>
<td>Capital A, diaeresis / umlaut</td>
</tr>
<tr>
<td>Å</td>
<td>Å</td>
<td>Capital A, ring</td>
</tr>
<tr>
<td>Æ</td>
<td>Æ</td>
<td>Capital AE ligature</td>
</tr>
<tr>
<td>Ç</td>
<td>Ç</td>
<td>Capital C, cedilla</td>
</tr>
<tr>
<td>È</td>
<td>È</td>
<td>Capital E, grave accent</td>
</tr>
<tr>
<td>É</td>
<td>É</td>
<td>Capital E, acute accent</td>
</tr>
<tr>
<td>Ê</td>
<td>Ê</td>
<td>Capital E, circumflex</td>
</tr>
<tr>
<td>Ë</td>
<td>Ë</td>
<td>Capital E, diaeresis / umlaut</td>
</tr>
<tr>
<td>Ï</td>
<td>Ì</td>
<td>Capital I, grave accent</td>
</tr>
<tr>
<td>Í</td>
<td>Í</td>
<td>Capital I, acute accent</td>
</tr>
<tr>
<td>Character</td>
<td>HTML Entity</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Í</td>
<td>Î</td>
<td>I, circumflex</td>
</tr>
<tr>
<td>Ï</td>
<td>Ï</td>
<td>I, diaeresis / umlaut</td>
</tr>
<tr>
<td>Ð</td>
<td>Ð</td>
<td>Eth, Icelandic</td>
</tr>
<tr>
<td>Ñ</td>
<td>Ñ</td>
<td>N, tilde</td>
</tr>
<tr>
<td>Ò</td>
<td>Ò</td>
<td>O, grave accent</td>
</tr>
<tr>
<td>Ó</td>
<td>Ó</td>
<td>O, acute accent</td>
</tr>
<tr>
<td>Ô</td>
<td>Ô</td>
<td>O, circumflex</td>
</tr>
<tr>
<td>Õ</td>
<td>Õ</td>
<td>O, tilde</td>
</tr>
<tr>
<td>Ö</td>
<td>Ö</td>
<td>O, diaeresis / umlaut</td>
</tr>
<tr>
<td>×</td>
<td>×</td>
<td>Multiply sign</td>
</tr>
<tr>
<td>Ø</td>
<td>Ø</td>
<td>O, slash</td>
</tr>
<tr>
<td>Ù</td>
<td>Ù</td>
<td>U, grave accent</td>
</tr>
<tr>
<td>Ú</td>
<td>Ú</td>
<td>U, acute accent</td>
</tr>
<tr>
<td>Õ</td>
<td>Û</td>
<td>U, circumflex</td>
</tr>
<tr>
<td>Ú</td>
<td>Ü</td>
<td>U, diaeresis / umlaut</td>
</tr>
<tr>
<td>Ý</td>
<td>Ý</td>
<td>Y, acute accent</td>
</tr>
<tr>
<td>ð</td>
<td>Þ</td>
<td>Thorn, Icelandic</td>
</tr>
<tr>
<td>ß</td>
<td>ß</td>
<td>Small sharp s, German sz</td>
</tr>
<tr>
<td>à</td>
<td>à</td>
<td>a, grave accent</td>
</tr>
<tr>
<td>á</td>
<td>á</td>
<td>a, acute accent</td>
</tr>
<tr>
<td>â</td>
<td>â</td>
<td>a, circumflex</td>
</tr>
<tr>
<td>ã</td>
<td>ã</td>
<td>a, tilde</td>
</tr>
<tr>
<td>ã</td>
<td>ä</td>
<td>a, diaeresis / umlaut</td>
</tr>
<tr>
<td>ã</td>
<td>å</td>
<td>a, ring</td>
</tr>
<tr>
<td>ãæ</td>
<td>æ</td>
<td>ae ligature</td>
</tr>
<tr>
<td>ç</td>
<td>ç</td>
<td>c, cedilla</td>
</tr>
<tr>
<td>è</td>
<td>è</td>
<td>e, grave accent</td>
</tr>
<tr>
<td>é</td>
<td>é</td>
<td>e, acute accent</td>
</tr>
<tr>
<td>è</td>
<td>ê</td>
<td>e, circumflex</td>
</tr>
<tr>
<td>ë</td>
<td>ë</td>
<td>e, diaeresis / umlaut</td>
</tr>
<tr>
<td>í</td>
<td>ì</td>
<td>i, grave accent</td>
</tr>
<tr>
<td>í</td>
<td>í</td>
<td>i, acute accent</td>
</tr>
<tr>
<td>ï</td>
<td>î</td>
<td>i, circumflex</td>
</tr>
</tbody>
</table>
Additional Named Entities for HTML

The following table contains additional named entities, their numeric character references, and a description of each. With the exception of the left and right-pointing brackets (&#9001; and &#9002;), these entities are rendered using Lucida sans Unicode.

<table>
<thead>
<tr>
<th>Character</th>
<th>Named entity</th>
<th>Numeric character reference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latin Extended-B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>ƒ</td>
<td>ƒ</td>
<td>Latin small f with hook, =function, =florin, U0192 ISOtech</td>
</tr>
<tr>
<td>Greek</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Α</td>
<td>Α</td>
<td>Greek capital letter alpha, U0391</td>
</tr>
</tbody>
</table>
Greek capital letter beta, U0392
Greek capital letter gamma, U0393 ISOgrk3
Greek capital letter delta, U0394 ISOgrk3
Greek capital letter epsilon, U0395
Greek capital letter zeta, U0396
Greek capital letter eta, U0397
Greek capital letter theta, U0398 ISOgrk3
Greek capital letter iota, U0399
Greek capital letter kappa, U039A
Greek capital letter lambda, U039B ISOgrk3
Greek capital letter mu, U039C
Greek capital letter nu, U039D
Greek capital letter xi, U039E ISOgrk3
Greek capital letter omicron, U039F
Greek capital letter pi, U03A0 ISOgrk3
Greek capital letter rho, U03A1
Greek capital letter sigma, U03A3 ISOgrk3
Greek capital letter tau, U03A4
Greek capital letter upsilon, U03A5 ISOgrk3
Greek capital letter phi, U03A6 ISOgrk3
Greek capital letter chi, U03A7
Greek capital letter psi, U03A8 ISOgrk3
Greek capital letter omega, U03A9 ISOgrk3
Greek small letter alpha, U03B1
<table>
<thead>
<tr>
<th>Greek Letter</th>
<th>HTML Entity</th>
<th>Unicode</th>
<th>ISOgrk3</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>α</td>
<td>U03B1</td>
<td>ISOgrk3</td>
</tr>
<tr>
<td>β</td>
<td>β</td>
<td>U03B2</td>
<td>ISOgrk3</td>
</tr>
<tr>
<td>γ</td>
<td>γ</td>
<td>U03B3</td>
<td>ISOgrk3</td>
</tr>
<tr>
<td>δ</td>
<td>δ</td>
<td>U03B4</td>
<td>ISOgrk3</td>
</tr>
<tr>
<td>ε</td>
<td>ε</td>
<td>U03B5</td>
<td>ISOgrk3</td>
</tr>
<tr>
<td>ζ</td>
<td>ζ</td>
<td>U03B6</td>
<td>ISOgrk3</td>
</tr>
<tr>
<td>η</td>
<td>η</td>
<td>U03B7</td>
<td>ISOgrk3</td>
</tr>
<tr>
<td>θ</td>
<td>θ</td>
<td>U03B8</td>
<td>ISOgrk3</td>
</tr>
<tr>
<td>ι</td>
<td>ι</td>
<td>U03B9</td>
<td>ISOgrk3</td>
</tr>
<tr>
<td>κ</td>
<td>κ</td>
<td>U03BA</td>
<td>ISOgrk3</td>
</tr>
<tr>
<td>λ</td>
<td>λ</td>
<td>U03BB</td>
<td>ISOgrk3</td>
</tr>
<tr>
<td>μ</td>
<td>μ</td>
<td>U03BC</td>
<td>ISOgrk3</td>
</tr>
<tr>
<td>ν</td>
<td>ν</td>
<td>U03BD</td>
<td>ISOgrk3</td>
</tr>
<tr>
<td>ξ</td>
<td>ξ</td>
<td>U03BE</td>
<td>ISOgrk3</td>
</tr>
<tr>
<td>ο</td>
<td>ο</td>
<td>U03BF</td>
<td>ISOgrk3</td>
</tr>
<tr>
<td>π</td>
<td>π</td>
<td>U03C0</td>
<td>ISOgrk3</td>
</tr>
<tr>
<td>ρ</td>
<td>ρ</td>
<td>U03C1</td>
<td>ISOgrk3</td>
</tr>
<tr>
<td>ζ</td>
<td>ς</td>
<td>U03C2</td>
<td>ISOgrk3</td>
</tr>
</tbody>
</table>
\( \sigma \) \&sigma; \&#963; Greek small letter sigma, U03C3 ISOgrk3

\( \tau \) \&tau; \&#964; Greek small letter tau, U03C4 ISOgrk3

\( \upsilon \) \&upsilon; \&#965; Greek small letter upsilon, U03C5 ISOgrk3

\( \phi \) \&phi; \&#966; Greek small letter phi, U03C6 ISOgrk3

\( \chi \) \&chi; \&#967; Greek small letter chi, U03C7 ISOgrk3

\( \psi \) \&psi; \&#968; Greek small letter psi, U03C8 ISOgrk3

\( \omega \) \&omega; \&#969; Greek small letter omega, U03C9 ISOgrk3

\( \theta \) \&thetasym; \&#977; Greek small letter theta symbol, U03D1 NEW

\( \Upsilon \) \&upsih; \&#978; Greek upsilon with hook symbol, U03D2 NEW

\( \pi \) \&piv; \&#982; Greek pi symbol, U03D6 ISOgrk3

**General Punctuation**

\( \bullet \) \&bull; \&#8226; bullet, =black small circle, U2022 ISOpub

\( \ldots \) \&hellip; \&#8230; horizontal ellipsis, =three dot leader, U2026 ISOpub

\( \prime \) \&prime; \&#8242; prime, =minutes, =feet, U2032 ISOtech

\( \prime \) \&Prime; \&#8243; double prime, =seconds, =inches, U2033 ISOtech

\( \overline{\ } \) \&oline; \&#8254; overline, =spacing overscore, U203E NEW

\( / \) \&frasl; \&#8260; fraction slash, U2044 NEW

**Letterlike Symbols**

\( \wp \) \&weierp; \&#8472; script capital P, =power set,
Arrows

\&larr; \#8592; leftward arrow, U2190 ISOnum
\&uarr; \#8593; upward arrow, U2191 ISOnum
\&rarr; \#8594; rightward arrow, U2192 ISOnum
\&darr; \#8595; downward arrow, U2193 ISOnum
\&harr; \#8596; left right arrow, U2194 ISOamsa
downward arrow with corner
\&crarr; \#8629; leftward, =carriage return, U21B5 NEW

\&lArr; \#8656; leftward double arrow, U21D0 ISOtech
\&uArr; \#8657; upward double arrow, U21D1 ISOamsa
\&rArr; \#8658; rightward double arrow, U21D2 ISOtech
\&dArr; \#8659; downward double arrow, U21D3 ISOamsa
\&hArr; \#8660; left right double arrow, U21D4 ISOamsa

Mathematical Operators

\&forall; \#8704; for all, U2200 ISOtech
\&part; \#8706; partial differential, U2202 ISOtech
\&exist; \#8707; there exists, U2203 ISOtech
\&empty; \#8709; empty set, =null set, =diameter, U2205 ISOamso
∇ &nabla;  &\#8711; nabla, =backward difference, U2207 ISOtech
ε &isin;  &\#8712; element of, U2208 ISOtech
∉ &notin;  &\#8713; not an element of, U2209 ISOtech
∃ &ni;  &\#8715; contains as member, U220B ISOtech
Π &prod;  &\#8719; n-ary product, =product sign, U220F ISOamsb
− &sum;  &\#8722; n-ary summation, U2211 ISOamsb
− &minus;  &\#8722; minus sign, U2212 ISOtech
∗ &lowast;  &\#8727; asterisk operator, U2217 ISOtech
√ &radic;  &\#8730; square root, =radical sign, U221A ISOtech
∝ &prop;  &\#8733; proportional to, U221D ISOtech
∞ &infin;  &\#8734; infinity, U221E ISOtech
∠ &ang;  &\#8736; angle, U2220 ISOamso
∧ &and;  &\#8869; logical and, =wedge, U2227 ISOtech
∨ &or;  &\#8870; logical or, =vee, U2228 ISOtech
∩ &cap;  &\#8745; intersection, =cap, U2229 ISOtech
∪ &cup;  &\#8746; union, =cup, U222A ISOtech
∫ &int;  &\#8747; integral, U222B ISOtech
∴ &there4;  &\#8756; therefore, U2234 ISOtech
∼ &sim;  &\#8764; tilde operator, =varies with, =similar to, U223C ISOtech
≡ &cong;  &\#8773; approximately equal to, U2245 ISOtech
≈ &asymp;  &\#8773; almost equal to, =asymptotic to, U2248 ISOamsr
≠ &ne;  &\#8800; not equal to, U2260 ISOtech
≡ &equiv;  &\#8801; identical to, U2261 ISOtech
≤ &le;  &\#8804; less-than or equal to, U2264 ISOtech
≥ greater-than or equal to, U2265 ISOtech
≥ &ge; &#8805; ISOtech

⊂ &sub; &#8834; subset of, U2282 ISOtech

⊃ &sup; &#8835; superset of, U2283 ISOtech

⊄ &nsub; &#8836; not a subset of, U2284 ISOamsn

⊆ &sube; &#8838; subset of or equal to, U2286 ISOtech

⊇ &supe; &#8839; superset of or equal to, U2287 ISOtech

⊕ &oplus; &#8853; circled plus, =direct sum, U2295 ISOamsb

⊗ &otimes; &#8855; circled times, =vector product, U2297 ISOamsb

⊥ &perp; &#8869; up tack, =orthogonal to, =perpendicular, U22A5 ISOtech

⋅ &sdot; &#8901; dot operator, U22C5 ISOamsb

Miscellaneous

Technical

⌈ &lceil; &#8968; left ceiling, =apl upstile, U2308, ISOamsc

⌉ &rceil; &#8969; right ceiling, U2309, ISOamsc

⌊ &lfloor; &#8970; left floor, =apl downstile, U230A, ISOamsc

⌋ &rfloor; &#8971; right floor, U230B, ISOamsc

⟨ &lang; &#9001; left-pointing angle bracket, =bra, U2329 ISOtech

⟩ &rang; &#9002; right-pointing angle bracket, =ket, U232A ISOtech

Geometric

Shapes

◊ &loz; &#9674; lozenge, U25CA ISOpub

Miscellaneous

Symbols

♠ &spades; &#9824; black spade suit, U2660 ISOpub

♣ &clubs; &#9827; black club suit, =shamrock, U2663
Character Entities for Special Symbols and BIDI Text

<table>
<thead>
<tr>
<th>Using NE</th>
<th>NE</th>
<th>NCR</th>
<th>Using NCR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C0 Controls and Basic Latin</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;</td>
<td>&amp;quot    &amp; #34;</td>
<td></td>
<td>quotation mark, =apl quote, U0022 ISOnum</td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp;amp     &amp; #38;</td>
<td></td>
<td>ampersand, U0026 ISOnum</td>
</tr>
<tr>
<td>&lt;</td>
<td>&amp;lt      &amp; #60;</td>
<td></td>
<td>less-than sign, U003C ISOnum</td>
</tr>
<tr>
<td>&gt;</td>
<td>&amp;gt      &amp; #62;</td>
<td></td>
<td>greater-than sign, U003E ISOnum</td>
</tr>
<tr>
<td><strong>Latin Extended-A</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ö</td>
<td>&amp;OElig   &amp; #338;</td>
<td></td>
<td>Latin capital ligature oe, U0152 ISOlat2</td>
</tr>
<tr>
<td>ö</td>
<td>&amp;oelig   &amp; #339;</td>
<td></td>
<td>Latin small ligature oe, U0153 ISOlat2</td>
</tr>
<tr>
<td>Š</td>
<td>&amp;Scaron  &amp; #352;</td>
<td></td>
<td>Latin capital letter s with caron, U0160 ISOlat2</td>
</tr>
<tr>
<td>š</td>
<td>&amp;scaron  &amp; #353;</td>
<td></td>
<td>Latin small letter s with caron, U0161 ISOlat2</td>
</tr>
<tr>
<td>Ŷ</td>
<td>&amp;Yuml    &amp; #376;</td>
<td></td>
<td>Latin capital letter y with diaeresis, U0178 ISOlat2</td>
</tr>
<tr>
<td><strong>Spacing Modifier Letters</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>^</td>
<td>&amp;circ    &amp; #710;</td>
<td></td>
<td>modifier letter circumflex accent, U02C6 ISOpub</td>
</tr>
<tr>
<td>~</td>
<td>&amp;tilde   &amp; #732;</td>
<td></td>
<td>small tilde, U02DC ISOdia</td>
</tr>
<tr>
<td><strong>General Punctuation</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&amp;ensp    &amp; #8194;</td>
<td></td>
<td>en space, U2002 ISOpub</td>
<td></td>
</tr>
<tr>
<td>&amp;emsp    &amp; #8195;</td>
<td></td>
<td>em space, U2003 ISOpub</td>
<td></td>
</tr>
<tr>
<td>&amp;thinsp  &amp; #8201;</td>
<td></td>
<td>thin space, U2009 ISOpub</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>zero width non-joiner, U200C NEW</td>
</tr>
</tbody>
</table>
Character Set Recognition

Microsoft Internet Explorer uses the character set specified for a document to determine how to translate the bytes in the document into characters on the screen or on paper. By default, Internet Explorer uses the character set specified in the HTTP content type returned by the server to determine this translation. If
this parameter is not given, Internet Explorer uses the character set specified by the META element in the document. It uses the user's preferences if no META element is given.

You can use the META element to explicitly set the character set for a document. In this case, you set the HTTP-EQUIV= attribute to "Content-Type" and specify a character set identifier in the CONTENT= attribute. For example, the following META element identifies windows-1251 as the character set for the document.

```html
<META HTTP-EQUIV="Content-Type"
    CONTENT="text/html; CHARSET=windows-1251">
```

As long as you place the META element before the BODY element, it affects the whole document, including the TITLE element. For clarity, it should appear as the first element after HEAD so that all readers know the encoding before the first element that can be displayed is parsed. Note that the META element applies to the document containing it. This means, for example, that a compound document (a document consisting of two or more documents in a set of frames) can use different character sets in different frames.

The following table contains information concerning the character sets supported by Internet Explorer 5. The information provided is:

1. Display Name— the name used to refer to the character set.
2. Preferred Charset ID— the most common identifier used to set character sets in Internet Explorer. For example, in the previous code sample windows-1251 is the Charset ID.
3. Additional Aliases— other identifiers that may be used to set character sets.
4. MLang Code Pages— numeric value of the code pages used by the Internet Explorer MLang API.
5. Supported by Version— the versions of Internet Explorer that support the listed character sets.

6. **Note** CS indicates that the version of Internet Explorer must support complex scripts such as Arabic, Hebrew, or Thai.
#Charsets in Microsoft Internet Explorer 5#

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Preferred Charset ID</th>
<th>Additional Aliases</th>
<th>MLang Code</th>
<th>Supported by Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>ASMO-708</td>
<td></td>
<td>708</td>
<td>4CS, 5</td>
</tr>
<tr>
<td>Arabic (DOS)</td>
<td>DOS-720</td>
<td></td>
<td>720</td>
<td>4CS, 5</td>
</tr>
<tr>
<td>Arabic (Windows)</td>
<td>windows-1256</td>
<td></td>
<td>1256</td>
<td>4CS, 5</td>
</tr>
<tr>
<td>Baltic (Windows)</td>
<td>Windows-1257</td>
<td></td>
<td>1257</td>
<td>4, 5</td>
</tr>
<tr>
<td>Central European</td>
<td>ibm852</td>
<td>cp852</td>
<td>852</td>
<td>4, 5</td>
</tr>
<tr>
<td>Central European</td>
<td>windows-1250</td>
<td>x-cp1250</td>
<td>1250</td>
<td>3, 4, 5</td>
</tr>
<tr>
<td>Chinese Simplified</td>
<td>gb2312</td>
<td>chinese, csGB2312, csISO58GB23128, GB2312, GBK, GB_2312-80, iso-ir-58</td>
<td>936</td>
<td>3, 4, 5</td>
</tr>
<tr>
<td>Chinese Simplified</td>
<td>hz-gb-2312</td>
<td></td>
<td>52936</td>
<td>4, 5</td>
</tr>
<tr>
<td>Language</td>
<td>Encoding</td>
<td>Description</td>
<td>Code</td>
<td>Version(s)</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------</td>
<td>------------</td>
</tr>
<tr>
<td>Chinese Traditional</td>
<td>big5</td>
<td>csbig5, x-x-big5</td>
<td>950</td>
<td>3, 4, 5</td>
</tr>
<tr>
<td>Cyrillic (DOS)</td>
<td>cp866</td>
<td>ibm866</td>
<td>866</td>
<td>4, 5</td>
</tr>
<tr>
<td>Cyrillic (KOI8-R)</td>
<td>koi8-r</td>
<td>csKOI8R, koi</td>
<td>20866</td>
<td>3, 4, 5</td>
</tr>
<tr>
<td>Cyrillic (Windows)</td>
<td>windows-1251</td>
<td>x-cp1251</td>
<td>1251</td>
<td>3, 4, 5</td>
</tr>
<tr>
<td>Greek (ISO)</td>
<td>iso-8859-7</td>
<td>csISOLatinGreek, ECMA-118, ELOT_928, greek, greek8, iso-ir-126, ISO_8859-7, ISO_8859-7:1987</td>
<td>28597</td>
<td>3, 4, 5</td>
</tr>
<tr>
<td>Greek (Windows)</td>
<td>Windows-1253</td>
<td>windows-1253</td>
<td>1253</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>csEUCPkdFmtJapanese,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Encoding</td>
<td>Description</td>
<td>Version</td>
<td>Sections</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>Japanese (Shift-JIS)</td>
<td>shift_jis</td>
<td></td>
<td>932</td>
<td>3, 4, 5</td>
</tr>
<tr>
<td>Korean</td>
<td>ks_c_5601-1987</td>
<td>csKSC56011987, euc-kr, korean, ks_c_5601</td>
<td>949</td>
<td>3, 4, 5</td>
</tr>
<tr>
<td>Latin 3 (ISO)</td>
<td>iso-8859-3</td>
<td></td>
<td>28593</td>
<td>4, 5</td>
</tr>
<tr>
<td>Thai (Windows)</td>
<td>iso-8859-11</td>
<td>windows-874</td>
<td>874</td>
<td>3, 4, 5</td>
</tr>
<tr>
<td>Ukrainian (KOI8-U)</td>
<td>koi8-u</td>
<td></td>
<td>21866</td>
<td>4, 5</td>
</tr>
<tr>
<td>Unicode (UTF-7)</td>
<td>utf-7</td>
<td>csUnicode11UTF7, unicode-1-1-utf-7, x-unicode-2-0-utf-7</td>
<td>65000</td>
<td>4, 5</td>
</tr>
<tr>
<td>Unicode (UTF-8)</td>
<td>utf-8</td>
<td>unicode-1-1-utf-8, unicode-2-0-utf-8, x-unicode-2-0-utf-8</td>
<td>65001</td>
<td>4, 5</td>
</tr>
<tr>
<td>Vietnamese (Windows)</td>
<td>windows-1258</td>
<td></td>
<td>1258</td>
<td>3, 4, 5</td>
</tr>
<tr>
<td>Western European (Windows)</td>
<td>Windows-1252</td>
<td></td>
<td>1252</td>
<td>5</td>
</tr>
</tbody>
</table>
Nonstandard Charsets with Special Meaning Inside Internet Explorer and MLang

These character sets are not to be used for labeling documents.

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Preferred Charset ID</th>
<th>Additional Aliases</th>
<th>MLang Code</th>
<th>Supported by Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Japanese (Auto Select)</td>
<td>_autodetect</td>
<td></td>
<td>50932</td>
<td>3, 4, 5</td>
</tr>
<tr>
<td>Korean (Auto Select)</td>
<td>_autodetect_kr</td>
<td></td>
<td>50949</td>
<td>4, 5</td>
</tr>
<tr>
<td>Unicode</td>
<td>unicode</td>
<td></td>
<td>1200</td>
<td>4, 5</td>
</tr>
<tr>
<td>Unicode (BigEndian)</td>
<td>unicodeFEFF</td>
<td></td>
<td>1201</td>
<td>4, 5</td>
</tr>
<tr>
<td>User Defined</td>
<td>x-user-defined</td>
<td></td>
<td>50000</td>
<td>4, 5</td>
</tr>
</tbody>
</table>