

[MS-HTML401E]:

Internet Explorer Extensions to HTML 4.01 and DOM Level 2 HTML Specifications

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft [Open Specification Promise](#) or the [Community Promise](#). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
03/26/2010	1.0	New	Released new document.
04/16/2010	1.1	Major	Significantly changed the technical content.
05/26/2010	1.2	Major	Significantly changed the technical content.
06/29/2010	1.21	Editorial	Changed language and formatting in the technical content.
09/08/2010	1.3	Major	Significantly changed the technical content.
10/13/2010	1.4	Minor	Clarified the meaning of the technical content.
02/10/2011	2.0	No change	Introduced no new technical or language changes.
02/22/2012	3.0	Major	Significantly changed the technical content.
07/25/2012	3.1	Minor	Clarified the meaning of the technical content.

Table of Contents

1	Introduction	8
1.1	Glossary	8
1.2	References.....	8
1.2.1	Normative References.....	8
1.2.2	Informative References	9
1.3	Extension Overview (Synopsis).....	9
1.3.1	Element Behaviors and HTML Components	16
1.3.2	Organization of This Documentation	18
1.4	Relationship to Standards and Other Extensions	18
1.5	Applicability Statement.....	18
2	Extensions.....	20
2.1	Elements	20
2.1.1	?IMPORT?.....	20
2.1.2	COMMENT	21
2.1.3	MARQUEE.....	21
2.1.4	NOBR.....	22
2.1.5	PUBLIC:ATTACH.....	22
2.1.6	PUBLIC:COMPONENT	22
2.1.7	PUBLIC:DEFAULTS	23
2.1.8	PUBLIC:EVENT	23
2.1.9	PUBLIC:METHOD	23
2.1.10	PUBLIC:PROPERTY.....	24
2.2	Interfaces	24
2.2.1	Internet Explorer-Only Interfaces	26
2.2.1.1	HTCComponentElement.....	26
2.2.1.2	HTCElementBehaviorDefaults	26
2.2.1.3	HTCPropertyElement	26
2.2.1.4	HTMLBlockElement	26
2.2.1.5	HTMLCommentElement	27
2.2.1.6	HTMLDDElement.....	28
2.2.1.7	HTMLDTElement	28
2.2.1.8	HTMLMarqueeElement	28
2.2.2	DOM Interfaces	29
2.2.2.1	HTMLAnchorElement	29
2.2.2.2	HTMLAreaElement	30
2.2.2.3	HTMLAreasCollection	31
2.2.2.4	HTMLBodyElement.....	31
2.2.2.5	HTMLButtonElement.....	32
2.2.2.6	HTMLCollection.....	32
2.2.2.7	HTMLDivElement	33
2.2.2.8	HTMLDocument	33
2.2.2.9	HTMLElement.....	34
2.2.2.10	HTMLFieldSetElement	35
2.2.2.11	HTMLFormElement.....	36
2.2.2.12	HTMLFrameElement	36
2.2.2.13	HTMLFrameSetElement.....	37
2.2.2.14	HTMLHeadingElement.....	37
2.2.2.15	HTMLHRElement	38
2.2.2.16	HTMLIFrameElement	38

2.2.2.17	HTMLImageElement	39
2.2.2.18	HTMLInputElement.....	40
2.2.2.19	HTMLIsIndexElement.....	41
2.2.2.20	HTMLLabelElement.....	41
2.2.2.21	HTMLLegendElement	41
2.2.2.22	HTMLLinkElement	42
2.2.2.23	HTMLMetaElement	42
2.2.2.24	HTMLObjectElement	42
2.2.2.25	HTMLOptionElement.....	43
2.2.2.26	HTMLParagraphElement.....	43
2.2.2.27	HTMLSelectElement.....	44
2.2.2.28	HTMLSpanElement	44
2.2.2.29	HTMLStyleElement	44
2.2.2.30	HTMLTableCaptionElement.....	45
2.2.2.31	HTMLTableCellElement	45
2.2.2.32	HTMLTableElement.....	45
2.2.2.33	HTMLTableRowElement.....	46
2.2.2.34	HTMLTableSectionElement	47
2.2.2.35	HTMLTextAreaElement.....	47
2.3	Attributes	48
2.3.1	DOM and Content Attributes	53
2.3.1.1	action.....	53
2.3.1.2	align	53
2.3.1.3	allowTransparency	54
2.3.1.4	alt	54
2.3.1.5	background.....	54
2.3.1.6	BaseHref	55
2.3.1.7	behavior	55
2.3.1.8	bgColor	55
2.3.1.9	bgProperties	56
2.3.1.10	border	56
2.3.1.11	borderColor.....	56
2.3.1.12	borderColorDark	57
2.3.1.13	borderColorLight.....	57
2.3.1.14	bottomMargin.....	57
2.3.1.15	canHaveHTML.....	58
2.3.1.16	charset	58
2.3.1.17	classid	58
2.3.1.18	clear	58
2.3.1.19	color	59
2.3.1.20	cols.....	59
2.3.1.21	dataFld	59
2.3.1.22	dataFormatAs.....	60
2.3.1.23	dataPageSize.....	60
2.3.1.24	dataSrc	61
2.3.1.25	direction	61
2.3.1.26	dynsrc	62
2.3.1.27	frameBorder.....	62
2.3.1.28	frameSpacing	62
2.3.1.29	height	63
2.3.1.30	href.....	63
2.3.1.31	hspace	64
2.3.1.32	leftMargin	64

2.3.1.33	loop	64
2.3.1.34	lowsrc	65
2.3.1.35	Methods	65
2.3.1.36	name	65
2.3.1.37	noResize	66
2.3.1.38	noWrap	66
2.3.1.39	rightMargin	66
2.3.1.40	scroll	67
2.3.1.41	scrollAmount	67
2.3.1.42	scrollDelay	67
2.3.1.43	start.....	67
2.3.1.44	tabStop	68
2.3.1.45	topMargin	68
2.3.1.46	trueSpeed	68
2.3.1.47	urn	68
2.3.1.48	vAlign.....	69
2.3.1.49	value.....	69
2.3.1.50	viewInheritStyle	69
2.3.1.51	viewMasterTab	70
2.3.1.52	vspace.....	70
2.3.1.53	width.....	70
2.3.1.54	wrap	71
2.3.2	DOM Attributes Only	71
2.3.2.1	alinkColor	71
2.3.2.2	altHtml.....	71
2.3.2.3	contentWindow	71
2.3.2.4	dir	72
2.3.2.5	encoding	72
2.3.2.6	fgColor.....	72
2.3.2.7	fileCreatedDate	73
2.3.2.8	fileModifiedDate.....	73
2.3.2.9	fileSize	73
2.3.2.10	fileUpdatedDate	73
2.3.2.11	hash.....	73
2.3.2.12	host	74
2.3.2.13	hostname	74
2.3.2.14	indeterminate.....	74
2.3.2.15	isDisabled	75
2.3.2.16	isMultiLine.....	75
2.3.2.17	linkColor	75
2.3.2.18	contentType.....	75
2.3.2.19	nameProp	75
2.3.2.20	object.....	76
2.3.2.21	parentDocument.....	76
2.3.2.22	pathname	76
2.3.2.23	port	76
2.3.2.24	protocol	77
2.3.2.25	protocolLong	77
2.3.2.26	search	77
2.3.2.27	status.....	78
2.3.2.28	styleSheet.....	78
2.3.2.29	uniqueID	78
2.3.2.30	uniqueNumber.....	79

2.3.2.31	url	80
2.3.2.32	viewLink	80
2.3.2.33	vlinkColor	81
2.3.3	Content Attributes Only	81
2.3.3.1	application	81
2.3.3.2	atomicselection	81
2.3.3.3	event	82
2.3.3.4	for	82
2.3.3.5	get	82
2.3.3.6	implementation	82
2.3.3.7	internalName	83
2.3.3.8	lightWeight	83
2.3.3.9	literalContent	83
2.3.3.10	namespace	84
2.3.3.11	onevent	84
2.3.3.12	persist	84
2.3.3.13	put	85
2.3.3.14	supportsEditMode	85
2.3.3.15	tagName	85
2.3.3.16	unselectable	85
2.3.3.17	viewLinkContent	85
2.3.3.18	xmlns	86
2.3.4	Event Attributes	86
2.3.4.1	onactivate	86
2.3.4.2	onafterupdate	86
2.3.4.3	onbeforeactivate	87
2.3.4.4	onbeforecopy	87
2.3.4.5	onbeforecut	87
2.3.4.6	onbeforedeactivate	88
2.3.4.7	onbeforeeditfocus	88
2.3.4.8	onbeforepaste	88
2.3.4.9	onbeforeunload	89
2.3.4.10	onbeforeupdate	89
2.3.4.11	onbounce	90
2.3.4.12	oncellchange	90
2.3.4.13	oncontentready	90
2.3.4.14	oncontentsave	91
2.3.4.15	oncontrolselect	91
2.3.4.16	oncopy	91
2.3.4.17	oncut	91
2.3.4.18	ondataavailable	92
2.3.4.19	ondatasetchanged	92
2.3.4.20	ondatasetcomplete	92
2.3.4.21	ondeactivate	93
2.3.4.22	ondetach	93
2.3.4.23	ondocumentready	93
2.3.4.24	onerrorupdate	93
2.3.4.25	onfilterchange	94
2.3.4.26	onfinish	94
2.3.4.27	onfocusin	94
2.3.4.28	onfocusout	95
2.3.4.29	onhelp	95
2.3.4.30	onlosecapture	95

2.3.4.31	onmouseenter	95
2.3.4.32	onmouseleave	96
2.3.4.33	onmove	96
2.3.4.34	onmoveend	96
2.3.4.35	onmovestart.....	97
2.3.4.36	onpaste	97
2.3.4.37	onpropertychange	97
2.3.4.38	onresizeend	98
2.3.4.39	onresizestart	98
2.3.4.40	onrowenter	98
2.3.4.41	onrowexit	98
2.3.4.42	onrowsdelete.....	99
2.3.4.43	onrowsinserted	99
2.3.4.44	onselect.....	99
2.3.4.45	onselectstart	100
2.3.4.46	onstart	100
2.4	Methods	100
2.4.1	add	101
2.4.2	item	101
2.4.3	moveRow	102
2.4.4	namedItem	102
2.4.5	remove	103
2.4.6	setActive	103
2.4.7	tags.....	103
2.4.8	urns	104
2.5	Collections	104
2.5.1	all	104
2.5.2	cells.....	104
2.5.3	frames	104
3	Security Considerations.....	105
4	Appendix A: Product Behavior.....	106
5	Change Tracking.....	107
6	Index	109

1 Introduction

This document describes extensions provided by Windows® Internet Explorer® for the *HTML Specification* [HTML], W3C Recommendation 24 December 1999, and *Document Object Model (DOM) Level 2 HTML Specification Version 1.0* [DOM Level 2 - HTML], W3C Recommendation 09 January, 2003.

Sections 1.7 and 2 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. All other sections and examples in this specification are informative.

1.1 Glossary

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the technical documents, which are updated frequently. References to other documents include a publishing year when one is available.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, <http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624>, as an additional source.

[CSS-Level2-2009] Bos, B., Celik, T., Hickson, I., and Wium Lie, H., Eds., "Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification", W3C Candidate Recommendation 08 September 2009, <http://www.w3.org/TR/2009/CR-CSS2-20090908/>

[DOM Level 2 - Core] W3C, "Document Object Model (DOM) Level 2 Core Specification Version 1.0", W3C Recommendation 13 November, 2000, <http://www.w3.org/TR/DOM-Level-2-Core/>

[DOM Level 2 - HTML] W3C, "Document Object Model (DOM) Level 2 HTML Specification Version 1.0", W3C Recommendation, January 2003, <http://www.w3.org/TR/2003/REC-DOM-Level-2-HTML-20030109/>

[HTML] World Wide Web Consortium, "HTML 4.01 Specification", December 1999, <http://www.w3.org/TR/html4/>

[MS-DOM2CE] Microsoft Corporation, "[Internet Explorer Extensions to the Document Object Model \(DOM\) Level 2 Core Specification](#)".

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.rfc-editor.org/rfc/rfc2119.txt>

1.2.2 Informative References

[DOM Level 2 - Style] W3C, "Document Object Model (DOM) Level 2 Style Specification Version 1.0", W3C Recommendation, November 2000, <http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113/>

[ECMA-262/5] ECMA International, "Standard ECMA-262 ECMAScript Language Specification", 5th Edition (December 2009), <http://www.ecma-international.org/publications/files/ECMA-ST-ARCH/ECMA-262%205th%20edition%20December%202009.pdf>

[ECMA-262] ECMA International, "ECMAScript Language Specification" ECMA-262, December 1999, <http://www.ecma-international.org/publications/standards/Ecma-262.htm>

[MS-CSS21E] Microsoft Corporation, "[Internet Explorer Extensions to the Cascading Style Sheets \(CSS\) 2.1 Specification](#)".

[MSDN-DHTMLBehaviors] Microsoft Corporation, "Introduction to DHTML Behaviors", [http://msdn.microsoft.com/en-us/library/ms531079\(v=VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms531079(v=VS.85).aspx)

[MS-DOM2CEX] Microsoft Corporation, "[Microsoft XML Extensions to the Document Object Model \(DOM\) Level 2 Core Specification](#)".

[MS-DOM2E] Microsoft Corporation, "[Internet Explorer Document Object Model \(DOM\) Level 2 Events Standards Support Document](#)".

[MS-DOM2EE] Microsoft Corporation, "[Internet Explorer Extensions to the DOM Level 2 Events Specification](#)".

[MS-ES3EX] Microsoft Corporation, "[Microsoft JScript Extensions to the ECMAScript Language Specification 3rd Edition](#)".

[MS-ES5EX] Microsoft Corporation, "[Microsoft Internet Explorer Extensions to the ECMAScript Language Specification Fifth Edition](#)".

1.3 Extension Overview (Synopsis)

The extensions described in this document were selected for their applicability to [\[HTML\]](#) and [\[DOM Level 2 - HTML\]](#).

The extensions to [\[HTML\]](#) and [\[DOM Level 2 - HTML\]](#) are organized as follows:

Elements

- [?IMPORT?](#)
- [COMMENT](#)
- [MARQUEE](#)
- [NOBR](#)
- [PUBLIC:ATTACH](#)
- [PUBLIC:COMPONENT](#)
- [PUBLIC:DEFAULTS](#)
- [PUBLIC:EVENT](#)

- [PUBLIC:METHOD](#)
- [PUBLIC:PROPERTY](#)

Interfaces

- [HTCComponentElement](#)
- [HTCElementBehaviorDefaults](#)
- [HTCPropertyElement](#)
- [HTMLAnchorElement](#)
- [HTMLAreaElement](#)
- [HTMLAreasCollection](#)
- [HTMLBlockElement](#)
- [HTMLBodyElement](#)
- [HTMLButtonElement](#)
- [HTMLCollection](#)
- [HTMLDDElement](#)
- [HTMLDivElement](#)
- [HTMLDocument](#)
- [HTMLDTElement](#)
- [HTMLElement](#)
- [HTMLFieldSetElement](#)
- [HTMLFormElement](#)
- [HTMLFrameElement](#)
- [HTMLFrameSetElement](#)
- [HTMLHeadingElement](#)
- [HTMLHRElement](#)
- [HTMLIFrameElement](#)
- [HTMLImageElement](#)
- [HTMLInputElement](#)
- [HTMLIsIndexElement](#)
- [HTMLLabelElement](#)
- [HTMLLegendElement](#)

- [HTMLMarqueeElement](#)
- [HTMLMetaElement](#)
- [HTMLObjectElement](#)
- [HTMLOptionElement](#)
- [HTMLParagraphElement](#)
- [HTMLSelectElement](#)
- [HTMLSpanElement](#)
- [HTMLStyleElement](#)
- [HTMLTableCaptionElement](#)
- [HTMLTableCellElement](#)
- [HTMLTableElement](#)
- [HTMLTableRowElement](#)
- [HTMLTableSectionElement](#)
- [HTMLTextAreaElement](#)

The following attributes are listed alphabetically, but indicate their type: (DOM) for DOM attribute only, and (markup) for content attribute only.

Attributes

- [action](#)
- [align](#)
- [alinkColor](#) (DOM)
- [allowTransparency](#)
- [alt](#)
- [altHtml](#) (DOM)
- [application](#) (markup)
- [atomicselection](#) (markup)
- [background](#)
- [BaseHref](#)
- [behavior](#)
- [bgColor](#)
- [bgProperties](#)
- [border](#)

- [borderColor](#)
- [borderColorDark](#)
- [borderColorLight](#)
- [bottomMargin](#)
- [canHaveHTML](#)
- [charset](#)
- [classid](#)
- [clear](#)
- [color](#)
- [cols](#)
- [contentWindow](#) (DOM)
- [dataFld](#)
- [dataFormatAs](#)
- [dataPageSize](#)
- [dataSrc](#)
- [dir](#) (DOM)
- [direction](#)
- [dynsrc](#)
- [encoding](#)
- [event](#) (markup)
- [fgColor](#) (DOM)
- [fileCreatedDate](#) (DOM)
- [fileModifiedDate](#) (DOM)
- [fileSize](#) (DOM)
- [fileUpdatedDate](#) (DOM)
- [for](#) (markup)
- [frameBorder](#)
- [frameSpacing](#)
- [get](#) (markup)
- [hash](#) (DOM)

- [height](#)
- [host](#) (DOM)
- [hostname](#) (DOM)
- [href](#)
- [hspace](#)
- [implementation](#) (markup)
- [indeterminate](#) (DOM)
- [internalName](#) (markup)
- [isDisabled](#) (DOM)
- [isMultiLine](#) (DOM)
- [leftMargin](#)
- [lightWeight](#) (markup)
- [linkColor](#) (DOM)
- [literalContent](#) (markup)
- [loop](#)
- [lowsrc](#)
- [methods](#)
- [mimeType](#) (DOM)
- [name](#)
- [nameProp](#) (DOM)
- [namespace](#) (markup)
- [noResize](#)
- [noWrap](#)
- [object](#) (DOM)
- [onactivate](#)
- [onafterupdate](#)
- [onbeforeactivate](#)
- [onbeforecopy](#)
- [onbeforecut](#)
- [onbeforedeactivate](#)

- [onbeforeeditfocus](#)
- [onbeforepaste](#)
- [onbeforeunload](#)
- [onbeforeupdate](#)
- [onbounce](#)
- [oncellchange](#)
- [oncontrolselect](#)
- [oncopy](#)
- [oncut](#)
- [ondataavailable](#)
- [ondatasetchanged](#)
- [ondatasetcomplete](#)
- [ondeactivate](#)
- [onerrorupdate](#)
- [onevent](#) (markup)
- [onfilterchange](#)
- [onfinish](#)
- [onfocusin](#)
- [onfocusout](#)
- [onhelp](#)
- [onlosecapture](#)
- [onmouseenter](#)
- [onmouseleave](#)
- [onmove](#)
- [onmoveend](#)
- [onmovestart](#)
- [onpaste](#)
- [onpropertychange](#)
- [onresizeend](#)
- [onresizestart](#)

- [onrowenter](#)
- [onrowexit](#)
- [onrowsdelete](#)
- [onrowsinserted](#)
- [onselect](#)
- [onselectstart](#)
- [onstart](#)
- [parentDocument](#) (DOM)
- [pathname](#) (DOM)
- [persist](#) (markup)
- [port](#) (DOM)
- [protocol](#) (DOM)
- [protocolLong](#) (DOM)
- [put](#) (markup)
- [rightMargin](#)
- [scroll](#)
- [scrollAmount](#)
- [scrollDelay](#)
- [search](#) (DOM)
- [start](#)
- [status](#) (DOM)
- [styleSheet](#) (DOM)
- [supportsEditMode](#) (markup)
- [tabStop](#)
- [tagName](#) (markup)
- [topMargin](#)
- [trueSpeed](#)
- [uniqueID](#) (DOM)
- [uniqueNumber](#) (DOM)
- [unselectable](#) (markup)

- [url](#) (DOM)
- [urn](#)
- [vAlign](#)
- [value](#)
- [viewInheritStyle](#)
- [viewLink](#)
- [viewLinkContent](#) (markup)
- [viewMasterTab](#)
- [vlinkColor](#) (DOM)
- [vspace](#)
- [width](#)
- [wrap](#)
- [xmlns](#) (markup)

Methods

- [add\(\)](#)
- [item\(\)](#)
- [moveRow\(\)](#)
- [namedItem\(\)](#)
- [remove\(\)](#)
- [setActive\(\)](#)
- [tags\(\)](#)
- [urns\(\)](#)

Collections

- [all](#)
- [cells](#)
- [frames](#)

1.3.1 Element Behaviors and HTML Components

Element behaviors allow a developer to define custom elements that can be used in the same way as normal HTML elements in a webpage. Because element behaviors are encapsulated components, they can add functionality to a webpage while improving the reusability and organization of content, script, and style elements.

Element behaviors are synchronously attached to elements through special style sheet rules, or an IMPORT processing instruction such as the following:

```
<HTML xmlns:games>
<HEAD>
<?IMPORT namespace="games" implementation="checkers.htc" ?>
</HEAD>
<BODY>
<games:checkers />
</BODY>
</HTML>
```

Element behaviors, which are used to define custom elements, are also called HTML Components (HTC). The implementation of the HTC is encapsulated in an .htc file.

The following elements and attributes are used to define the public interface and event model of an HTC:

- [PUBLIC:ATTACH](#)
 - [event](#)
 - [for](#)
 - [onevent](#)
- [PUBLIC:COMPONENT](#)
 - [lightWeight](#)
 - [literalContent](#)
 - [supportsEditMode](#)
 - [tagName](#)
 - [urn](#)
- [PUBLIC:DEFAULTS](#)
 - [canHaveHTML](#)
 - [tabStop](#)
 - [viewInheritStyle](#)
 - [viewLinkContent](#)
 - [viewMasterTab](#)
- [PUBLIC:EVENT](#)
- [PUBLIC:METHOD](#)
 - [internalName](#)
- [PUBLIC:PROPERTY](#)

- [get](#)
- `internalName`
- [persist](#)
- [put](#)
- [value](#)

For more information about Element Behaviors and HTC, see *Introduction to DHTML Behaviors* [[MSDN-DHTMLBehaviors](#)].

1.3.2 Organization of This Documentation

This document is organized as follows:

- **Elements:** New HTML elements are described.
- **Interfaces:** The extensions are listed according to interface at the highest level.

Interface members are described at the next levels.

- **Attributes:** Includes all markup and DOM properties.
 - **DOM and Content Attribute:** Attributes that are supported by both DOM and markup.
 - **DOM Attributes Only:** Attributes only available through the object model.
 - **Content Attributes Only:** Attributes declared only in markup.
 - **Event Attributes:** Attributes that support events and event handlers.
- **Methods:** Extends DOM functionality.
- **Collections:** Objects that support array-type behavior.

1.4 Relationship to Standards and Other Extensions

The following documents provide additional extensions.

- [[MS-CSS21E](#)]: Extensions to the [[CSS-Level2-2009](#)] and [[DOM Level 2 - Style](#)] specifications.
- [[MS-DOM2CE](#)] and [[MS-DOM2CEX](#)]: Extensions to the [[DOM Level 2 - Core](#)] specification.
- [[MS-DOM2EE](#)]: Extensions to the [[MS-DOM2E](#)] specification.
- [[MS-ES3EX](#)]: Extensions to the ECMAScript [[ECMA-262](#)] specification.
- [[MS-ES5EX](#)]: Extensions to the ECMAScript [[ECMA-262/5](#)] specification.

1.5 Applicability Statement

This document specifies a set of extensions to the [[HTML](#)] and [[DOM Level 2 - HTML](#)] specifications that are unique to Windows® Internet Explorer® 7, Windows® Internet Explorer® 8, Windows® Internet Explorer® 9, and Windows® Internet Explorer® 10.

In addition, each version of Windows® Internet Explorer® implements multiple document modes, which can vary individually in their support of the standard. The following table lists the document modes available in each version of Internet Explorer.

Browser Version	Document Modes Supported
Internet Explorer 7	Quirks mode Standards mode
Internet Explorer 8	Quirks mode IE7 mode IE8 mode
Internet Explorer 9	Quirks mode IE7 mode IE8 mode IE9 mode
Internet Explorer 10	Quirks mode IE7 mode IE8 mode IE9 mode IE10 Mode

Throughout this document, the document mode appears first followed by the browser version in parentheses. Only those document modes and versions of Internet Explorer for which there is an extension will be listed. If the document mode is not listed, it can be assumed that all document modes support the extension.

2 Extensions

This section specifies additional elements, interfaces, attributes, methods and collections to [\[HTML\]](#) and [\[DOM Level 2 - HTML\]](#) that are available in Windows® Internet Explorer®.

The extensions to [\[HTML\]](#) and [\[DOM Level 2 - HTML\]](#) are as follows:

- Extensions to the [Elements](#)
- Extensions to the [Interfaces](#)
- Extensions to the [Attributes](#)
- Extensions to the [Methods](#)
- Extensions to the [Collections](#)

2.1 Elements

The following elements are extensions to [\[HTML\]](#):

- [?IMPORT?](#)
- [comment](#) ([HTMLCommentElement](#))
- [marquee](#) ([HTMLMarqueeElement](#))
- [noBR](#)
- [PUBLIC:ATTACH](#)
- [PUBLIC:COMPONENT](#)
- [PUBLIC:DEFAULTS](#)
- [PUBLIC:EVENT](#)
- [PUBLIC:METHOD](#)
- [PUBLIC:PROPERTY](#)

2.1.1 ?IMPORT?

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

A processing instruction that imports a tag definition from an element behavior. Two attributes are required:

- [namespace](#)
- [implementation](#)

The value of an XMLNS attribute that has been defined in the HTML tag must correspond to the value of the **namespace** attribute defined in the **IMPORT** processing instruction (PI). Otherwise, the imported tag definition cannot be used.

An imported tag will not render if the value of the implementation attribute is not valid. When using an HTC file, the file must comply with the same security rules as all behaviors. For more information, see *Introduction to DHTML Behaviors* [\[MSDN-DHTMLBehaviors\]](#).

The **IMPORT** PI must be placed before the first instance of a custom element that uses the imported tag definition. If the **IMPORT** PI is placed after the custom element, the behavior does not attach to the custom element. This also means that the **document.write** method should not be used to add the **IMPORT** PI to a document.

The **IMPORT** PI is only processed during the initial parsing of the document. Therefore, a document that uses an element behavior must include the **IMPORT** PI in the HTML file. Provided that the **IMPORT** PI is specified in the primary document, the **document.write** method can be used to add the custom element and attach to the behavior.

2.1.2 COMMENT

Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)

Start tag: **required**, End tag: **required**

Indicates an HTML comment. Neither the **comment** element nor text and markup contained within it are rendered. An HTML comment declared with a **comment** element is not considered to be atomic.

Additional details about the **comment** element are found at [HTMLCommentElement](#).

IE9 Mode and IE10 Mode (All Versions)

The **HTMLCommentElement** interface is not supported.

2.1.3 MARQUEE

Start tag: **required**, End tag: **required**

Defines a scrolling text marquee.

The following attributes are associated with the **marquee** element:

- [behavior](#)
- [bgColor](#)
- [dataFld](#)
- [dataFormatAs](#)
- [dataSrc](#)
- [direction](#)
- [height](#)
- [hspace](#)
- [loop](#)
- [scrollAmount](#)

- [scrollDelay](#)
- [trueSpeed](#)
- [vspace](#)
- [width](#)

Additional details on the **marquee** element is found at [HTMLMarqueeElement](#).

2.1.4 NOBR

*Start tag: **required**, End tag: **required***

The **noBR** element renders text without line breaks.

This element is not rendered. This element derives from the **HTMLElement** interface (as defined in [\[HTML\]](#)).

2.1.5 PUBLIC:ATTACH

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

*Start tag: **required**, End tag: **forbidden***

The PUBLIC:ATTACH element binds a function or script to an event in an HTC.

The element defines the following attributes:

- [event](#)
- [for](#)
- id (as defined by [\[HTML\]](#))
- [onevent](#)

2.1.6 PUBLIC:COMPONENT

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

*Start tag: **required**, End tag: **required***

The PUBLIC:COMPONENT element identifies the content of the file as an HTC.

The element defines the following attributes:

- id (as defined by [\[HTML\]](#))
- [lightWeight](#)
- [literalContent](#)
- name (as defined by [\[HTML\]](#))
- [supportsEditMode](#)
- [tagName](#)

- [urn](#)

The following child element can occur at most once:

- [PUBLIC:DEFAULTS](#)

The following child elements can occur one or more times:

- [PUBLIC:ATTACH](#)
- [PUBLIC:EVENT](#)
- [PUBLIC:METHOD](#)
- [PUBLIC:PROPERTY](#)

Additional details on the PUBLIC:COMPONENT element is found at [HTCComponentElement](#).

2.1.7 PUBLIC:DEFAULTS

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

Start tag: **required**, End tag: **forbidden**

The PUBLIC:DEFAULTS element sets default properties for an HTC.

The element defines the following attributes:

- [canHaveHTML](#)
- style (as defined by [\[HTML\]](#))
- [tabStop](#)
- [viewInheritStyle](#)
- [viewLinkContent](#)
- [viewMasterTab](#)

Additional details on the PUBLIC:DEFAULTS element is found at [HTCElementBehaviorDefaults](#).

2.1.8 PUBLIC:EVENT

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

Start tag: **required**, End tag: **forbidden**

The PUBLIC:EVENT element defines an event that is exposed by the HTC to the primary document.

The element defines the following attributes:

- id (as defined by [\[HTML\]](#))
- name (as defined by [\[HTML\]](#))

2.1.9 PUBLIC:METHOD

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

Start tag: **required**, End tag: **forbidden**

The PUBLIC:METHOD element defines a method that is exposed by an HTC to the primary document.

The element defines the following attributes:

- id (as defined by [\[HTML\]](#))
- [internalName](#)
- name (as defined by [\[HTML\]](#))

2.1.10 PUBLIC:PROPERTY

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

Start tag: **required**, End tag: **forbidden**

The PUBLIC:PROPERTY element defines a property that is exposed by the HTC to the primary document.

The element defines the following attributes:

- [get](#)
- id (as defined by [\[HTML\]](#))
- [internalName](#)
- name (as defined by [\[HTML\]](#))
- [persist](#)
- [put](#)
- [value](#)

Additional details on the PUBLIC:PROPERTY element is found at [HTCPropertyElement](#).

2.2 Interfaces

Internet Explorer-only Interfaces

The following interfaces are extensions of HTML:

- [HTCComponentElement](#)
- [HTCElementBehaviorDefaults](#)
- [HTCPropertyElement](#)
- [HTMLBlockElement](#)
- [HTMLCommentElement](#)
- [HTMLDDElement](#)
- [HTMLDTElement](#)

- [HTMLMarqueeElement](#)

DOM Interfaces

The following interfaces are defined in [\[DOM Level 2 - HTML\]](#) and extended by Windows® Internet Explorer®:

- [HTMLAnchorElement](#)
- [HTMLAreaElement](#)
- [HTMLAreaCollection](#)
- [HTMLBodyElement](#)
- [HTMLButtonElement](#)
- [HTMLCollection](#)
- [HTMLDivElement](#)
- [HTMLDocument](#)
- [HTMLElement](#)
- [HTMLFieldSetElement](#)
- [HTMLFormElement](#)
- [HTMLFrameElement](#)
- [HTMLFrameSetElement](#)
- [HTMLHeadingElement](#)
- [HTMLHRElement](#)
- [HTMLIFrameElement](#)
- [HTMLImageElement](#)
- [HTMLInputElement](#)
- [HTMLIsIndexElement](#)
- [HTMLLabelElement](#)
- [HTMLLegendElement](#)
- [HTMLMetaElement](#)
- [HTMLObjectElement](#)
- [HTMLOptionElement](#)
- [HTMLParagraphElement](#)
- [HTMLSelectElement](#)
- [HTMLSpanElement](#)

- [HTMLStyleElement](#)
- [HTMLTableCaptionElement](#)
- [HTMLTableCellElement](#)
- [HTMLTableElement](#)
- [HTMLTableRowElement](#)
- [HTMLTableSectionElement](#)
- [HTMLTextAreaElement](#)

2.2.1 Internet Explorer-Only Interfaces

The following interfaces are extensions of HTML.

2.2.1.1 HTCComponentElement

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

The **HTCComponentElement** (IHTCDescBehavior) interface provides access to the following attribute:

- [urn](#)

2.2.1.2 HTCElementBehaviorDefaults

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

The **HTCElementBehaviorDefaults** (IHTMLElementDefaults) provides access to the following attributes:

- [canHaveHTML](#)
- [tabStop](#)
- [viewInheritStyle](#)
- [viewMasterTab](#)

2.2.1.3 HTCPropertyElement

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

The **HTCPropertyElement** (IHTCPropertyBehavior) provides access to the following attribute:

- [value](#)

2.2.1.4 HTMLBlockElement

The **HTMLBlockElement** interface provides access to properties and elements available to most block elements.

The **HTMLBlockElement** interface has been added, with the following attribute:

- [clear](#)

This interface provides access to the **blockquote** element.

```
//Introduced in Internet Explorer.
interface HTMLBlockElement : HTMLElement {
    attribute DOMString      clear;
};
```

2.2.1.5 HTMLCommentElement

Quirks Mode, IE7 Mode, and IE8 Mode (All Versions)

The **HTMLCommentElement** interface indicates a comment that will not be displayed.

The **HTMLCommentElement** interface has been added with the following attributes:

- data – Defined in [\[DOM Level 2 - Core\]](#)
- length – Defined in [\[DOM Level 2 - Core\]](#)
- text – see [\[MS-DOM2CE\]](#)

The **HTMLCommentElement** interface has been added with the following methods:

- appendData() – Defined in [\[DOM Level 2 - Core\]](#)
- atomic – See [\[MS-DOM2CE\]](#)
- deleteData() – Defined in [\[DOM Level 2 - Core\]](#)
- insertData() – Defined in [\[DOM Level 2 - Core\]](#)
- replaceData() – Defined in [\[DOM Level 2 - Core\]](#)
- substringData() – Defined in [\[DOM Level 2 - Core\]](#)

For more information about the **HTMLCommentElement** interface, see [comment](#).

```
//Introduced in Internet Explorer.
interface HTMLCommentElement : HTMLElement {
    attribute DOMString      datatext
    Node                      appendData(in DOMstring data);
    Node                      atomic();
    Node                      deleteData(in DOMstring data);
    Node                      insertData(in DOMstring data);
    Node                      replaceData(in DOMstring data);
    Node                      substringData(in DOMstring data);
};
```

IE9 Mode and IE10 Mode (All Versions)

The **HTMLCommentElement** interface is not supported.

2.2.1.6 HTMLDDElement

The **HTMLDDElement** interface provides access to the **dd** object.

The **HTMLDDElement** interface has been added, with the following attribute:

- [noWrap](#)

```
//Introduced in Internet Explorer.  
interface HTMLDDElement : HTMLElement {  
    attribute boolean        noWrap;  
};
```

2.2.1.7 HTMLDTElement

The **HTMLDTElement** interface provides access for controlling the text-wrapping characteristics of the **HTMLDTElement** object.

The **HTMLDTElement** interface has been added, with the following attribute:

- [noWrap](#)

```
//Introduced in Internet Explorer.  
interface HTMLDTElement : HTMLElement {  
    attribute boolean        noWrap;  
};
```

2.2.1.8 HTMLMarqueeElement

The **HTMLMarqueeElement** interface is used to specify a marquee element. For more information about the **HTMLMarqueeElement** interface, see [marquee](#).

The **HTMLMarqueeElement** interface has been added, with the following attributes:

- [behavior](#)
- [bgColor](#)
- [dataFld](#)
- [dataFormatAs](#)
- [dataSrc](#)
- [direction](#)
- [height](#)
- [hspace](#)
- [loop](#)
- [onbounce](#)
- [onfinish](#)

- [onstart](#)
- [scrollAmount](#)
- [scrollDelay](#)
- [trueSpeed](#)
- [vspace](#)
- [width](#)

```
//Introduced in Internet Explorer.
interface HTMLMarqueeElement : HTMLElement {
    attribute DOMString    behavior;
    attribute DOMString    bgColor;
    attribute DOMString    dataFld;
    attribute DOMString    dataFormatAs;
    attribute DOMString    dataSrc;
    attribute DOMString    direction;
    attribute DOMString    height;
    attribute long          hspace;
    attribute long          loop;
    attribute Function      onbounce;
    attribute Function      onfinish;
    attribute Function      onstart;
    attribute long          scrollAmount;
    attribute long          scrollDelay;
    attribute boolean       trueSpeed;
    attribute long          vspace;
    attribute DOMString     width;
};
```

2.2.2 DOM Interfaces

The following interfaces are defined in [\[DOM Level 2 - HTML\]](#) and extended by Windows® Internet Explorer®.

2.2.2.1 HTMLAnchorElement

The **HTMLAnchorElement** interface designates the destination of a hypertext link.

The **HTMLAnchorElement** interface has been extended with the following attributes:

- [dataFld](#)
- [dataFormatAs](#) (not supported by the element)
- [dataSrc](#)
- [hash](#)
- [host](#)
- [hostname](#)
- [Methods](#)

- [mimeType](#)
- [nameProp](#)
- [pathname](#)
- [port](#)
- [protocol](#)
- [protocolLong](#)
- [search](#)
- [urn](#)

When an anchor contains an image, the image has a border by default. The color of the border reflects the visited or unvisited status of the link.

```
//Introduced in Internet Explorer.
interface HTMLAnchorElement : HTMLElement {
    attribute DOMString      dataFld;
    attribute DOMString      dataFormatAs;
    attribute DOMString      dataSrc;
    attribute DOMString      hash;
    attribute DOMString      host;
    attribute DOMString      hostname;
    attribute DOMString      Methods;
    attribute DOMString      mimeType;
    attribute DOMString      nameProp;
    attribute DOMString      pathname;
    attribute DOMString      port;
    attribute DOMString      protocol;
    attribute DOMString      protocolLong;
    attribute DOMString      search;
    attribute DOMString      urn;
};
```

2.2.2.2 HTMLAreaElement

The **HTMLAreaElement** interface specifies the shape of a hot spot in a client-side image map.

The **HTMLAreaElement** interface has been extended with the following attributes:

- [hash](#)
- [host](#)
- [hostname](#)
- [pathname](#)
- [port](#)
- [protocol](#)
- [search](#)

```
//Introduced in Internet Explorer.
interface HTMLAreaElement : HTMLElement {
    attribute DOMString    hash;
    attribute DOMString    host;
    attribute DOMString    hostname;
    attribute DOMString    pathname;
    attribute DOMString    port;
    attribute DOMString    protocol;
    attribute DOMString    search;
};
```

2.2.2.3 HTMLAreasCollection

The **HTMLAreasCollection** interface specifies the collection of areas that make up the image map.

The **HTMLAreasCollection** interface has been extended with the following methods:

- [add\(\)](#)
- [remove\(\)](#)
- [tags\(\)](#)
- [urns\(\)](#)

Areas can be added or removed from the collection. If duplicate names are found, a collection of those named items is returned. Subsequently, collections of duplicate names must be referenced by ordinal position.

```
//Introduced in Internet Explorer.
interface HTMLAreasCollection : HTMLCollection {
    void    add(in HTMLElement oElement,
               in long oBefore);
    void    remove(in long iIndex);
    Node    tags(in DOMString sTag);
    Node    urns(in DOMString sUrn);
};
```

2.2.2.4 HTMLBodyElement

The **HTMLBodyElement** interface provides access to the **body** element, and specifies the beginning and end of the document body.

The **HTMLBodyElement** interface has been extended with the following attributes:

- [bgProperties](#)
- [bottomMargin](#)
- [leftMargin](#)
- [noWrap](#)
- [onbeforeunload](#)
- [onselect](#)

- [rightMargin](#)
- [scroll](#)
- [topMargin](#)

The **body** element is a block element.

```
//Introduced in Internet Explorer.
interface HTMLBodyElement : HTMLElement {

    attribute DOMString      bgProperties;
    attribute DOMString      bottomMargin;
    attribute DOMString      leftMargin;
    attribute boolean         noWrap;
    attribute Function        onbeforeunload;
    attribute Function        onselect;
    attribute DOMString      rightMargin;
    attribute DOMString      scroll;
    attribute DOMString      topMargin;
};
```

2.2.2.5 HTMLButtonElement

The **HTMLButtonElement** interface renders content as an HTML button.

The **HTMLButtonElement** interface has been extended with the following attributes:

- [status](#)
- [dataFld](#)
- [dataFormatAs](#)
- [dataSrc](#)

The **button** element is a block element.

```
//Introduced in Internet Explorer.
interface HTMLButtonElement : HTMLElement {
    attribute boolean      status;
    attribute DOMString    dataFld;
    attribute DOMString    dataFormatAs;
    attribute DOMString    dataSrc;
};
```

2.2.2.6 HTMLCollection

The **HTMLCollection** interface provides access to collections of objects.

The **HTMLCollection** interface has been extended with the following methods:

- [tags\(\)](#)
- [urns\(\)](#)


```
//Introduced in Internet Explorer.
interface HTMLCollection {
    Node          tags(in DOMString sTag);
    Node          urns(in DOMString sUrn);
};
```

2.2.2.7 HTMLDivElement

The **HTMLDivElement** interface provides access to the **div** object.

The **HTMLDivElement** interface has been extended with the following attributes:

- [dataFld](#)
- [dataFormatAs](#)
- [dataSrc](#)
- [noWrap](#)

The **div** element is a block element.

```
//Introduced in Internet Explorer.
interface HTMLDivElement : HTMLElement {
    attribute DOMString      dataFld;
    attribute DOMString      dataFormatAs;
    attribute DOMString      dataSrc;
    attribute boolean        noWrap;
};
```

2.2.2.8 HTMLDocument

The **HTMLDocument** interface retrieves information about the document, and examines and modifies the HTML elements and text in the document.

The **HTMLDocument** interface has been extended with the following attributes:

- [alinkColor](#)
- [bgColor](#)
- [dir](#)
- [fgColor](#)
- [linkColor](#)
- [parentDocument](#)
- [uniqueID](#)
- [vlinkColor](#)
- [all](#)
- [frames](#)

```
//Introduced in Internet Explorer.
interface HTMLDocument : Document {
    attribute DOMString    alinkColor;
    attribute DOMString    bgColor;
    attribute DOMString    dir;
    attribute DOMString    fgColor;
    attribute DOMString    linkColor;
    attribute DOMString    parentDocument;
    attribute DOMString    uniqueID;
    attribute DOMString    vlinkColor;
    attribute HTMLCollection all;
    attribute HTMLCollection frames;
};
```

2.2.2.9 HTMLInputElement

The **HTMLInputElement** interface provides the ability to programmatically access the properties and methods that are common to all elements.

The **HTMLInputElement** interface has been added, with the following attributes:

- [isDisabled](#)
- [isMultiLine](#)
- [onactivate](#)
- [onafterupdate](#)
- [onbeforeactivate](#)
- [onbeforecopy](#)
- [onbeforecut](#)
- [onbeforedeactivate](#)
- [onbeforeeditfocus](#)
- [onbeforepaste](#)
- [onbeforeupdate](#)
- [oncellchange](#)
- [oncontrolselect](#)
- [oncopy](#)
- [oncut](#)
- [ondataavailable](#)
- [ondatasetchanged](#)
- [ondatasetcomplete](#)
- [ondeactivate](#)

- [onerrorupdate](#)
- [onfilterchange](#)
- [onfocusin](#)
- [onfocusout](#)
- [onhelp](#)
- [onlosecapture](#)
- [onmouseenter](#)
- [onmouseleave](#)
- [onmove](#)
- [onmoveend](#)
- [onmovestart](#)
- [onpaste](#)
- [onpropertychange](#)
- [onresizeend](#)
- [onresizestart](#)
- [onrowenter](#)
- [onrowexit](#)
- [onrowsdelete](#)
- [onrowsinserted](#)
- [onselectstart](#)
- [uniqueID](#)
- [uniqueNumber](#)

The **HTMLElement** interface has been added, with the following method:

- [setActive](#)

2.2.2.10 HTMLFieldSetElement

The **HTMLFieldSetElement** interface draws a box around the text and other elements it contains.

The **HTMLFieldSetElement** interface has been extended with the following attribute:

- [align](#)

This element is useful for grouping elements in a form and for distinctively marking text in a document.

```
//Introduced in Internet Explorer.
interface HTMLFieldSetElement : HTMLElement {
    attribute DOMString    align;
};
```

2.2.2.11 HTMLFormElement

The **HTMLFormElement** interface specifies that the contained controls take part in a form.

The **HTMLFormElement** interface has been extended with the following attribute:

- [encoding](#)

The **HTMLFormElement** interface has been extended with the following methods:

- [item\(\)](#)
- [namedItem\(\)](#)
- [tags\(\)](#)
- [urns\(\)](#)

```
//Introduced in Internet Explorer.
interface HTMLFormElement : HTMLElement {
    attribute DOMString    encoding;
    Node                  item(in unsigned long index);
    Node                  namedItem(in DOMString name);
    Node                  tags(in DOMString sTag);
    Node                  urns(in DOMString sUrn);
};
```

2.2.2.12 HTMLFrameElement

The **HTMLFrameElement** interface specifies an individual frame within a frameset.

The **HTMLFrameElement** interface has been extended with the following attributes:

- [contentWindow](#)
- [allowTransparency](#)
- [border](#)
- [borderColor](#)
- [dataFld](#)
- [dataFormatAs](#) (not supported by this element)
- [dataSrc](#)
- [frameBorder](#)
- [frameSpacing](#)

- [height](#)
- [width](#)

```
//Introduced in Internet Explorer.
interface HTMLFrameElement : HTMLElement {
    attribute Window        contentWindow;
    attribute boolean       allowTransparency;
    attribute DOMString     border;
    attribute DOMString     borderColor;
    attribute DOMString     dataFld;
    attribute DOMString     dataFormatAs;
    attribute DOMString     dataSrc;
    attribute DOMString     frameSpacing;
    attribute DOMString     frameBorder;
    attribute DOMString     height;
    attribute DOMString     width;
};
```

2.2.2.13 HTMLFrameSetElement

The **HTMLFrameSetElement** interface specifies a frameset.

The **HTMLFrameSetElement** interface has been extended with the following attributes:

- [onbeforeunload](#)
- [border](#)
- [borderColor](#)
- [frameBorder](#)
- [frameSpacing](#)
- [name](#)

Use framesets to organize multiple frames and nested framesets. A frameset organizes multiple frames on the screen. The only tags valid within a frameset are frame, nested frameSet, and noFrames.

```
//Introduced in Internet Explorer.
interface HTMLFrameSetElement : HTMLElement {
    attribute Function       onbeforeunload;
    attribute DOMString     border;
    attribute DOMString     borderColor;
    attribute DOMString     frameBorder;
    attribute DOMString     frameSpacing;
    attribute DOMString     name;
};
```

2.2.2.14 HTMLHeadingElement

The **HTMLHeadingElement** interface renders text as a heading style.

The **HTMLHeadingElement** interface has been extended with the following attribute:

- [clear](#)

```
//Introduced in Internet Explorer.
interface HTMLHeadingElement : HTMLElement {
    attribute DOMString      clear;
};
```

2.2.2.15 HTMLHRElement

The **HTMLHRElement** interface draws a horizontal rule.

The **HTMLHRElement** interface has been extended with the following attribute:

- [color](#)

```
//Introduced in Internet Explorer.
interface HTMLHRElement : HTMLElement {
    attribute DOMString      color;
};
```

2.2.2.16 HTMLIFrameElement

The **HTMLIFrameElement** interface creates inline floating frames.

The **HTMLIFrameElement** interface has been extended with the following attributes:

- [contentWindow](#)
- [allowTransparency](#)
- [border](#)
- [dataFld](#)
- [dataFormatAs](#) (Not supported by this element.)
- [dataSrc](#)
- [frameSpacing](#)
- [hspace](#)
- [noResize](#)
- [vspace](#)

```
//Introduced in Internet Explorer.
interface HTMLIFrameElement : HTMLElement {
    attribute Window      contentWindow;
    attribute boolean     allowTransparency;
    attribute DOMString   border;
    attribute DOMString   dataFld;
    attribute DOMString   dataFormatAs;
    attribute DOMString   dataSrc;
    attribute DOMString   frameSpacing;
    attribute long        hspace;
```

```

        attribute boolean    noResize;
        attribute long       vspace;
    };

```

2.2.2.17 HTMLImageElement

The **HTMLImageElement** interface provides access to properties and methods supported by the **img** element and the **input** element of the **image** type.

The **HTMLImageElement** interface has been extended with the following attributes:

- [dataFld](#)
- [dataFormatAs](#)
- [dataSrc](#)
- [dynsrc](#)
- [fileCreatedDate](#)
- [fileModifiedDate](#)
- [fileSize](#)
- [fileUpdatedDate](#)
- [href](#)
- [loop](#)
- [lowsrc](#)
- [mimeType](#)
- [nameProp](#)
- [protocol](#)
- [start](#)

```

//Introduced in Internet Explorer.
interface HTMLImageElement : HTMLElement {
    attribute DOMString    dataFld;
    attribute DOMString    dataFormatAs;
    attribute DOMString    dataSrc;
    attribute DOMString    dynsrc;
    attribute DOMString    fileCreatedDate;
    attribute DOMString    fileModifiedDate;
    attribute DOMString    fileSize;
    attribute DOMString    fileUpdatedDate;
    attribute DOMString    href;
    attribute long         loop;
    attribute DOMString    lowsrc;
    attribute DOMString    mimeType;
    attribute DOMString    nameProp;
    attribute DOMString    protocol;
}

```

```

        attribute DOMString      start;
    };

```

2.2.2.18 HTMLInputElement

The **HTMLInputElement** interface specifies any type of input control.

The **HTMLInputElement** interface has been extended with the following attributes:

- [border](#)
- [dataFld](#)
- [dataFormatAs](#)
- [dataSrc](#)
- [dynsrc](#)
- [height](#)
- [hspace](#)
- [lowsrc](#)
- [indeterminate](#)
- [loop](#)
- [start](#)
- [status](#)
- [vspace](#)
- [width](#)

```

//Introduced in Internet Explorer.
interface HTMLInputElement : HTMLElement {
    attribute boolean      status;
    attribute DOMString    border;
    readonly attribute boolean complete;
    attribute DOMString    dataFld;
    attribute DOMString    dataFormatAs;
    attribute DOMString    dataSrc;
    attribute DOMString    dynsrc;
    attribute DOMString    height;
    attribute long         hspace;
    attribute DOMString    lowsrc;
    attribute boolean      indeterminate;
    attribute long         loop;
    attribute DOMString    start;
    attribute long         vspace;
    attribute DOMString    width;
};

```


2.2.2.19 HTMLIsIndexElement

The **HTMLIsIndexElement** interface indicates that the document contains a searchable index.

The **HTMLIsIndexElement** interface has been extended with the following attribute:

- [action](#)

To use the `isIndex` tag, the server must provide a search engine that supports this tag. The script to be executed on the server can be specified by the **HTMLIsIndexElement::action** property on this tag.

The **isIndex** element can be used only within the `head` tag.

```
//Introduced in Internet Explorer.  
interface HTMLIsIndexElement : HTMLElement {  
    attribute DOMString    action;  
};
```

2.2.2.20 HTMLLabelElement

The **HTMLLabelElement** interface specifies a label for a control-like element.

The **HTMLLabelElement** interface has been extended with the following attributes:

- [dataFld](#)
- [dataFormatAs](#)
- [dataSrc](#)

```
//Introduced in Internet Explorer.  
interface HTMLLabelElement : HTMLElement {  
    attribute DOMString    dataFld;  
    attribute DOMString    dataFormatAs;  
    attribute DOMString    dataSrc;  
};
```

2.2.2.21 HTMLLegendElement

The **HTMLLegendElement** interface inserts a caption into the box drawn by the `fieldSet` element.

The **HTMLLegendElement** interface has been extended with the following attributes:

- [dataFld](#)
- [dataFormatAs](#)
- [dataSrc](#)

If provided, this element must be the first element in **fieldSet**.

```
//Introduced in Internet Explorer.  
interface HTMLLegendElement : HTMLElement {  
    attribute DOMString    dataFld;  
};
```

```

        attribute DOMString      dataFormatAs;
        attribute DOMString      dataSrc;
};

```

2.2.2.22 HTMLLinkElement

The **HTMLLinkElement** interface provides methods to access a link element that is used to specify a link to an external document.

The **HTMLLinkElement** interface has been extended with the following attribute:

- [styleSheet](#)

```

//Introduced in Internet Explorer.
interface HTMLLinkElement : HTMLElement {
    attribute CSSStyleSheet  styleSheet;
};

```

2.2.2.23 HTMLMetaElement

The **HTMLMetaElement** interface represents the **meta** element.

The **HTMLMetaElement** interface has been extended with the following attributes:

- [url](#)
- [charset](#)

```

//Introduced in Internet Explorer.
interface HTMLMetaElement : HTMLElement {
    attribute DOMString      url;
    attribute DOMString      charset;
};

```

You can use the **meta** element to include hidden information about the document and to insert document-related information used by some search engines.

The meta element can be used only within the **head** element.

2.2.2.24 HTMLObjectElement

The **HTMLObjectElement** interface inserts an object into the HTML page.

The **HTMLObjectElement** interface has been extended with the following attributes:

- [alt](#)
- [altHtml](#)
- [BaseHref](#)
- [classid](#)

- [dataFld](#)
- [dataFormatAs](#)
- [dataSrc](#)
- [object](#)

```
//Introduced in Internet Explorer.
interface HTMLObjectElement : HTMLElement {
    attribute DOMString      altHtml;
    attribute DOMString      BaseHref;
    attribute DOMString      classId;
    attribute DOMString      dataFld;
    attribute DOMString      dataFormatAs;
    attribute DOMString      dataSrc;
    readonly attribute any    object;
};
```

2.2.2.25 HTMLOptionElement

The **HTMLOptionElement** interface denotes one choice in a **select** block.

The **HTMLOptionElement** interface has been extended with the following attributes:

- [dataFld](#)
- [dataFormatAs](#)
- [dataSrc](#)

The **option** element is a block element.

```
//Introduced in Internet Explorer.
interface HTMLOptionElement : HTMLElement {
    attribute DOMString      dataFld;
    attribute DOMString      dataFormatAs;
    attribute DOMString      dataSrc;
};
```

2.2.2.26 HTMLParagraphElement

The **HTMLParagraphElement** interface denotes a paragraph.

The **HTMLParagraphElement** interface has been extended with the following attribute:

- [clear](#)

```
//Introduced in Internet Explorer.
interface HTMLParagraphElement : HTMLElement {
    attribute DOMString      clear;
};
```

2.2.2.27 HTMLSelectElement

The **HTMLSelectElement** interface provides access to the properties and methods of the **select** element.

The **HTMLSelectElement** interface has been extended with the following attributes:

- [dataFld](#)
- [dataFormatAs](#)
- [dataSrc](#)

The **HTMLSelectElement** interface has been extended with the following methods:

- [item\(\)](#)
- [namedItem\(\)](#)
- [tags\(\)](#)
- [urns\(\)](#)

```
//Introduced in Internet Explorer.
interface HTMLSelectElement : HTMLElement {
    attribute DOMString    dataFld;
    attribute DOMString    dataFormatAs;
    attribute DOMString    dataSrc;
    Node                    item(in unsigned long index);
    Node                    namedItem(in DOMString name);
    Node                    tags(in DOMString sTag);
    Node                    urns(in DOMString sUrn);
};
```

2.2.2.28 HTMLSpanElement

The **HTMLSpanElement** interface provides access to the properties of the **span** element.

The **HTMLSpanElement** interface has been extended with the following attributes:

- [dataFld](#)
- [dataFormatAs](#)
- [dataSrc](#)

```
//Introduced in Internet Explorer.
interface HTMLSpanElement : HTMLElement {
    attribute DOMString    dataFld;
    attribute DOMString    dataFormatAs;
    attribute DOMString    dataSrc;
};
```

2.2.2.29 HTMLStyleElement

The **HTMLStyleElement** interface specifies the style sheet for the page.

The **HTMLStyleElement** interface has been extended with the following attribute:

- [styleSheet](#)

The **style** element can be used only within either the **head** or the **body** element.

```
//Introduced in Internet Explorer.
interface HTMLStyleElement : HTMLElement {
    attribute CSSStyleSheet    styleSheet;
};
```

2.2.2.30 HTMLTableCaptionElement

The **HTMLTableCaptionElement** interface specifies a caption for a table **element**.

The **HTMLTableCaptionElement** interface has been extended with the following attribute:

- [vAlign](#)

The **caption** element is a block element.

```
//Introduced in Internet Explorer.
interface HTMLTableCaptionElement : HTMLElement {
    attribute DOMString    vAlign;
};
```

2.2.2.31 HTMLTableCellElement

The **HTMLTableCellElement** interface specifies a cell in a table.

The **HTMLTableCellElement** interface has been extended with the following attributes:

- [background](#)
- [borderColor](#)
- [borderColorDark](#)
- [borderColorLight](#)

The **td** element is a block element.

```
//Introduced in Internet Explorer.
interface HTMLTableCellElement : HTMLElement {
    attribute DOMString    background;
    attribute DOMString    borderColor;
    attribute DOMString    borderColorDark;
    attribute DOMString    borderColorLight;
};
```

2.2.2.32 HTMLTableElement

The **HTMLTableElement** interface specifies that the contained content is organized into a table with rows and columns.

The **HTMLTableElement** interface has been extended with the following attributes:

- [background](#)
- [borderColor](#)
- [borderColorDark](#)
- [borderColorLight](#)
- [cells](#)
- [cols](#)
- [dataFld](#)
- [dataFormatAs](#)
- [dataPageSize](#)
- [dataSrc](#)
- [height](#)

The **HTMLTableElement** interface has been extended with the following method:

- [moveRow\(\)](#)

Valid tags within a table include **caption**, **col**, **colGroup**, **tBody**, **tHead**, and **tr**.

The **table** element is a block element.

```
//Introduced in Internet Explorer.
interface HTMLTableElement : HTMLElement {
    attribute DOMString      background;
    attribute DOMString      borderColor;
    attribute DOMString      borderColorDark;
    attribute DOMString      borderColorLight;
    attribute DOMString      cells;
    attribute long           cols;
    attribute DOMString      dataFld;
    attribute DOMString      dataFormatAs;
    attribute DOMString      dataSrc;
    attribute DOMString      height;
    Node                      moveRow(in long iSource, in long iTarget);
};
```

2.2.2.33 HTMLTableRowElement

The **HTMLTableRowElement** interface specifies a row in a table.

The **HTMLTableRowElement** interface has been extended with the following attributes:

- [borderColor](#)
- [height](#)
- [borderColorDark](#)

- [borderColorLight](#)

Within a row, the **td** and **th** tags are valid.

The **tr** element is a block element.

```
//Introduced in Internet Explorer.
interface HTMLTableRowElement : HTMLElement {
    attribute DOMString      borderColor;
    attribute DOMString      height;
    attribute DOMString      borderColorDark;
    attribute DOMString      borderColorLight;
};
```

2.2.2.34 HTMLTableSectionElement

The **HTMLTableSectionElement** interface designates rows as the body of the table.

The **HTMLTableSectionElement** interface has been extended with the following attribute:

- [bgColor](#)

The **HTMLTableSectionElement** interface has been extended with the following method:

- [moveRow\(\)](#)

The **HTMLTableSectionElement** is exposed for all tables, even if the table does not explicitly define a **tBody** element.

The **tBody** element is a block element.

```
//Introduced in Internet Explorer.
interface HTMLTableSectionElement : HTMLElement {
    attribute DOMString      bgColor;
    Node                    moveRow(in long iSource, in long iTarget);
};
```

2.2.2.35 HTMLTextAreaElement

The **HTMLTextAreaElement** interface specifies a multiline text input control.

The **HTMLTextAreaElement** interface has been extended with the following attributes:

- [status](#)
- [dataFld](#)
- [dataFormatAs](#)
- [dataSrc](#)
- [wrap](#)

```
//Introduced in Internet Explorer.
interface HTMLTextAreaElement : HTMLElement {
    attribute boolean      status;
```

```

        attribute DOMString      dataFld;
        attribute DOMString      dataFormatAs;
        attribute DOMString      dataSrc;
        attribute DOMString      wrap;
    };

```

2.3 Attributes

DOM and Content Attributes

The following attributes are supported by both DOM and markup:

- [action](#)
- [align](#)
- [allowTransparency](#)
- [alt](#)
- [background](#)
- [BaseHref](#)
- [behavior](#)
- [bgColor](#)
- [bgProperties](#)
- [border](#)
- [borderColor](#)
- [borderColorDark](#)
- [borderColorLight](#)
- [bottomMargin](#)
- [canHaveHTML](#)
- [charset](#)
- [classid](#)
- [clear](#)
- [color](#)
- [cols](#)
- [dataFld](#)
- [dataFormatAs](#)
- [dataPageSize](#)

- [dataSrc](#)
- [direction](#)
- [dynsrc](#)
- [encoding](#)
- [frameBorder](#)
- [frameSpacing](#)
- [height](#)
- [href](#)
- [hspace](#)
- [leftMargin](#)
- [loop](#)
- [lowsrc](#)
- [methods](#)
- [name](#)
- [noResize](#)
- [noWrap](#)
- [rightMargin](#)
- [scroll](#)
- [scrollAmount](#)
- [scrollDelay](#)
- [start](#)
- [tabStop](#)
- [topMargin](#)
- [trueSpeed](#)
- [urn](#)
- [vAlign](#)
- [value](#)
- [viewInheritStyle](#)
- [viewMasterTab](#)
- [vspace](#)

- [width](#)
- [wrap](#)

DOM Attributes Only

The following attributes are only available through the object model:

- [alinkColor](#)
- [altHtml](#)
- [contentWindow](#)
- [dir](#)
- [fgColor](#)
- [fileCreatedDate](#)
- [fileModifiedDate](#)
- [fileSize](#)
- [fileUpdatedDate](#)
- [hash](#)
- [host](#)
- [hostname](#)
- [indeterminate](#)
- [isDisabled](#)
- [isMultiLine](#)
- [linkColor](#)
- [mimeType](#)
- [nameProp](#)
- [object](#)
- [parentDocument](#)
- [pathname](#)
- [port](#)
- [protocol](#)
- [protocolLong](#)
- [search](#)
- [status](#)

- [styleSheet](#)
- [uniqueID](#)
- [uniqueNumber](#)
- [url](#)
- [viewLink](#)
- [vlinkColor](#)

Content Attributes Only

The following attributes are declared only in markup:

- [application](#)
- [atomicselection](#)
- [event](#)
- [for](#)
- [get](#)
- [implementation](#)
- [internalName](#)
- [lightWeight](#)
- [literalContent](#)
- [namespace](#)
- [onevent](#)
- [persist](#)
- [put](#)
- [supportsEditMode](#)
- [tagName](#)
- [unselectable](#)
- [viewLinkContent](#)
- [xmlns](#)

Event Attributes

The following attributes support events and event handlers:

- [onactivate](#)
- [onafterupdate](#)

- [onbeforeactivate](#)
- [onbeforecopy](#)
- [onbeforecut](#)
- [onbeforedeactivate](#)
- [onbeforeeditfocus](#)
- [onbeforepaste](#)
- [onbeforeunload](#)
- [onbeforeupdate](#)
- [onbounce](#)
- [oncellchange](#)
- [oncontentready](#)
- [oncontentsave](#)
- [oncontrolselect](#)
- [oncopy](#)
- [oncut](#)
- [ondataavailable](#)
- [ondatasetchanged](#)
- [ondatasetcomplete](#)
- [ondeactivate](#)
- [ondetach](#)
- [ondocumentready](#)
- [onerrorupdate](#)
- [onfilterchange](#)
- [onfinish](#)
- [onfocusin](#)
- [onfocusout](#)
- [onhelp](#)
- [onlosecapture](#)
- [onmouseenter](#)
- [onmouseleave](#)

- [onmove](#)
- [onmoveend](#)
- [onmovestart](#)
- [onpaste](#)
- [onpropertychange](#)
- [onresizeend](#)
- [onresizestart](#)
- [onrowenter](#)
- [onrowexit](#)
- [onrowsdelete](#)
- [onrowsinserted](#)
- [onselect](#)
- [onselectstart](#)
- [onstart](#)

2.3.1 DOM and Content Attributes

This section contains a list of attributes that are supported in markup and can be accessed programmatically from the DOM.

2.3.1.1 action

action of type `DOMString`

Sets or retrieves the URL to which the form content is sent for processing.

The attribute's value is a string that specifies or receives the URL to use. If a relative path is specified, the base URL of the document is assumed.

For Windows® Internet Explorer® 8 or later, when the browser is in Internet Explorer 8 mode, the value of the **action** attribute depends on the context of the reference to the attribute. When read as a Document Object Model (DOM) attribute, **action** returns an absolute URL. The value specified by the page author is returned when **action** is read as a content attribute, when the page is displayed in an earlier document compatibility mode, or when the page is viewed with an earlier version of the browser.

The **action** attribute extends the [HTMLIsIndexElement](#) interface.

2.3.1.2 align

align of type `DOMString`

Sets or retrieves how the object is aligned with adjacent text.

Possible values are:

- `absbottom` (as defined in [HTML])
- `baseline`
- `bottom`
- `left` – Default.
- `middle`
- `right`
- `texttop`
- `top`

The **align** attribute extends the [HTMLFieldSetElement](#) interface.

2.3.1.3 allowTransparency

allowTransparency of type `boolean`

Sets or retrieves whether the object can be transparent.

Returns `true` if the object can be transparent. If `false`, the object is opaque. The default value is `false`.

When the property is set to:

- `false`, the `backgroundColor` property of the object can only be that of the window.
- `true`, the `backgroundColor` property of the object can be set to any value, including the default value of transparent.

The **allowTransparency** attribute extends the following interfaces:

- [HTMLFrameElement](#)
- [HTMLIFrameElement](#)

2.3.1.4 alt

alt of type `DOMString`

Alternate text for user agents not rendering the normal content of this element.

Note This attribute is standard on **img**, **area**, **applet**, and **input** tags.

The alt text is displayed in the object only if the object fails to load and there is no fallback markup. The alt text is not displayed as a tooltip when the mouse pointer hovers over the object.

The **alt** attribute extends the [HTMLObjectElement](#) interface.

2.3.1.5 background

background of type `DOMString`

Sets or retrieves the URL of the file that is used as a background picture tiled behind text and graphics in the object.

For Windows® Internet Explorer® 8 or later, when the browser is in Internet Explorer 8 mode, the value of the **background** attribute depends on the context of the reference to the attribute. When read as a Document Object Model (DOM) attribute, **background** returns an absolute URL. The value specified by the page author is returned when **background** is read as a content attribute, when the page is displayed in an earlier document compatibility mode, or when the page is viewed with an earlier version of the browser.

This attribute extends the following interfaces:

- [HTMLTableCellElement](#)
- [HTMLTableElement](#)

2.3.1.6 BaseHref

BaseHref of type `DOMString`

Read-only attribute that retrieves a string of the URL where the object tag can be found. This is often the href of the document that the object is in, or the value set by a base element.

This attribute extends the [HTMLObjectElement](#) interface.

2.3.1.7 behavior

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

behavior of type `DOMString`

Sets or retrieves how the text scrolls in the marquee.

Possible values are:

- `scroll` – Default. Marquee scrolls in the direction specified by the direction property. The text scrolls off the end and starts over each time.
- `alternate` – Marquee scroll direction reverses when its content reaches the edge of the container.
- `slide` – Marquee scrolls in the direction specified by the direction property. Text scrolls to the end and stops.

This attribute extends the [HTMLMarqueeElement](#) interface.

2.3.1.8 bgColor

bgColor of type `DOMString`

Deprecated. Sets or retrieves the background color behind the object.

This attribute extends the following interfaces:

- [HTMLDocument](#)
- [HTMLMarqueeElement](#)

- [HTMLTableSectionElement](#)

2.3.1.9 bgProperties

bgProperties of type `DOMString`

Sets or retrieves the properties of the background picture.

Possible values are:

- `empty string` – Default. The background can scroll.
- `fixed` – The background cannot scroll.

This attribute extends the [HTMLBodyElement](#) interface.

2.3.1.10 border

border of type `DOMString`

Sets or retrieves the space between the frames, including the 3-D border.

Setting a border to zero or omitting the attribute causes no border to be displayed. Supplying the border attribute without a value defaults to a single border.

This attribute extends the following interfaces:

- [HTMLFrameElement](#)
- [HTMLFrameSetElement](#)
- [HTMLIFrameElement](#)
- [HTMLInputElement](#)

2.3.1.11 borderColor

borderColor of type `DOMString`

Sets or retrieves the border color of the object.

Possible values are the color name or RGB value. For more information, see [\[CSS-Level2-2009\]](#) section 4.3.6 Colors.

Returns a six-digit hex value.

To render the color specified, the **border** attribute must be set to an integer greater than zero.

This attribute extends the following interfaces:

- [HTMLFrameElement](#)
- [HTMLFrameSetElement](#)
- [HTMLTableCellElement](#)
- [HTMLTableElement](#)

- [HTMLTableRowElement](#)

2.3.1.12 **borderColorDark**

borderColorDark of type `DOMString`

Sets or retrieves the color for one of the two colors used to draw the 3-D border of the object.

Possible values are the color name or RGB value. For more information, see [\[CSS-Level2-2009\]](#) section 4.3.6 Colors.

Returns a six-digit hex value.

This property is the opposite of [borderColorLight](#) and must be used with the border property that corresponds to the **border** attribute. This property does not affect the Cascading Style Sheets (CSS) border composite properties.

This attribute extends the following interfaces:

- [HTMLTableCellElement](#)
- [HTMLTableElement](#)
- [HTMLTableRowElement](#)

2.3.1.13 **borderColorLight**

borderColorLight of type `DOMString`

Sets or retrieves the color for one of the two colors used to draw the 3-D border of the object.

Possible values are the color name or RGB value. For more information, see [\[CSS-Level2-2009\]](#) section 4.3.6 Colors.

Returns a six-digit hex value.

This property is the opposite of [borderColorDark](#) and must be used with the border property that corresponds to the **border** attribute. This property does not affect the Cascading Style Sheets (CSS) border composite properties.

This attribute extends the following interfaces:

- [HTMLTableCellElement](#)
- [HTMLTableElement](#)
- [HTMLTableRowElement](#)

2.3.1.14 **bottomMargin**

bottomMargin of type `DOMString`

Sets or retrieves the bottom margin of the entire body of the page.

The default value is 15 pixels. If the value is an empty string, the bottom margin is set on the bottom edge of the window or frame.

The specified value overrides the default margin. By default, when the value of this property is set, the opposite margin is set to the same value.

This attribute extends the [HTMLBodyElement](#) interface.

2.3.1.15 canHaveHTML

canHaveHTML of type `boolean`

Returns `true` if the element can contain HTML markup, or `false` otherwise.

This attribute is read-only for all objects except the **defaults** object. It is read/write for the **defaults** object.

This attribute extends the [HTMLElement](#) and [HTCElementBehaviorDefaults](#) interface.

2.3.1.16 charset

charset of type `DOMString`

Sets or retrieves the character set used to encode the object.

This attribute extends the [HTMLMetaElement](#) interface.

2.3.1.17 classid

classid of type `DOMString`

Sets or retrieves the class identifier for the object.

This attribute extends the [HTMLObjectElement](#) interface.

2.3.1.18 clear

clear of type `DOMString`

Sets or retrieves the side on which floating objects are not to be positioned when any block-level element is inserted into the document.

Possible values are:

- `all` – Object is moved below any floating object.
- `left` – Object is moved below any floating object on the left side.
- `right` – Object is moved below any floating object on the right side.
- `none` – Floating objects are allowed on all sides.

This attribute extends the following interfaces:

- [HTMLBlockElement](#)
- [HTMLHeadingElement](#)
- [HTMLParagraphElement](#)

2.3.1.19 color

color of type `DOMString`

Sets or retrieves the color to be used by the object.

Possible values are the color name or RGB value. For more information, see [\[CSS-Level2-2009\]](#) section 4.3.6 Colors.

Returns a six-digit hex value.

This attribute extends the [HTMLHRElement](#) interface.

2.3.1.20 cols

cols of type `long`

Sets or retrieves the number of columns in the table.

This attribute extends the [HTMLTableElement](#) interface.

2.3.1.21 dataFld

dataFld of type `DOMString`

Sets or retrieves a field of a given data source, as specified by the **dataSrc** property, to bind to the specified object.

This attribute extends the following interfaces:

- [HTMLAnchorElement](#)
- [HTMLButtonElement](#)
- [HTMLDivElement](#)
- [HTMLFrameElement](#)
- [HTMLIFrameElement](#)
- [HTMLImageElement](#)
- [HTMLInputElement](#)
- [HTMLLabelElement](#)
- [HTMLLegendElement](#)
- [HTMLMarqueeElement](#)
- [HTMLObjectElement](#)
- [HTMLOptionElement](#)
- [HTMLSelectElement](#)
- [HTMLSpanElement](#)
- [HTMLTableElement](#)

- [HTMLTextAreaElement](#)

2.3.1.22 dataFormatAs

dataFormatAs of type `DOMString`

Sets or retrieves the rendering format for the data supplied to the object.

Possible values are:

- `text` – Default. Data is rendered as text.
- `html` – Data is rendered as HTML.
- `localized-text` – Data is rendered using the locale settings of the client machine.

This attribute extends the following interfaces:

- [HTMLAnchorElement](#)
- [HTMLButtonElement](#)
- [HTMLDivElement](#)
- [HTMLFrameElement](#)
- [HTMLIFrameElement](#)
- [HTMLImageElement](#)
- [HTMLInputElement](#)
- [HTMLLabelElement](#)
- [HTMLLegendElement](#)
- [HTMLMarqueeElement](#)
- [HTMLObjectElement](#)
- [HTMLOptionElement](#)
- [HTMLSelectElement](#)
- [HTMLSpanElement](#)
- [HTMLTableElement](#)
- [HTMLTextAreaElement](#)

2.3.1.23 dataPageSize

dataPageSize of type `long`

Sets or retrieves the number of records displayed in a table bound to a data source.

Use the **firstPage** and **lastPage** methods to navigate directly to the first and last pages of a data-bound table, respectively. Use the **nextPage** and **previousPage** methods to navigate sequentially through the pages of a data-bound table.

This attribute extends the [HTMLTableElement](#) interface.

2.3.1.24 dataSrc

dataSrc of type `DOMString`

Sets or retrieves the source of the data for data binding.

Tabular and single-valued data consumers use the **dataSrc** property to specify a binding. The property takes a string that corresponds to the unique identifier of a data source object (DSO) on the page. The string must be prefixed by a number sign (#).

When the **dataSrc** property is applied to a tabular data consumer, the entire data set is repeated by the consuming elements.

When the **dataSrc** property is applied to a table, any contained single-valued consumer objects that specify a **dataFld** property are repeated for each record in the supplied data set. To complete the binding, the binding agent interrogates the enclosing table for its data source. A tabular data consumer contained within another tabular data consumer (table) must specify an explicit **dataSrc**.

This attribute extends the following interfaces:

- [HTMLAnchorElement](#)
- [HTMLButtonElement](#)
- [HTMLDivElement](#)
- [HTMLFrameElement](#)
- [HTMLIFrameElement](#)
- [HTMLImageElement](#)
- [HTMLInputElement](#)
- [HTMLLabelElement](#)
- [HTMLLegendElement](#)
- [HTMLMarqueeElement](#)
- [HTMLObjectElement](#)
- [HTMLOptionElement](#)
- [HTMLSelectElement](#)
- [HTMLSpanElement](#)
- [HTMLTableElement](#)
- [HTMLTextAreaElement](#)

2.3.1.25 direction

direction of type `DOMString`

Sets or retrieves the direction in which the text should scroll.

Possible values are:

- `left` – Default. Marquee scrolls left.
- `right` – Marquee scrolls right.
- `down` – Marquee scrolls down.
- `up` – Marquee scrolls up.

This attribute extends the [HTMLMarqueeElement](#) interface.

2.3.1.26 **dynsrc**

dynsrc of type `DOMString`

Sets or retrieves the address of a video clip or VRML world to display in the window.

This attribute extends the following interfaces:

- [HTMLImageElement](#)
- [HTMLInputElement](#)

2.3.1.27 **frameBorder**

frameBorder of type `DOMString`

Sets or retrieves whether to display a border for the frame.

This attribute extends the [HTMLFrameSetElement](#) interface.

2.3.1.28 **frameSpacing**

frameSpacing of type `DOMString`

Sets or retrieves the amount of additional space, in pixels, between the frames.

Possible values are:

- `1` – Default. Inset border is drawn.
- `0` – No border is drawn.
- `no` – No border is drawn.
- `yes` – Inset border is drawn.

Be aware that invalid settings default to displaying borders.

The amount of space defined for **frameSpacing** does not include the width of the frame border. Frame spacing can be set on one or more **frameSet** objects and applies to all contained **frameSet** objects, unless the contained object defines a different frame spacing.

This attribute extends the following interfaces:

- [HTMLFrameElement](#)
- [HTMLFrameSetElement](#)
- [HTMLIFrameElement](#)

2.3.1.29 height

height of type `DOMString`

Sets or retrieves the height of the object.

Possible values are:

- `height` - Integer that specifies the height of the object, in pixels.
- `percentage` - Integer, followed by a percent sign (%). The value is a percentage of the height of the parent object.

Percentage values are based on the height of the parent object.

When scripting the height property, use either the **pixelHeight** or **posHeight** property to numerically manipulate the height value.

If dynamic changes are intended for the height and width, the original values should be set through style (for example, "`style=height:200px; width:200px`") rather than through the height and width attributes.

This property specifies the calculated height of the object, in pixels. For table rows and table cells, this property has a range of 0 to 32750 pixels.

This attribute extends the following interfaces:

- [HTMLFrameElement](#)
- [HTMLImageElement](#)
- [HTMLMarqueeElement](#)
- [HTMLTableElement](#)
- [HTMLTableRowElement](#)

2.3.1.30 href

href of type `DOMString`

Sets or retrieves a destination URL or an anchor point.

For Windows® Internet Explorer® 8 or later, when the browser is in Internet Explorer 8 mode, the value of the **href** depends on the context of the reference to the attribute. When read as a Document Object Model (DOM) attribute, **href** returns a URL relative to the domain hosting the webpage. When the page is displayed in an earlier document compatibility mode, or the page is viewed with an earlier version of the browser, **href** is read as a content attribute and returns the value specified by the page author

For Internet Explorer 8 or later, when Protected Mode is enabled and a webpage contains an anchor link that has a named target, Windows® Internet Explorer® opens the target of the link in a new window when the target has a different integrity level than the webpage that contains the link.

The **img** element does not support the **href** content attribute. In addition, the **href** property is read-only for the **img** DOM object.

If **href** is specified as a blank value (`href=""` or `href=`), executing the link might display the directory that contains the current page, or it might generate an error, depending on other elements on the webpage and the server environment.

This attribute extends the [HTMLImageElement](#) interface.

2.3.1.31 hspace

hspace of type `long`

Sets or retrieves the horizontal margin for the object.

The **hspace** attribute is similar to the [border](#) attribute, except the margins do not have color when the element is a link.

When the **hspace** attribute is set dynamically for an **img** or **iframe** object, the property value will be updated but the display will show no visible change.

This attribute extends the following interfaces:

- [HTMLIFrameElement](#)
- [HTMLInputElement](#)
- [HTMLMarqueeElement](#)

2.3.1.32 leftMargin

leftMargin of type `DOMString`

Sets or retrieves the left margin for the entire body of the page.

The default value is 10 pixels. If the value is an empty string, the left margin is the left edge.

The value set on the property overrides the default margin.

By default, when the value of this property is set, the opposite margin is set to the same value.

The **leftMargin** attribute extends the [HTMLBodyElement](#) interface.

2.3.1.33 loop

loop of type `long`

Sets or retrieves the number of times a marquee, sound, or video clip will loop.

When defining the **marquee** element, possible values are:

- 0, -1 - Loops infinitely. The default value is -1.
- count - Number of times to loop

In each of the following boundary cases, the marquee loops infinitely:

- `<MARQUEE SCROLLAMOUNT=30 LOOP>This is some scrolling text.</MARQUEE>`
- `<MARQUEE SCROLLAMOUNT=30 LOOP=0>This is some scrolling text.</MARQUEE>`
- `<MARQUEE SCROLLAMOUNT=30 LOOP=>This is some scrolling text.</MARQUEE>`

If you set the **loop** property to `null` or `0` in script, a scripting error occurs.

The **Loop** attribute is supported by the [marquee](#) element.

When defining the **image** or **input** elements, possible values are:

- `0`, `-1` - Loops infinitely. The default value is `1`.
- `count` - Number of times to loop

To restart a sound or video clip after changing its **loop** property, set the **src** property or **dynsrc** property to itself.

This attribute extends the following interfaces:

- [HTMLImageElement](#)
- [HTMLInputElement](#)

2.3.1.34 lowsrc

lowsrc of type `DOMString`

Sets or retrieves a lower resolution image to display.

If the **src** property is set in code, the new URL starts loading into the image area and aborts the transfer of any image data that is already loading into the same area. If you want to alter the **lowsrc** property, you must do so before setting the **src** property. If the URL in the **src** property references an image that is not the same size as the image cell it is loaded into, the source image is scaled to fit.

The **lowsrc** attribute extends the following interfaces:

- [HTMLImageElement](#)
- [HTMLInputElement](#)

2.3.1.35 Methods

Methods of type `DOMString`

Sets or retrieves the comma-separated list of HTTP methods that are supported by the object for public use.

The **Methods** attribute extends the [HTMLAnchorElement](#) interface.

2.3.1.36 name

name of type `DOMString`

Sets or retrieves the name of the object. In the case of **HTMLFrameSetElement**, the **name** property identifies which frame displays the content of a linked document.

The **name** attribute extends the [HTMLFrameSetElement](#) interface.

2.3.1.37 noResize

noResize of type `boolean`

Sets or retrieves whether the user can resize the frame.

Returns `false` if the user can resize the frame, or `true` otherwise. The default value is `false`.

The **noResizes** attribute extends the [HTMLIFrameElement](#) interface.

2.3.1.38 noWrap

noWrap of type `boolean`

Sets or retrieves whether the browser automatically performs word wrap.

Returns `false` if the browser automatically wraps the text, or `true` otherwise. The default value is `false`.

Be aware of the following when using the **noWrap** property in conjunction with the **width** attribute of **table** or **td** elements:

- **Wordwrap** still occurs in a **td** element that has its **width** attribute set to a value smaller than the unwrapped content of the cell, even if the **noWrap** property is set to `true`. Therefore, the **width** attribute takes precedence over the **noWrap** property in this scenario.
- If a **td** element has its **noWrap** set to `true` and the **width** attribute of its **table** element is set to a smaller dimension than the rendered content of the **td** element, **wordwrap** does not occur. In this case, the **noWrap** setting takes precedence over the **width** attribute.

The **noWrap** attribute extends the following interfaces:

- [HTMLBodyElement](#)
- [HTMLDDElement](#)
- [HTMLDivElement](#)
- [HTMLDTElement](#)

2.3.1.39 rightMargin

rightMargin of type `DOMString`

Sets or retrieves the right margin for the entire body of the page.

If the value is an empty string, the right margin is on the right edge.

The value set on the property overrides the default margin.

By default, when you set the value of this property, the opposite margin is set to the same value. The property has a default value of 10.

This attribute extends the [HTMLBodyElement](#) interface.

2.3.1.40 scroll

scroll of type `DOMString`

Sets or retrieves a value that indicates whether the scroll bars are turned on or off.

Possible values are:

- `yes` – Default. Scroll bars are turned on.
- `no` – Scroll bars are turned off.
- `auto` – Scroll bars are displayed when the page content exceeds the client area.

When you use the !DOCTYPE declaration to specify standards-compliant mode, this attribute applies to the **html** element. When standards-compliant mode is not specified, as with earlier versions of Windows® Internet Explorer®, this attribute applies to the **body** element, not the **html** element.

This attribute extends the [HTMLBodyElement](#) interface.

2.3.1.41 scrollAmount

scrollAmount of type `long`

Sets or retrieves the number of pixels that the text scrolls between each subsequent rendering of the marquee. The default value is 6 pixels.

This attribute extends the [HTMLMarqueeElement](#) interface.

2.3.1.42 scrollDelay

scrollDelay of type `long`

Sets or retrieves the speed of the text scroll in a marquee. The default value is 85 milliseconds.

This attribute extends the [HTMLMarqueeElement](#) interface.

2.3.1.43 start

start of type `DOMString`

Sets or retrieves the time that a video clip file should begin playing.

Possible values are:

- `fileopen` – Default. Video begins as soon as it starts loading.
- `mouseover` – Video begins when the user moves the mouse over the animation.

The start property applies only to **img** elements with the [dynsrc](#) property specified.

This attribute extends the following interfaces:

- [HTMLImageElement](#)
- [HTMLInputElement](#)

2.3.1.44 **tabStop**

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

tabStop of type `boolean`

Set to `true` to enable an element behavior to receive focus and participate in tabbing sequence; or `false` otherwise. The default value is `false`.

This attribute is defined by the [HTCElementBehaviorDefaults](#) interface.

2.3.1.45 **topMargin**

topMargin of type `DOMString`

Sets or retrieves the margin for the top of the page.

The default value is 15 pixels. If the value is an empty string or is set to 0, the top margin is set on the top edge of the window or frame.

The specified value overrides the default margin.

By default, when the value of this property is set, the opposite margin is set to the same value.

This attribute extends the [HTMLBodyElement](#) interface.

2.3.1.46 **trueSpeed**

trueSpeed of type `boolean`

Sets or retrieves whether the position of the marquee is calculated using the [scrollDelay](#) and [scrollAmount](#) properties and the actual time elapsed from the last clock tick.

Possible values are:

- `false` – Default. Marquee computes movement based on 60-millisecond ticks of the clock. This means every **scrollDelay** value under 60 is ignored, and the marquee advances the amount of **scrollAmount** each 60 milliseconds. For example, if **scrollDelay** is 6 and **scrollAmount** is 10, the marquee advances 10 pixels every 60 milliseconds.
- `true` - Marquee advances the pixel value of **scrollAmount** by the number of milliseconds set for **scrollDelay**. For example, the marquee would advance 10 pixels for every 6 milliseconds if **scrollDelay** is 6, **scrollAmount** is 10, and the value of **trueSpeed** is true.

The **trueSpeed** attribute indicates that the exact **scrollDelay** value specified is used to move the marquee text. If **trueSpeed** is `false`, all **scrollDelay** values of 59 or less are rounded up to 60 milliseconds.

This attribute extends the [HTMLMarqueeElement](#) interface.

2.3.1.47 **urn**

urn of type `DOMString`

Sets or retrieves a Uniform Resource Name (URN) for a target document.

URNs are an adjunct to URLs. A URL, which is the address used on the World Wide Web, specifies a particular file on a particular machine. A URN specifies the identity of a resource instead of its location.

This attribute extends the [HTMLAnchorElement](#) interface.

urn of type `DOMString`

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

The URN that uniquely identifies an HTC. This value enables events to be uniquely identified when multiple behaviors may fire events of the same name. When an event is fired, the event object's **srcUrn** property is set to the URN of the behavior that fired the event.

This attribute is used by the [PUBLIC:COMPONENT](#) element and can be accessed with the [HTCComponentElement](#) interface.

2.3.1.48 vAlign

vAlign of type `DOMString`

Sets or retrieves whether the caption appears at the top or bottom of the table.

Possible values are:

- `top` – Default. Places the caption at the top of the table.
- `bottom` – Places the caption at the bottom of the table.

When this property is set dynamically, the value of the property will be updated, but there will be no visible change in the display.

This attribute extends the [HTMLTableCaptionElement](#) interface.

2.3.1.49 value

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

value of type `any`

Sets or retrieves a value associated with the [PUBLIC:PROPERTY](#) element.

This attribute is defined by the [HTCPropertyElement](#) interface.

2.3.1.50 viewInheritStyle

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

viewInheritStyle of type `boolean`

Returns `true` if a viewlink should inherit styles from the primary document; or `false` otherwise. The default value is `true`.

This attribute is defined by the [HTCElementBehaviorDefaults](#) interface.

2.3.1.51 viewMasterTab

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

viewMasterTab of type `boolean`

Returns `true` if the master element of a viewlink is included in the tab sequence of the primary document; or `false` otherwise. The default value is `true`.

This attribute is defined by the [HTCElementBehaviorDefaults](#) interface.

2.3.1.52 vspace

vspace of type `long`

Sets or retrieves the vertical margin, in pixels, for the object.

This property is similar to the **border** property, except the margins do not have color when the object is a link.

When this property is set dynamically for an **img** or **iframe** element, the property value will be updated but the display will show no visible change.

This attribute extends the following interfaces:

- [HTMLIFrameElement](#)
- [HTMLInputElement](#)
- [HTMLMarqueeElement](#)

2.3.1.53 width

width of type `DOMString`

Retrieves the width of the object, in pixels.

Possible values are:

- `width` – Integer that specifies the width of the object, in pixels, or as a percentage.
- `percentage` – String that specifies an integer value followed by a %. The value is a percentage of the width of the parent object.

If you specify the **width** property of an **img** element, but not the **height** property, the resulting height of the image is sized proportionally to the specified **width** property and the actual height, in pixels, of the source image file.

If you specify the **width** property of an **img**, and the height and width of the image in the source file are identical, the height of the image matches the width.

If dynamic changes are intended for the height and width, the original values should be set through style (for example, "`style='height:200px; width:200px'`") rather than through the **height** and **width** attributes.

The **width** attribute extends the following interfaces:

- [HTMLFrameElement](#)

- [HTMLInputElement](#)
- [HTMLMarqueeElement](#)

2.3.1.54 wrap

wrap of type `DOMString`

Sets or retrieves how to handle word wrapping in the object.

Possible values are:

- `soft` – Default. Text is displayed wrapped and submitted without CR/LF characters.
- `hard` – Text is displayed wrapped and submitted with CR/LF characters.
- `off` – Word wrapping is disabled. The lines appear exactly as the user types them.

This attribute extends the [HTMLTextAreaElement](#) interface.

2.3.2 DOM Attributes Only

This section contains a list of attributes that are accessible only through the object model.

2.3.2.1 alinkColor

alinkColor of type `DOMString`

Sets or gets the color of all active links in the document.

Possible values are the color name or RGB value. For more information, see [\[CSS-Level2-2009\]](#) section 4.3.6 Colors. Returns a six-digit hex value. The default value is #0000FF.

This is a DOM attribute only.

The **alinkColor** attribute extends the [HTMLDocument](#) interface.

2.3.2.2 altHtml

altHtml of type `DOMString`

Returns the alternative HTML markup that would be rendered if the object failed to load.

During fallback, the object fires the **onerror** event to determine if the event handler can process the alternative HTML. The default action is to replace the failed object with the alternative HTML.

This DOM attribute is write-only.

This attribute extends the [HTMLObjectElement](#) interface.

2.3.2.3 contentWindow

contentWindow of type `Window`

Retrieves the window object of the specified **frame** or **iframe**.

This DOM attribute is read-only.

This attribute extends the [HTMLFrameElement](#) and [HTMLIFrameElement](#) interface.

2.3.2.4 **dir**

dir of type `DOMString`

Sets or retrieves a value that indicates the reading order of the object.

Possible values are:

- `ltr` – Default. Content flows from left to right.
- `rtl` – Content flows from right to left.

Unless explicitly set, the **dir** property has no return value when accessed in script.

The **dir** property does not affect alphanumeric characters in Latin documents. These characters always render ltr. However, the property does affect punctuation characters in Latin documents. For example, punctuation marks such as periods and question marks will render to the left of a sentence when the **dir** property is set to `rtl`.

The value of **dir** property has no effect on the orientation of coordinates for an object's positioning properties. For example, the **left** property and the **right** property perform the same placement in both cases. However, when both the **left** and **right** properties are specified, the **left** property takes precedence when the **dir** property is set to `ltr`. Likewise, the **right** property takes precedence when the **dir** property is set to `rtl`.

This is a DOM attribute only.

This attribute extends the [HTMLDocument](#) interface.

2.3.2.5 **encoding**

encoding of type `DOMString`

Sets or retrieves the MIME encoding for the form.

As defined in [\[HTML\]](#), this is a DOM attribute only and reflects the value of the **enctype** attribute in the markup.

This attribute has a default value of `application/x-www-form-urlencoded`.

This attribute extends the [HTMLFormElement](#) interface.

2.3.2.6 **fgColor**

fgColor of type `DOMString`

Sets or retrieves the foreground (text) color of the document.

Possible values are the color name or RGB value. For more information, see [\[CSS-Level2-2009\]](#) section 4.3.6 Colors. Returns a six-digit hex value. The default value is `#000000`.

This is a DOM attribute only.

This attribute extends the [HTMLDocument](#) interface.

2.3.2.7 fileCreatedDate

fileCreatedDate of type `DOMString`

Retrieves the date the file was created.

This DOM attribute is read-only.

This attribute extends the [HTMLImageElement](#) interface.

2.3.2.8 fileModifiedDate

fileModifiedDate of type `DOMString`

Retrieves the date the file was last modified.

This DOM attribute is read-only.

This attribute extends the [HTMLImageElement](#) interface.

2.3.2.9 fileSize

fileSize of type `DOMString`

Retrieves the file size.

This DOM attribute is read-only.

This attribute extends the [HTMLImageElement](#) interface.

2.3.2.10 fileUpdatedDate

fileUpdatedDate of type `DOMString`

Retrieves the date the file was last updated.

This DOM attribute is read-only.

This attribute extends the [HTMLImageElement](#) interface.

2.3.2.11 hash

hash of type `DOMString`

Sets or retrieves the subsection of the [href](#) property that follows the number sign (#). If there is no number sign, this property returns an empty string.

This DOM attribute is read-only.

This attribute extends the following interfaces:

- [HTMLAnchorElement](#)
- [HTMLAreaElement](#)

2.3.2.12 host

host of type `DOMString`

Sets or retrieves the [hostname](#) and [port](#) number of the location or URL.

The **host** property is the concatenation of the **hostname** and **port** properties, separated by a colon (hostname:port). When the **port** property is `null`, the **host** property is the same as the **hostname** property.

The **host** property may be set at any time, although it is safer to set the **href** property to change a location. If the specified host cannot be found, an error is returned.

This DOM attribute is read-only.

This attribute extends the following interfaces:

- [HTMLAnchorElement](#)
- [HTMLAreaElement](#)

2.3.2.13 hostname

hostname of type `DOMString`

Sets or retrieves the host name part of the location or URL. If no host name is available, this property returns an empty string.

This is a DOM attribute only.

This attribute extends the following interfaces:

- [HTMLAnchorElement](#)
- [HTMLAreaElement](#)

2.3.2.14 indeterminate

indeterminate of type `boolean`

Set to `true` to indicate that a checkbox is neither checked nor cleared (unchecked); or `false` otherwise. The default value is `false`.

The **indeterminate** attribute can be used to indicate whether the user has acted on a control. For example, setting **indeterminate** to `true` causes the check box to appear dimmed, indicating neither a checked nor cleared state.

The value of the **indeterminate** attribute acts independently of the values of the **checked** and [status](#) attributes. Creating an indeterminate state is different from disabling the control. A check box in the indeterminate state can still receive the focus. When the user clicks an indeterminate control, the indeterminate state turns off and the check box toggles between cleared and checked as usual.

This is a DOM attribute only. There is no equivalent way of specifying **indeterminate** in the markup.

This attribute extends the [HTMLInputElement](#) interface.

2.3.2.15 isDisabled

isDisabled of type `boolean`

Set to `true` to indicate that the user cannot interact with the object; or `false` otherwise.

This DOM attribute is read-only.

The **isDisabled** attribute extends the [HTMLElement](#) interface.

2.3.2.16 isMultiLine

isMultiLine of type `boolean`

Set to `true` to indicate that the content contains more than one line; or `false` if the content contains exactly one line.

This DOM attribute is read-only.

The **isMultiLine** attribute extends the [HTMLElement](#) interface.

2.3.2.17 linkColor

linkColor of type `DOMString`

Sets or retrieves the color of the document links.

Possible values are the color name or RGB value. For more information, see [\[CSS-Level2-2009\]](#) section 4.3.6 Colors. Returns a six-digit hex value. The default value is `#0000ff`.

The **linkColor** property can be set by using the **body** object's **onload** event. However, you cannot use the **onload** event to set the **link** property.

This is a DOM attribute only.

The **linkColor** attribute extends the [HTMLDocument](#) interface.

2.3.2.18 mimeType

mimeType of type `DOMString`

Retrieves the MIME type for the file.

Returns the registered file type of the image or anchor's target. If the object hasn't loaded yet, the empty string is returned. The file type string returned need not be a standard MIME type.

This is a DOM attribute only.

The **mimeType** attribute extends the following interfaces:

- [HTMLAnchorElement](#)
- [HTMLImageElement](#)

2.3.2.19 nameProp

nameProp of type `DOMString`

Retrieves the file name specified in the `href` or `src` property of the object.

This DOM attribute is read-only.

The **nameProp** attribute extends the following interfaces:

- [HTMLAnchorElement](#)
- [HTMLImageElement](#)

2.3.2.20 object

object of type `any`

Retrieves the contained object. The **object** attribute may return an object reference of any type and is not limited to DOM objects.

This DOM attribute is read-only.

The **object** attribute extends the [HTMLObjectElement](#) interface.

2.3.2.21 parentDocument

parentDocument of type `DOMString`

Returns the document interface of the parent object, or null if the current object is the top-most document.

This is a DOM attribute only.

The **parentDocument** attribute extends the [HTMLDocument](#) interface.

2.3.2.22 pathname

pathname of type `DOMString`

Sets or retrieves the file name or path specified by the object.

This is a DOM attribute only.

The **pathname** attribute extends the following interfaces:

- [HTMLAnchorElement](#)
- [HTMLAreaElement](#)

2.3.2.23 port

port of type `DOMString`

Sets or retrieves the port number associated with a URL.

The port will resolve based on the default port for the protocol set in the **href** attribute: 21 for FTP, 80 for HTTP, and so forth. Proprietary protocols that do not require a port may return 0 or an empty string.

This is a DOM attribute only.

The **port** attribute extends the following interfaces:

- [HTMLAnchorElement](#)
- [HTMLAreaElement](#)

2.3.2.24 protocol

protocol of type `DOMString`

Sets or retrieves the protocol portion of a URL.

The **protocol** property specifies how to transfer information from the host to the client.

The **document**, **img**, and **location** objects expose the **protocol** property as read-only. The **location.protocol** property returns the initial substring of a URL, including the first colon (for example, http:). However, the **document.protocol** property returns the expanded text of the protocol acronym. For example, it returns the http protocol as Hypertext Transfer Protocol.

This is a DOM attribute only.

The **protocol** attribute extends the following interfaces:

- [HTMLAnchorElement](#)
- [HTMLAreaElement](#)
- [HTMLImageElement](#)

2.3.2.25 protocolLong

protocolLong of type `DOMString`

Gets the registered long name of the protocol, such as Hypertext Transfer Protocol, used in the URL associated with the anchor.

This is a DOM attribute only.

The **protocolLong** attribute extends the section [HTMLAnchorElement](#) interface.

2.3.2.26 search

search of type `DOMString`

Sets or retrieves the substring of the **href** property that follows the question mark.

The substring that follows the question mark is the query string or form data.

This is a DOM attribute only.

The **search** attribute extends the following interfaces:

- [HTMLAnchorElement](#)
- [HTMLAreaElement](#)

2.3.2.27 status

status of type `any`

Sets or retrieves `true` to indicate that the control is selected; or `false` otherwise. The **status** property of a **textArea** object has a default value of `null`.

Setting the **status** updates the value of the property and causes the [onpropertychange](#) event to fire. However, this change has no visual effect on the object.

This is a DOM attribute only.

The **status** attribute extends the following interfaces:

- [HTMLButtonElement](#)
- [HTMLInputElement](#)
- [HTMLTextAreaElement](#)

2.3.2.28 styleSheet

styleSheet of type `CSSStyleSheet`

Retrieves an interface pointer that provides access to the style sheet object's properties and methods. A `null` value for **p** indicates that there is no style sheet for the object.

This DOM attribute is read-only.

The **styleSheet** attribute extends the [HTMLStyleElement](#) and [HTMLLinkElement](#) interfaces.

2.3.2.29 uniqueID

uniqueID of type `DOMString`

Retrieves an automatically generated, unique identifier for the object.

When you apply this property to the **document** object, the browser automatically generates a new ID that you can assign to an element's **id** property.

A new ID is generated and assigned to the element the first time the property is retrieved. Every subsequent access to the property on the same element returns the same ID.

Note: The unique ID generated is not guaranteed to be the same every time the page is loaded.

This is a DOM attribute only.

The **uniqueID** attribute extends the following interfaces:

- [HTMLAnchorElement](#)
- [HTMLAreaElement](#)
- [HTMLAreasCollection](#)
- [HTMLBlockElement](#)
- [HTMLButtonElement](#)

- [HTMLDDElement](#)
- [HTMLDivElement](#)
- [HTMLDocument](#)
- [HTMLDTElement](#)
- [HTMLFieldSetElement](#)
- [HTMLFormElement](#)
- [HTMLFrameElement](#)
- [HTMLHRElement](#)
- [HTMLInputElement](#)
- [HTMLIsIndexElement](#)
- [HTMLLabelElement](#)
- [HTMLLegendElement](#)
- [HTMLMarqueeElement](#)
- [HTMLMetaElement](#)
- [HTMLObjectElement](#)
- [HTMLOptionElement](#)
- [HTMLSelectElement](#)
- [HTMLSpanElement](#)
- [HTMLTextAreaElement](#)

2.3.2.30 **uniqueNumber**

uniqueNumber of type `DOMString`

Retrieves the element's unique number.

The number is obtained by removing the string prefix from the element's [uniqueID](#).

This is a DOM attribute only.

The **uniqueNumber** attribute extends the following interfaces:

- [HTMLAnchorElement](#)
- [HTMLAreaElement](#)
- [HTMLAreasCollection](#)
- [HTMLBlockElement](#)
- [HTMLButtonElement](#)

- [HTMLDDElement](#)
- [HTMLDivElement](#)
- [HTMLDocument](#)
- [HTMLDTElement](#)
- [HTMLFieldSetElement](#)
- [HTMLFormElement](#)
- [HTMLFrameElement](#)
- [HTMLHRElement](#)
- [HTMLInputElement](#)
- [HTMLIsIndexElement](#)
- [HTMLLabelElement](#)
- [HTMLLegendElement](#)
- [HTMLMarqueeElement](#)
- [HTMLMetaElement](#)
- [HTMLObjectElement](#)
- [HTMLOptionElement](#)
- [HTMLSelectElement](#)
- [HTMLSpanElement](#)
- [HTMLTextAreaElement](#)

2.3.2.31 url

url of type `DOMString`

Sets or retrieves a URL string.

Be aware that in Windows® Internet Explorer® 8, relative URL strings are converted to absolute URLs when retrieved.

This is a DOM attribute only.

The **url** attribute extends the [HTMLMetaElement](#) interface.

2.3.2.32 viewLink

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

viewLink of type `document`

Sets or retrieves the document object that supplies content to the master element.

A viewlink displays its document fragment in the primary document at the location of the master HTML Components (HTC) element. Components declared as a viewlink are encapsulated from the primary document and do not participate in the CSS style cascade or tab order unless allowed to do so by the component author.

This DOM attribute is read-only.

The **viewLink** attribute is defined by the [HTCElementBehaviorDefaults](#) interface.

See also [viewLinkContent](#).

2.3.2.33 vlinkColor

vlinkColor of type `DOMString`

Sets or retrieves the color of the links that the user has visited.

Possible values are the color name or RGB value. For more information, see [\[CSS-Level2-2009\]](#) section 4.3.6 Colors. Returns a six-digit hex value. The default value is #800080.

This is a DOM attribute only.

The **vlinkColor** attribute extends the [HTMLDocument](#) interface.

2.3.3 Content Attributes Only

This section contains a list of attributes that are declared in markup only.

2.3.3.1 application

Determines whether the content of the frame is an HTML Application (HTA), which is exempt from the browser security model.

Can be one of the following values:

- `yes` – All content of the frame or iframe is trusted.
- `no` – Default. Browser security rules for unsafe content are applied.

This attribute is available in markup only for **frame** and **iframe** elements.

2.3.3.2 atomicselection

Specifies whether the element and its contents must be selected as a whole, indivisible unit.

Can be one of the following values:

- `false` – Default. The element content can be selected individually.
- `true` – The element and its content can be selected only as a single unit.

If the document is not in atomic selection mode, the **atomicselection** attribute is ignored. To turn on the atomic selection mode, run the `IDM_ATOMICSELECTION` command with the value of `true`.

The **atomicselection** attribute is available only in markup for all elements.

See also [unselectable](#).

2.3.3.3 event

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

Attribute of the [PUBLIC:ATTACH](#) element that specifies the name of any standard event (prefixed with "on") or one of the following HTML Component events:

- [oncontentready](#) – fires when the master element has been parsed completely
- [oncontentsave](#) – fires before content that is attached to an element is saved or copied
- [ondetach](#) – fires before a behavior is detached from a master element
- [ondocumentready](#) – fires when the primary document has been parsed completely

The following example attaches the `fnInit()` function to the `oncontentready` event:

```
<PUBLIC:COMPONENT tagName=TOOLBAR_BUTTON>
  <PUBLIC:ATTACH event="oncontentready" onevent="fnInit()" />
</PUBLIC:COMPONENT>
```

The **event** attribute is declared in markup only.

2.3.3.4 for

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

Attribute of the [PUBLIC:ATTACH](#) element that identifies the source of the event. The value of **for** may be one of the following:

- `document` – an event of the **document** object
- `element` – Default. An event of the master element to which the behavior is attached.
- `window` – an event of the **window** object.

The following example attaches the **onclick** event to the `ExpandCollapse()` event handler:

```
<PUBLIC:ATTACH EVENT="onclick" ONEVENT="ExpandCollapse()" />
```

The **for** attribute is declared in markup only.

See also [event](#) and [onevent](#).

2.3.3.5 get

String that specifies the function to be called whenever the value of the property is retrieved. Any [PUBLIC:PROPERTY](#) element that specifies a **get** attribute without specifying a [put](#) attribute is a read-only property.

The **get** attribute is declared in markup only.

2.3.3.6 implementation

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

String that specifies the HTML Component (HTC) file that contains the tag definition.

This attribute is required when using the **IMPORT** processing instruction, and must include one of the following values:

- `#default` – an element behavior built into Microsoft Windows® Internet Explorer® 5.5 or later
- `#objectID` – string that specifies the id attribute of an object tag
- `sImplementation`

The **implementation** attribute is declared in markup only.

2.3.3.7 **internalName**

The name by which the property or method is called within the component. This internal name must be declared globally before it can be referenced; otherwise, a scripting error occurs, indicating that the name is undefined. If no internal name is specified, the [name](#) attribute is used instead.

The **internalName** attribute is declared in markup only.

This attribute is used by [PUBLIC:METHOD](#) and [PUBLIC:PROPERTY](#).

2.3.3.8 **lightWeight**

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

Set to `true` to indicate that the HTC file does not contain markup that would need to be parsed and rendered for each custom tag that is parsed in the primary document; or `false` otherwise.

The **lightWeight** attribute is declared in markup only.

This attribute is used by [PUBLIC:COMPONENT](#).

2.3.3.9 **literalContent**

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

String that specifies whether the content contained within the custom element it is to be treated as a data island. Although the HTC may still treat any content as a data island, literal content is not automatically parsed and rendered in the primary document. This attribute is only valid when the HTC defines an element behavior, which requires the use of the [tagName](#) attribute.

The **literalContent** attribute can contain one of the following values:

- `false` – Text and markup within the custom element is parsed and rendered normally.
- `true` – Content within the first opening and first closing tags of the element defined by **tagName** is treated as a data island.
- `nested` – Content between the first opening and last closing tags of the element defined by **tagName** is treated as a data island. This mode allows custom tags to be nested in the primary document but requires the HTC to import the custom element namespace.

The following example demonstrates how to read nested literal content in an HTC. First, the HTC imports the *myns:mytag* custom element into a viewlink, then it sets the body of the component from literal content of the master element.

```

<HTML XMLNS:myns>
  <HEAD>
    <?import namespace="myns" implementation="mytag"/>
  </HEAD>
  <PUBLIC:COMPONENT TAGNAME="mytag" LITERALCONTENT="nested">
    <PUBLIC:DEFAULTS VIEWLINKCONTENT />
    <PUBLIC:ATTACH EVENT="oncontentready" ONEVENT="main()" />
  </PUBLIC:COMPONENT>
  <SCRIPT>
    function main()
    {
      document.body.innerHTML = element.innerHTML;
      .
      .
      .
    }
  </SCRIPT>
</HTML>

```

The **literalContent** attribute is declared in markup only.

This attribute is used by the [PUBLIC:COMPONENT](#) element.

2.3.3.10 namespace

Any string that specifies the namespace used as a prefix to custom tags, or specifies a Uniform Resource Name (URN) that uniquely identifies the namespace.

The **namespace** attribute is required when using the [?IMPORT?](#) processing instruction.

This attribute is declared in markup only.

2.3.3.11 onevent

Attribute of the [PUBLIC:ATTACH](#) element that specifies the event handler script or function name.

The following example attaches the `fnInit()` function to the `oncontentready` event:

```

<PUBLIC:COMPONENT tagName=TOOLBAR_BUTTON>
  <PUBLIC:ATTACH event="oncontentready" onevent="fnInit()" />
</PUBLIC:COMPONENT>

```

The **onevent** attribute is declared in markup only.

2.3.3.12 persist

Set to `true` to persist the property as part of the page; or `false` otherwise.

The **persist** attribute is declared in markup only.

This attribute is used by the [PUBLIC:PROPERTY](#) element.

2.3.3.13 put

String that specifies the function to be called when the value of the property is set. A `PUBLIC:PROPERTY` element that specifies both a [get](#) and **put** attribute is a read/write property.

The **put** attribute is declared in markup only.

This attribute is used by the [PUBLIC:PROPERTY](#) element.

2.3.3.14 supportsEditMode

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

Set to `true` to specify whether the content within the HTC is editable; or `false` otherwise.

The **supportsEditMode** attribute is declared in markup only.

This attribute is used by the [PUBLIC:COMPONENT](#) element.

2.3.3.15 tagName

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

String that specifies the name of the custom tag, which is defined in the HTC and imported into the primary document. This attribute is only valid for an HTC file that defines an element behavior.

The **tagName** attribute is declared in markup only.

This attribute is used by the [PUBLIC:COMPONENT](#) element.

2.3.3.16 unselectable

Specifies that an element cannot be selected.

Note: Setting the **unselectable** attribute to `off` does not ensure that an element is selectable. One example is an HTML Application (HTA) with the **selection** attribute set to `no`. Elements in the body of the HTA cannot be selected, even if the **unselectable** attribute for an element is set to `off`.

When you click an element with the **unselectable** attribute set to `on`, any existing current selection is not destroyed.

An element with the **unselectable** attribute set to `on` can be included in a selection that starts somewhere outside the element.

The **unselectable** attribute is implemented as an expando. Setting the **expando** property of the **document** object to `false` precludes the functionality of all expandos.

The **unselectable** attribute is declared in markup only for all elements.

See also [atomicselection](#).

2.3.3.17 viewLinkContent

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

Set to `true` to use the markup in the .htc file is used as the viewlink; or `false` otherwise.

Here is a simple .htc file that uses a declaration to set the viewlink in the PUBLIC:DEFAULTS element.

```
<!-- "Toolbar_Button.htc" -->
<PUBLIC:COMPONENT tagName="TOOLBAR_BUTTON">
  <PUBLIC:DEFAULTS viewLinkContent="true"/>
</PUBLIC:COMPONENT>

<BODY>
  Someday this will be a toolbar button.
</BODY>
```

Setting this attribute in the PUBLIC:DEFAULTS element causes the current HTC markup to be saved as a document fragment in the **HTCElementBehaviorDefaults::viewLink** property.

The **viewLinkContent** attribute is declared in markup only by the [PUBLIC:DEFAULTS](#) element.

See also [viewInheritStyle](#) and [viewMasterTab](#).

2.3.3.18 xmlns

Declares a namespace for custom tags in an HTML document.

The **xmlns** attribute is declared in markup only.

See also [namespace](#).

2.3.4 Event Attributes

This section contains a list of attributes that bind events to event handlers.

2.3.4.1 onactivate

onactivate of type `Function`

Occurs when the object is set as the active element, which is the element that has focus when the parent document has focus. The active element retains focus in the parent document even when focus is shifted from the parent to another application.

To invoke, click an element (other than the active element), use the keyboard to move focus, or invoke the **focus()** method.

- Bubbles: Yes
- Cancelable: No
- Context Info: fromElement, srcElement

This attribute is an extension to [HTMLInputElement](#).

See also [onbeforeactivate](#), [onbeforedeactivate](#).

2.3.4.2 onafterupdate

onafterupdate of type `Function`

Occurs on a databound object after successfully updating the associated data in the data source object.

To invoke, change the data that the object contains.

- Bubbles: Yes
- Cancelable: No
- Context Info: None

This attribute is an extension to [HTMLElement](#).

See also [onbeforeupdate](#).

2.3.4.3 onbeforeactivate

onbeforeactivate of type `Function`

Occurs immediately before the object is set as the active element.

To invoke, click an element (other than the active element), use the keyboard to move focus, or invoke the **focus()** method.

- Bubbles: Yes
- Cancelable: Yes
- Context Info: fromElement, srcElement

This attribute is an extension to [HTMLElement](#).

See also [onactivate](#), [onbeforedeactivate](#).

2.3.4.4 onbeforecopy

onbeforecopy of type `Function`

Fires on the source object before the selection is copied to the system clipboard.

Return `true` from the event handler to disable the **Copy** shortcut.

To invoke, select text and right-click or press Ctrl+C.

- Bubbles: Yes
- Cancelable: Yes
- Context Info: dataTransfer

This attribute is an extension to [HTMLElement](#).

See also [onbeforecut](#), [onbeforepaste](#).

2.3.4.5 onbeforecut

onbeforecut of type `Function`

Occurs on the source object before the selection is deleted from the document.

Return `true` from the event handler to disable the **Cut** shortcut.

To invoke, select text and right-click or press Ctrl+X.

- Bubbles: Yes
- Cancelable: Yes
- Context Info: `dataTransfer`

This attribute is an extension to [HTMLElement](#).

See also [onbeforecopy](#), [onbeforepaste](#).

2.3.4.6 onbeforedeactivate

onbeforedeactivate of type `Function`

Occurs immediately before the active element is changed from the current element to another element in the parent document.

To invoke, click an element (other than the active element), use the keyboard to move focus, or invoke the **focus()** method.

- Bubbles: Yes
- Cancelable: Yes
- Context Info: `fromElement`, `srcElement`

This attribute is an extension to [HTMLElement](#).

See also [onactivate](#), [onbeforeactivate](#).

2.3.4.7 onbeforeeditfocus

onbeforeeditfocus of type `Function`

Occurs before an object contained in an editable element enters a UI-activated state or when an editable container object is control selected.

To invoke, press the Enter key, click an object when it has focus, or double-click an object.

- Bubbles: Yes
- Cancelable: Yes
- Context Info: `fromElement`, `srcElement`

This attribute is an extension to [HTMLElement](#).

2.3.4.8 onbeforepaste

onbeforepaste of type `Function`

Occurs on the target before the selection is pasted from the clipboard into the document.

Return `true` from the event handler to disable the **Paste** shortcut.

To invoke, select text and right-click or press Ctrl+V.

- Bubbles: Yes
- Cancelable: Yes
- Context Info: `dataTransfer`

This attribute is an extension to [HTMLElement](#).

See also [onbeforecopy](#), [onbeforecut](#).

2.3.4.9 onbeforeunload

onbeforeunload of type `Function`

Occurs prior to a page being unloaded.

To invoke, close the browser window; navigate to another page by entering a new address, selecting a Favorite, or clicking an anchor that refers to another webpage; click the Back, Forward, Refresh, or Home button; invoke `anchor.click()`, `document.write()`, `document.open()`, `document.close()`, `location.replace()`, `location.reload()`, `window.close()`, `window.navigate()`, or set a new value of `location.href` property; submit a form, or invoke `form.submit()`.

Return a string from the event handler to prompt the user with the option of staying on the current page. The statement that appears in the dialog box ("Are you sure you want to navigate away from this page? . . . Press OK to continue, or Cancel to stay on the current page") cannot be removed or altered.

- Bubbles: Yes
- Cancelable: Yes
- Context Info: None

The **onbeforeunload** attribute extends the following interfaces:

- [HTMLBodyElement](#)
- [HTMLFrameSetElement](#)

2.3.4.10 onbeforeupdate

onbeforeupdate of type `Function`

Occurs on a databound object before updating the associated data in the data source object.

To invoke, change the data that the object contains.

- Bubbles: Yes
- Cancelable: Yes
- Context Info: None

This attribute is an extension to [HTMLElement](#).

See also [onafterupdate](#).

2.3.4.11 onbounce

onbounce of type `Function`

Occurs when the contents of a marquee element begin to scroll in the opposite direction.

To invoke, set the **behavior** attribute of the [marquee](#) to `alternate`.

- Bubbles: No
- Cancelable: Yes
- Context Info: None

The **onbounce** attribute extends the [HTMLMarqueeElement](#) interface.

2.3.4.12 oncellchange

oncellchange of type `Function`

Occurs when the data changes in the data provider.

To invoke, cause the data in the data source to change.

- Bubbles: Yes
- Cancelable: No
- Context Info: dataFld, qualifier, recordset

This attribute is an extension of [HTMLElement](#).

See also [onafterupdate](#), [onbeforeupdate](#).

2.3.4.13 oncontentready

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

oncontentready of type `Function`

Occurs when the content of the master element, to which a behavior is attached, has been parsed.

If the [literalContent](#) attribute of the [PUBLIC:COMPONENT](#) element is `true`, the literal content of the element behavior is stored in the **innerHTML** property.

The following example demonstrates how to attach and respond to the **oncontentready** event.

```
<PUBLIC:ATTACH EVENT="oncontentready" ONEVENT="show_innerHTML()" />

<SCRIPT LANGUAGE="JScript">
function show_innerHTML()
{
    window.alert ('innerHTML = ' + element.innerHTML);
}
</SCRIPT>
```

This event is specific to HTC and element behaviors.

2.3.4.14 **oncontentsave**

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

oncontentsave of type `Function`

Occurs just before the content of a master element is saved or copied.

This event is specific to HTC and element behaviors

2.3.4.15 **oncontrolselect**

oncontrolselect of type `Function`

Occurs when the user is about to make a control selection in an editable region.

To invoke, select the control.

- Bubbles: Yes
- Cancelable: Yes
- Context Info: `altKey`, `shiftKey`

This attribute is an extension of [HTMLElement](#).

See also [onselect](#), [onselectstart](#).

2.3.4.16 **oncopy**

oncopy of type `Function`

Occurs on the source element when the user copies the object or selection, adding it to the system clipboard.

To invoke, right-click and select **Copy** from the shortcut menu, or press Ctrl+C.

- Bubbles: Yes
- Cancelable: Yes
- Context Info: `dataTransfer`

This attribute is an extension of [HTMLElement](#).

See also [onbeforecopy](#), [oncut](#), [onpaste](#).

2.3.4.17 **oncut**

oncut of type `Function`

Occurs on the source element when the object or selection is removed from the document and added to the system clipboard.

To invoke, right-click and select **Cut** from the shortcut menu, or press Ctrl+X.

- Bubbles: Yes
- Cancelable: Yes
- Context Info: dataTransfer

This attribute is an extension of [HTMLElement](#).

See also [onbeforecut](#), [oncopy](#), [onpaste](#).

2.3.4.18 ondataavailable

ondataavailable of type `Function`

Occurs periodically as data arrives from data source objects that asynchronously transmit data.

To invoke, request new data from the data source.

- Bubbles: Yes
- Cancelable: No
- Context Info: recordset

This attribute is an extension to [HTMLElement](#).

See also [ondatasetchanged](#), [ondatasetcomplete](#).

2.3.4.19 ondatasetchanged

ondatasetchange of type `Function`

Occurs when the data set exposed by a data source object changes.

To invoke, make initial data available from a data source object, expose a different data set, or perform a filter operation.

- Bubbles: Yes
- Cancelable: No
- Context Info: qualifier, reason, recordset

This attribute is an extension to [HTMLElement](#).

See also [ondataavailable](#), [ondatasetcomplete](#).

2.3.4.20 ondatasetcomplete

ondatasetcomplete of type `Function`

Occurs when all data is available from the data source object.

To invoke, allow data set change to complete.

- Bubbles: Yes
- Cancelable: No

- Context Info: qualifier, reason, recordset

This attribute is an extension of [HTMLElement](#).

See also [ondataavailable](#), [ondatasetchanged](#).

2.3.4.21 **ondeactivate**

ondeactivate of type `Function`

Occurs when the active element is changed from the current object.

To invoke, click an element (other than the active element), use the keyboard to move focus, or invoke the `focus()` method.

- Bubbles: Yes
- Cancelable: Yes
- Context Info: `fromElement`, `srcElement`

This attribute is an extension to [HTMLElement](#).

See also [onactivate](#), [onbeforedeactivate](#).

2.3.4.22 **ondetach**

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

ondetach of type `Function`

Occurs before a behavior is detached from an element.

This event provides an opportunity for a dynamically attached behavior to stop receiving notifications from the page. Any behavior that attaches with [PUBLIC:ATTACH](#) will automatically stop receiving notifications when the behavior is detached.

This event is specific to HTC and element behaviors.

2.3.4.23 **ondocumentready**

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

ondocumentready of type `Function`

Occurs when the primary document that contains the behavior has been parsed completely.

This event is specific to HTC and element behaviors.

2.3.4.24 **onerrorupdate**

onerrorupdate of type `Function`

Occurs when an error occurs while updating the associated data in the data source object.

To invoke, cancel the data transfer, or return `false` from `onbeforeupdate` event handler.

- Bubbles: Yes

- Cancelable: No
- Context Info: None

This attribute is an extension to [HTMLElement](#).

See also [onbeforeupdate](#), [onafterupdate](#).

2.3.4.25 onfilterchange

onfilterchange of type `Function`

Occurs when a visual filter changes state or completes a transition.

To invoke, change the filter state.

- Bubbles: No
- Cancelable: No
- Context Info: srcFilter

This attribute is an extension to [HTMLElement](#).

2.3.4.26 onfinish

onfinish of type `Function`

Occurs when marquee looping is complete.

To invoke, specify a value for the **loop** attribute of the marquee.

- Bubbles: No
- Cancelable: Yes
- Context Info: None

The **onfinish** attribute extends the [HTMLMarqueeElement](#) interface.

2.3.4.27 onfocusin

onfocusin of type `Function`

Occurs just prior to setting focus on an element.

To invoke, click an element or document when the element or document does not have focus; use the keyboard to move focus; invoke the **focus** method.

- Bubbles: Yes
- Cancelable: No
- Context Info: fromElement

This attribute is an extension to [HTMLElement](#).

See also [onfocusout](#).

2.3.4.28 onfocusout

onfocusout of type `Function`

Occurs immediately after moving focus to another element.

To invoke, click an element or document when the element or document does not have focus; use the keyboard to move focus; invoke the **focus** method.

- Bubbles: Yes
- Cancelable: No
- Context Info: toElement

This attribute is an extension to [HTMLElement](#).

See also [onfocusin](#).

2.3.4.29 onhelp

onhelp of type `Function`

Occurs when the user presses the F1 key while the browser is the active window.

To invoke, press the F1 key.

- Bubbles: Yes
- Cancelable: Yes
- Context Info: srcElement

This attribute is an extension to [HTMLElement](#).

2.3.4.30 onlosecapture

onlosecapture of type `Function`

Occurs when an object loses the mouse capture.

To invoke, set mouse capture to a different object, change the active window, or invoke the **releaseCapture** method on the document or object.

- Bubbles: No
- Cancelable: No
- Context Info: None

This attribute is an extension to [HTMLElement](#).

2.3.4.31 onmouseenter

onmouseenter of type `Function`

Occurs when the user moves the mouse pointer into the object.

To invoke, move the mouse pointer inside the boundaries of an object.

- Bubbles: No
- Cancelable: No
- Context Info: fromElement, toElement

This attribute is an extension to [HTMLElement](#).

See also [onmouseleave](#).

2.3.4.32 onmouseleave

onmouseleave of type `Function`

Occurs when the user moves the mouse pointer outside the object.

To invoke, move the mouse pointer outside the boundaries of an object.

- Bubbles: No
- Cancelable: No
- Context Info: fromElement, toElement

This attribute is an extension to [HTMLElement](#).

See also [onmouseenter](#).

2.3.4.33 onmove

onmove of type `Function`

Occurs when the object is moved.

To invoke, change the absolute position of an object.

- Bubbles: Yes
- Cancelable: No
- Context Info: None

This attribute is an extension to [HTMLElement](#).

2.3.4.34 onmoveend

onmoveend of type `Function`

Occurs when an editable object stops moving.

To invoke, stop moving the editable object.

- Bubbles: Yes
- Cancelable: No

- Context Info: None

This attribute is an extension to [HTMLElement](#).

See also [onmovestart](#).

2.3.4.35 onmovestart

onmovestart of type `Function`

Occurs when an editable object starts to move.

To invoke, start moving the editable object.

- Bubbles: Yes
- Cancelable: No
- Context Info: None

This attribute is an extension to [HTMLElement](#).

See also [onmoveend](#).

2.3.4.36 onpaste

onpaste of type `Function`

Occurs when the user pastes data, transferring the data from the system clipboard to the document.

To invoke after copying or cutting text, right-click and select **Paste** from the shortcut menu, or press Ctrl+V.

- Bubbles: Yes
- Cancelable: Yes
- Context Info: dataTransfer

This attribute is an extension to [HTMLElement](#).

See also [onbeforecopy](#), [onbeforecut](#), [onbeforepaste](#), [oncopy](#), [oncut](#).

2.3.4.37 onpropertychange

onpropertychange of type `Function`

Occurs when a property changes on the object.

To invoke, cause a property to change value.

- Bubbles: No
- Cancelable: No
- Context Info: propertyName

This attribute is an extension to [HTMLElement](#).

2.3.4.38 onresizeend

onresizeend of type `Function`

Occurs when the user finishes changing the dimensions of the object in a control selection.

To invoke, release the sizing handle of the selected control. By default, the object's display area conforms to the coordinates of the sizing handles when the operation is complete.

- Bubbles: Yes
- Cancelable: No
- Context Info: None

This attribute is an extension to [HTMLElement](#).

2.3.4.39 onresizestart

onresizestart of type `Function`

Occurs when the user begins to change the dimensions of the object in a control selection.

To invoke, drag one of the sizing handles of an object in a control selection.

- Bubbles: Yes
- Cancelable: Yes
- Context Info: None

This attribute is an extension to [HTMLElement](#).

2.3.4.40 onrowenter

onrowenter of type `Function`

Occurs when the current row has changed in the data source and new data values are available on the object in the current row.

To invoke, change data values in the current row.

- Bubbles: Yes
- Cancelable: No
- Context Info: qualifier, recordset

This attribute is an extension to [HTMLElement](#).

See also [onrowexit](#).

2.3.4.41 onrowexit

onrowexit of type `Function`

Occurs just before the data source control changes the current row in the object. By default, signals that the row in the databound object is about to be changed.

To invoke, change rows in the data source.

- Bubbles: No
- Cancelable: Yes
- Context Info: qualifier, recordset

This attribute is an extension to [HTMLElement](#).

See also [onrowenter](#).

2.3.4.42 onrowsdelete

onrowsdelete of type `Function`

Occurs when rows are about to be deleted from the recordset, signaling that the rows are about to be deleted.

Invoked when the **delete** method is called on the recordset.

- Bubbles: Yes
- Cancelable: No
- Context Info: qualifier, recordset

This attribute is an extension to [HTMLElement](#).

2.3.4.43 onrowsinserted

rowsinserted of type `Function`

Occurs just after new rows are inserted in the current recordset, signaling that a new row has been inserted.

Invoked when the **AddNew** method (ADO) is called on the current recordset.

- Bubbles: Yes
- Cancelable: No
- Context Info: reason, recordset

This attribute is an extension to [HTMLElement](#).

2.3.4.44 onselect

onselect of type `Function`

Occurs when the current selection changes. By default, moves the selection to a given character and highlights that selection.

To invoke, move the mouse from character to character during a drag selection, or press the SHIFT key while moving the cursor over text.

- Bubbles: No

- Cancelable: Yes
- Context Info: None

The **onselect** attribute extends the [HTMLBodyElement](#) interface

See also [onselectstart](#).

2.3.4.45 onselectstart

onselectstart of type `Function`

Occurs as the object is being selected.

To invoke, begin selecting one or more objects.

- Bubbles: Yes
- Cancelable: Yes
- Context Info: offsetX, offsetY

This attribute is an extension to [HTMLElement](#).

See also [onselect](#).

2.3.4.46 onstart

onstart of type `Function`

Occurs at the beginning of every loop of the marquee.

To invoke, either set the **loop** attribute to 1 or higher for a set number of loops, or omit the **loop** attribute so that the marquee loops indefinitely.

- Bubbles: No
- Cancelable: No
- Context info: None

The **onstart** attribute extends the [HTMLMarqueeElement](#) interface.

2.4 Methods

The following methods are extensions to [\[HTML\]](#):

- [add\(\)](#)
- [item\(\)](#)
- [moveRow\(\)](#)
- [namedItem\(\)](#)
- [remove\(\)](#)
- [setActive\(\)](#)

- [tags\(\)](#)
- [urns\(\)](#)

2.4.1 add

Adds an element to the areas, controlRange, or options collection.

The **add** method extends the [HTMLAreasCollection](#) interface.

Parameters

oElement of type `HTMLElement`

Required. **Element** that specifies the element to add to the collection.

oBefore of type `any`

Optional. **Object** that specifies an element before which to insert, or an **Integer** that specifies the index position in the collection where the element is placed. If no value is given, the object is appended to the end of the collection.

In IE7 Mode, `oBefore` only accepts an **Integer**.

No Return Value

No JScript Error

2.4.2 item

Retrieves an object from various collections, including the **all** collection.

The **item** method extends the following interfaces:

- [HTMLFormElement](#)
- [HTMLSelectElement](#)

All Document Modes (Internet Explorer 8) The `iSubindex` parameter is not used.

The **item** method cannot retrieve `input type=image` elements from a form. To access all elements contained in a form, use the **children** collection.

Parameters

index of type `DOMString`

Specifies the object or collection to retrieve. If this parameter is a number, it is the zero-based index of the object. If this parameter is a string, all objects with matching **name** or **id** properties are retrieved, and a collection is returned if more than one match is made.

iSubindex of type `long`

Optional. Specifies the zero-based index of the object to retrieve when a collection is returned.

Return Value

Returns an object or a collection of objects if successful, or null otherwise.

No JScript Error

2.4.3 moveRow

Moves a table row to a new position. Rows between the `iSource` and `iTarget` positions in the **rows** collection are shifted based on the direction the row moves.

The **moveRow** method extends the following interfaces:

- [HTMLTableElement](#)
- [HTMLTableSectionElement](#)

Parameters

iSource of type `long`

Specifies the index in the **rows** collection of the table row that is moved. The default value is -1.

iTarget of type `long`

Specifies where the row is moved within the **rows** collection. The default value is -1.

Return Value

Object. Returns a reference to the table row that is moved.

No JScript Error

2.4.4 namedItem

Retrieves an object or a collection from a specified collection. The **namedItem** method first searches for an object with a matching `id` attribute. If a match is not found, the method searches for an object with a matching `name` attribute, but only on those elements that are allowed a `name` attribute.

The **namedItem** method extends the following interfaces:

- [HTMLFormElement](#)
- [HTMLSelectElement](#)

All Document Modes (Internet Explorer 8) The **namedItem** method does not return collections if more than one named item is found; instead, it returns the first case-insensitive matched element.

Parameters

sName of type `DOMString`

Specifies the **name** or **id** property of the object to retrieve. A collection is returned if more than one match is made.

Return Value

Returns an object or a collection of objects if successful, or null otherwise.

No JScript Error

2.4.5 remove

Removes an element from the collection.

The **remove** method extends the [HTMLAreasCollection](#) interface.

Parameters

iIndex of type `long`

Specifies the zero-based index of the element to remove from the collection.

No Return Value

No JScript Error

2.4.6 setActive

Sets the object as active without setting focus to the object.

The **setActive** method extends the [HTMLElement](#) interface.

See also [onactivate](#), [onbeforeactivate](#), [onbeforedeactivate](#), [ondeactivate](#).

2.4.7 tags

Retrieves a collection of objects that have the specified HTML tag name.

The **tags** method extends the following interfaces:

- [HTMLAreasCollection](#)
- [HTMLCollection](#)
- [HTMLFormElement](#)
- [HTMLSelectElement](#)

Parameters

sTag of type `DOMString`

Specifies an HTML tag. It can be any one of the **elements** exposed by the DHTML Object Model.

Return Value

Returns a collection of element objects if successful, or null otherwise.

Successful execution includes the case where no elements having the given name are found. In this case, a collection containing zero elements is returned.

The **length** property of the collection contains the number of elements in the collection.

Null may be returned in cases where the collection cannot be constructed, such as inability to allocate memory for even a zero-length collection.

No JScript Error

2.4.8 urns

Quirks Mode, IE7 Mode, IE8 Mode, and IE9 Mode (All Versions)

Retrieves a collection of all objects to which a specified behavior is attached.

The **urns** method extends the [HTMLSelectElement](#) interface.

Parameters

sUrn of type `DOMString`

Specifies the behavior's Uniform Resource Name (URN).

Return Value

Returns a collection of objects if successful, or null otherwise.

The **urns** method returns an empty collection if no element has the specified behavior attached to it.

Use the **length** property on the collection to determine the number of elements it contains, and the [item](#) method to obtain a particular item in the collection.

No JScript Error

2.5 Collections

The following collections are extensions to [\[HTML\]](#):

- [all](#)
- [cells](#)
- [frames](#)

2.5.1 all

all of type **HTMLCollection** (`IHTMLCollection`)

Returns a reference to the collection of elements contained by the specified object.

2.5.2 cells

cells of type **HTMLCollection** (`IHTMLCollection`)

Retrieves a collection of all cells in the table row or in the entire table.

2.5.3 frames

frames of type **HTMLCollection** (`IHTMLFramesCollection2`)

Retrieves a collection of all window objects defined by the given document or defined by the document associated with the given window.

3 Security Considerations

There are no additional security considerations.

4 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Windows® Internet Explorer® 7
- Windows® Internet Explorer® 8
- Windows® Internet Explorer® 9
- Windows® Internet Explorer® 10

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

5 Change Tracking

This section identifies changes that were made to the [MS-HTML401E] protocol document between the February 2012 and July 2012 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated**.

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact protocol@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1 Introduction	Updated document to remove beta tagging.	N	Content updated.

6 Index

A

[Applicability](#) 18

Attributes

Content Only

[application](#) 81

[atomicselection](#) 81

[event](#) 82

[for](#) 82

[get](#) 82

[implementation](#) 82

[internalName](#) 83

[lightWeight](#) 83

[literalContent](#) 83

[namespace](#) 84

[onevent](#) 84

[persist](#) 84

[put](#) 85

[supportsEditMode](#) 85

[tagName](#) 85

[unselectable](#) 85

[viewLinkContent](#) 85

[xmlns](#) 86

DOM and Content

[action](#) 53

[align](#) 53

[allowTransparency](#) 54

[alt](#) 54

[background](#) 54

[BaseHref](#) 55

[behavior](#) 55

[bgColor](#) 55

[bgProperties](#) 56

[border](#) 56

[borderColor](#) 56

[borderColorDark](#) 57

[borderColorLight](#) 57

[bottomMargin](#) 57

[canHaveHTML](#) 58

[charset](#) 58

[classid](#) 58

[clear](#) 58

[color](#) 59

[cols](#) 59

[dataFld](#) 59

[dataFormatAs](#) 60

[dataPageSize](#) 60

[dataSrc](#) 61

[direction](#) 61

[dynsrc](#) 62

[frameBorder](#) 62

[frameSpacing](#) 62

[height](#) 63

[href](#) 63

[hspace](#) 64

[leftMargin](#) 64

[loop](#) 64

[lowsrc](#) 65

[Methods](#) 65

[name](#) 65

[noResize](#) 66

[noWrap](#) 66

[rightMargin](#) 66

[scroll](#) 67

[scrollAmount](#) 67

[scrollDelay](#) 67

[start](#) 67

[tabStop](#) 68

[topMargin](#) 68

[trueSpeed](#) 68

[urn](#) 68

[vAlign](#) 69

[value](#) 69

[viewInheritStyle](#) 69

[viewMasterTab](#) 70

[vspace](#) 70

[width](#) 70

[wrap](#) 71

DOM Only

[alinkColor](#) 71

[altHtml](#) 71

[contentWindow](#) 71

[dir](#) 72

[encoding](#) 72

[fgColor](#) 72

[fileCreatedDate](#) 73

[fileModifiedDate](#) 73

[fileSize](#) 73

[fileUpdatedDate](#) 73

[hash](#) 73

[host](#) 74

[hostname](#) 74

[indeterminate](#) 74

[isDisabled](#) 75

[isMultiLine](#) 75

[linkColor](#) 75

[mimeType](#) 75

[nameProp](#) 75

[object](#) 76

[parentDocument](#) 76

[pathname](#) 76

[port](#) 76

[protocol](#) 77

[protocolLong](#) 77

[search](#) 77

[status](#) 78

[styleSheet](#) 78

[uniqueID](#) 78

[uniqueNumber](#) 79

[url](#) 80

[viewLink](#) 80

[vlinkColor](#) 81

Event

[onactivate](#) 86

[onafterupdate](#) 86

[onbeforeactivate](#) 87

- [onbeforecopy](#) 87
- [onbeforecut](#) 87
- [onbeforedeactivate](#) 88
- [onbeforeeditfocus](#) 88
- [onbeforepaste](#) 88
- [onbeforeunload](#) 89
- [onbeforeupdate](#) 89
- [onbounce](#) 90
- [oncellchange](#) 90
- [oncontentready](#) 90
- [oncontentsave](#) 91
- [oncontrolselect](#) 91
- [oncopy](#) 91
- [oncut](#) 91
- [ondataavailable](#) 92
- [ondatasetchanged](#) 92
- [ondatasetcomplete](#) 92
- [ondeactivate](#) 93
- [ondetach](#) 93
- [ondocumentready](#) 93
- [onerrorupdate](#) 93
- [onfilterchange](#) 94
- [onfinish](#) 94
- [onfocusin](#) 94
- [onfocusout](#) 95
- [onhelp](#) 95
- [onlosecapture](#) 95
- [onmouseenter](#) 95
- [onmouseleave](#) 96
- [onmove](#) 96
- [onmoveend](#) 96
- [onmovestart](#) 97
- [onpaste](#) 97
- [onpropertychange](#) 97
- [onresizeend](#) 98
- [onresizestart](#) 98
- [onrowenter](#) 98
- [onrowexit](#) 98
- [onrowsdelete](#) 99
- [onrowsinserted](#) 99
- [onselect](#) 99
- [onselectstart](#) 100
- [onstart](#) 100

C

[Change tracking](#) 107

Collections

- [all](#) 104
- [cells](#) 104
- [frames](#) 104

E

[Element behaviors and HTML Components](#) 16

Elements

- [?import?](#) 20
- [comment](#) 21
- [marquee](#) 21
- [noBR](#) 22
- [PUBLIC:ATTACH](#) 22
- [PUBLIC:COMPONENT](#) 22

- [PUBLIC:DEFAULTS](#) 23
- [PUBLIC:EVENT](#) 23
- [PUBLIC:METHOD](#) 23
- [PUBLIC:PROPERTY](#) 24

G

[Glossary](#) 8

I

[Implementer - security considerations](#) 105

[Informative references](#) 9

Interfaces

- [HTCComponentElement](#) 26
- [HTCElementBehaviorDefaults](#) 26
- [HTCPropertyElement](#) 26
- [HTMLAnchorElement](#) 29
- [HTMLAreaElement](#) 30
- [HTMLAreasCollection](#) 31
- [HTMLBlockElement](#) 26
- [HTMLBodyElement](#) 31
- [HTMLButtonElement](#) 32
- [HTMLCollection](#) 32
- [HTMLCommentElement](#) 27
- [HTMLDDElement](#) 28
- [HTMLDivElement](#) 33
- [HTMLDocument](#) 33
- [HTMLDTEElement](#) 28
- [HTMLElement](#) 34
- [HTMLFieldSetElement](#) 35
- [HTMLFormElement](#) 36
- [HTMLFrameElement](#) 36
- [HTMLFrameSetElement](#) 37
- [HTMLHeadingElement](#) 37
- [HTMLHRElement](#) 38
- [HTMLIFrameElement](#) 38
- [HTMLImageElement](#) 39
- [HTMLInputElement](#) 40
- [HTMLIsIndexElement](#) 41
- [HTMLLabelElement](#) 41
- [HTMLLegendElement](#) 41
- [HTMLLinkElement](#) 42
- [HTMLMarqueeElement](#) 28
- [HTMLMetaElement](#) 42
- [HTMLObjectElement](#) 42
- [HTMLOptionElement](#) 43
- [HTMLParagraphElement](#) 43
- [HTMLSelectElement](#) 44
- [HTMLSpanElement](#) 44
- [HTMLStyleElement](#) 44
- [HTMLTableCaptionElement](#) 45
- [HTMLTableCellElement](#) 45
- [HTMLTableElement](#) 45
- [HTMLTableRowElement](#) 46
- [HTMLTableSectionElement](#) 47
- [HTMLTextAreaElement](#) 47

[Introduction](#) 8

M

Methods

[add](#) 101
[item](#) 101
[moveRow](#) 102
[namedItem](#) 102
[remove](#) 103
[setActive](#) 103
[tags](#) 103
[urns](#) 104

N

[Normative references](#) 8

O

[Overview \(synopsis\)](#) 9

P

[Product behavior](#) 106

R

[References](#) 8
[informative](#) 9
[normative](#) 8

S

[Security - implementer considerations](#) 105

T

[Tracking changes](#) 107