Document Object Model (DOM) Level 2 HTML Specification

Version 1.0

W3C Working Draft 07 December 2001

This version:
http://www.w3.org/TR/2001/WD-DOM-Level-2-HTML-20011207
(PostScript file , PDF file , plain text , ZIP file , single HTML file)
Latest version:
http://www.w3.org/TR/DOM-Level-2-HTML
Previous version:
http://www.w3.org/TR/2000/WD-DOM-Level-2-HTML-20001113

Editors:
Johnny Stenback, *Netscape*
Arnaud Le Hors, *W3C team contact until October 1999, then IBM*
Philippe Le Hégaret, *W3C*
Chris Wilson, *Microsoft Corp. (for DOM Level 1 after January 1998)*
Ian Jacobs, *W3C (for DOM Level 1)*
Mike Champion, *ArborText (for DOM Level 1)*
Scott Isaacs, *Microsoft Corp. (for DOM Level 1 until January 1998)*
Vidur Apparao, *Netscape Communications Corp. (for DOM Level 1)*

Copyright ©2001 W3C® (MIT, INRIA, Keio), All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply.

Abstract

This specification defines the Document Object Model Level 2 HTML, a platform- and language-neutral interface that allows programs and scripts to dynamically access and update the content and structure of HTML 4.0 and XHTML 1.0 documents. The Document Object Model Level 2 HTML builds on the Document Object Model Level 2 Core [DOM Level 2 Core].

The DOM Level 2 HTML is made of a set of specific interfaces to manipulate the structure and contents of an HTML document.

Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the W3C.

This is a W3C Last Call Working Draft for review by W3C members and other interested parties. The Last Call review period ends on 7 January 2001. Please send review comments before the review period ends to www-dom@w3.org (archive).

It is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to use W3C Working Drafts as reference material or to cite them as other than "work in progress". This is work in progress and does not imply endorsement by, or the consensus of, either W3C or members of the DOM Working Group or members of the HTML Working Group.

This document has been produced as part of the W3C DOM Activity. The authors of this document are the DOM Working Group members.

A list of current W3C Recommendations and other technical documents can be found at http://www.w3.org/TR.

Table of contents

Expanded Table of Contents 3
Copyright Notice 5

1. Document Object Model HTML 9

Appendix A: Changes 57
Appendix B: IDL Definitions 59
Appendix C: Java Language Binding 69
Appendix D: ECMAScript Language Binding 95
Appendix E: Acknowledgements 117
Glossary 119
References 121
Index 123

Expanded Table of Contents

Expanded Table of Contents 3
Copyright Notice 5
W3C Document Copyright Notice and License 5
W3C Software Copyright Notice and License 6

1. Document Object Model HTML 9
1.1. Introduction 9
1.2. HTML Application of Core DOM 10
1.2.1. Naming Conventions 10
1.3. XHTML and the HTML DOM 11
1.4. Miscellaneous Object Definitions 11
1.5. Objects related to HTML documents 13
1.6. HTML Elements 16
1.6.1. Property Attributes 16
1.6.2. Naming Exceptions 17
1.6.3. Exposing Element Type Names (tagName, (nodeName)) 17
1.6.4. The HTMLElement interface 17
1.6.5. Object definitions 18

Appendix A: Changes 57
A.1. Changes between DOM Level 1 and DOM Level 2 57
A.1.1. Changes to DOM Level 1 interfaces and exceptions 57
A.1.2. New Interfaces 57
Appendix B: IDL Definitions 59
Appendix C: Java Language Binding 69
Appendix D: ECMAScript Language Binding 95
Appendix E: Acknowledgements 117
E.1. Production Systems 117
Glossary 119
References 121
1. Normative references 121
2. Informative references 121
Index 123

Expanded Table of Contents

Copyright Notice

Copyright © 2001 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved.

This document is published under the W3C Document Copyright Notice and License [p.5] . The bindings within this document are published under the W3C Software Copyright Notice and License [p.6] . The software license requires "Notice of any changes or modifications to the W3C files, including the date changes were made." Consequently, modified versions of the DOM bindings must document that they do not conform to the W3C standard; in the case of the IDL definitions, the pragma prefix can no longer be 'w3c.org'; in the case of the Java Language binding, the package names can no longer be in the 'org.w3c' package.

W3C Document Copyright Notice and License

Note: This section is a copy of the W3C Document Notice and License and could be found at http://www.w3.org/Consortium/Legal/copyright-documents-19990405.

Copyright © 1994-2001 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved.

http://www.w3.org/Consortium/Legal/

Public documents on the W3C site are provided by the copyright holders under the following license. The software or Document Type Definitions (DTDs) associated with W3C specifications are governed by the Software Notice. By using and/or copying this document, or the W3C document from which this statement is linked, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions:

Permission to use, copy, and distribute the contents of this document, or the W3C document from which this statement is linked, in any medium for any purpose and without fee or royalty is hereby granted, provided that you include the following on *ALL* copies of the document, or portions thereof, that you use:

1. A link or URL to the original W3C document.
2. The pre-existing copyright notice of the original author, or if it doesn't exist, a notice of the form: "Copyright © [$date-of-document] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. http://www.w3.org/Consortium/Legal/" (Hypertext is preferred, but a textual representation is permitted.)
3. *If it exists*, the STATUS of the W3C document.

When space permits, inclusion of the full text of this **NOTICE** should be provided. We request that authorship attribution be provided in any software, documents, or other items or products that you create pursuant to the implementation of the contents of this document, or any portion thereof.

No right to create modifications or derivatives of W3C documents is granted pursuant to this license. However, if additional requirements (documented in the Copyright FAQ) are satisfied, the right to create modifications or derivatives is sometimes granted by the W3C to individuals complying with those requirements.

THIS DOCUMENT IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE DOCUMENT OR THE PERFORMANCE OR IMPLEMENTATION OF THE CONTENTS THEREOF.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to this document or its contents without specific, written prior permission. Title to copyright in this document will at all times remain with copyright holders.

W3C Software Copyright Notice and License

Note: This section is a copy of the W3C Software Copyright Notice and License and could be found at http://www.w3.org/Consortium/Legal/copyright-software-19980720

Copyright © 1994-2001 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved.

http://www.w3.org/Consortium/Legal/

This W3C work (including software, documents, or other related items) is being provided by the copyright holders under the following license. By obtaining, using and/or copying this work, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions:

Permission to use, copy, and modify this software and its documentation, with or without modification, for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the software and documentation or portions thereof, including modifications, that you make:

1. The full text of this NOTICE in a location viewable to users of the redistributed or derivative work.
2. Any pre-existing intellectual property disclaimers. If none exist, then a notice of the following form: "Copyright © [$date-of-software] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. http://www.w3.org/Consortium/Legal/."

3. Notice of any changes or modifications to the W3C files, including the date changes were made. (We recommend you provide URIs to the location from which the code is derived.)

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to the software without specific, written prior permission. Title to copyright in this software and any associated documentation will at all times remain with copyright holders.

W3C Software Copyright Notice and License

1. Document Object Model HTML

Editors:
Johnny Stenback, Netscape
Arnaud Le Hors, IBM
Philippe Le Hégaret, W3C
Chris Wilson, Microsoft Corp. (for DOM Level 1 after January 1998)
Ian Jacobs, W3C (for DOM Level 1)
Vidur Apparao, Netscape Communications Corp. (for DOM Level 1)
Mike Champion, ArborText (for DOM Level 1)
Scott Isaacs, Microsoft Corp. (for DOM Level 1 until January 1998)

1.1. Introduction

This section extends the DOM Level 2 Core API [DOM Level 2 Core] to describe objects and methods specific to *HTML* [p.119] documents [HTML4.0], and XHTML documents [XHTML 1.0]. In general, the functionality needed to manipulate hierarchical document structures, elements, and attributes will be found in the core section; functionality that depends on the specific elements defined in HTML will be found in this section.

The goals of the HTML-specific DOM API are:

- to specialize and add functionality that relates specifically to HTML documents and elements.
- to address issues of backwards compatibility with the *DOM Level 0* [p.119] .
- to provide *convenience* [p.119] mechanisms, where appropriate, for common and frequent operations on HTML documents.

The key differences between the core DOM and the HTML application of DOM is that the HTML Document Object Model exposes a number of *convenience* [p.119] methods and properties that are consistent with the existing models and are more appropriate to script writers. In many cases, these enhancements are not applicable to a general DOM because they rely on the presence of a predefined DTD. The transitional or frameset DTD for HTML 4.0, or the XHTML 1.0 DTD are assumed. Interoperability between implementations is only guaranteed for elements and attributes that are specified in the HTML 4.0 and XHTML 1.0 DTDs.

More specifically, this document includes the following specializations for HTML:

- An `HTMLDocument` [p.13] interface, derived from the core `Document` interface. `HTMLDocument` specifies the operations and queries that can be made on a HTML document.
- An `HTMLElement` [p.17] interface, derived from the core `Element` interface. `HTMLElement` specifies the operations and queries that can be made on any HTML element. Methods on `HTMLElement` include those that allow for the retrieval and modification of attributes that apply to all HTML elements.
- Specializations for all HTML elements that have attributes that extend beyond those specified in the `HTMLElement` [p.17] interface. For all such attributes, the derived interface for the element contains explicit methods for setting and getting the values.

The DOM Level 2 includes mechanisms to access and modify style specified through CSS and defines an event model that can be used with HTML documents.

The interfaces found within this section are not mandatory. A DOM application may use the `hasFeature(feature, version)` method of the `DOMImplementation` interface with parameter values "HTML" and "2.0" (respectively) to determine whether or not this module is supported by the implementation. In addition to the feature string "HTML", the feature string "XHTML" (version string "2.0") can be used to check if the implementation supports XHTML (this is equivalent to checking the features "XML" and "HTML"). In order to fully support this module, an implementation must also support the "Core" feature defined [DOM Level 2 Core]. Please refer to additional information about *conformance* in the DOM Level 2 Core specification [DOM Level 2 Core].

A DOM application can use the `hasFeature` method of the `DOMImplementation` interface to determine whether they are supported or not. The feature string for all the interfaces listed in this section is "HTML" and the version is "2.0". Note that, since DOM Level 2 HTML is not backward compatible with DOM Level 1 [DOM Level 1], a DOM Level 2 HTML implementation must not claim to implement the feature "HTML" version "1.0". In order to fully support this feature, an implementation needs also to support the "Core" feature defined in the Document Object Model Level 2 Core [DOM Level 2 Core] (see also *Conformance*).

The interfaces in this specification are designed for HTML 4.0 documents, as well as for XHTML 1.0 documents.

1.2. HTML Application of Core DOM

1.2.1. Naming Conventions

The HTML DOM follows a naming convention for properties, methods, events, collections, and data types. All names are defined as one or more English words concatenated together to form a single string.

1.2.1.1. Properties and Methods

The property or method name starts with the initial keyword in lowercase, and each subsequent word starts with a capital letter. For example, a property that returns document meta information such as the date the file was created might be named "fileDateCreated". In the ECMAScript binding, properties are exposed as properties of a given object. In Java, properties are exposed with get and set methods.

1.2.1.2. Non-HTML 4.0 interfaces and attributes

While most of the interfaces defined below can be mapped directly to elements defined in the HTML 4.0 Recommendation, some of them cannot. Similarly, not all attributes listed below have counterparts in the HTML 4.0 specification (and some do, but have been renamed to avoid conflicts with scripting languages). Interfaces and attribute definitions that have links to the HTML 4.0 specification have corresponding element and attribute definitions there; all others are added by this specification, either for convenience or backwards compatibility with *DOM Level 0* [p.119] implementations.

1.3. XHTML and the HTML DOM

The DOM HTML Level 1 API's [DOM Level 1] were originally intended to be used only for HTML 4.0 documents [HTML4.0] and the APIs were defined well before XHTML 1.0 [XHTML 1.0] became a specification, or was even being worked on by the HTML working group. The biggest difference between HTML 4.0 (and earlier) and XHTML 1.0 (from the DOM point of view) is that XHTML is case sensitive, whereas HTML 4.0 is case insensitive. The HTML case insensitivity is also reflected in the DOM HTML API's in some ways, for instance, element and attribute names are exposed as all upper case (for consistency) when used on an HTML document, regardless of the character case used in the markup. In XHTML everything is case sensitive (since XHTML is based on XML) and element and attribute names must be lower case in the markup. Because of this there are certain things that developers need to keep in mind when writing code that should work on both HTML and XHTML documents. There are basically two things that need to be taken into account, one is that when comparing element or attribute names to strings the string compare needs to be case insensitive, or the element or attribute name needs to be converted into lowercase before comparing against a lowercase string. The other thing is that when calling methods that are case insensitive when used on a HTML document (such as `getElementsByTagName()` and `namedItem()`) the string that is passed in should be lower case to work on both HTML and XHTML documents.

1.4. Miscellaneous Object Definitions

Interface *HTMLDOMImplementation* (introduced in **DOM Level 2**)

The `HTMLDOMImplementation` interface extends the `DOMImplementation` interface with a method for creating an HTML document instance. The core `DOMImplementation` interface can be used to create XHTML documents by passing the XHTML namespace as the namespace for the root element.

IDL Definition

```
// Introduced in DOM Level 2:
interface HTMLDOMImplementation : DOMImplementation {
  HTMLDocument       createHTMLDocument(in DOMString title);
};
```

Methods

`createHTMLDocument`

Creates an `HTMLDocument` [p.13] object with the minimal tree made of the following elements: `HTML`, `HEAD`, `TITLE`, and `BODY`.

Parameters

`title` of type `DOMString`

The title of the document to be set as the content of the `TITLE` element, through a child `Text` node.

Return Value

`HTMLDocument` [p.13]	A new `HTMLDocument` object.

No Exceptions

Interface *HTMLCollection*

An `HTMLCollection` is a list of nodes. An individual node may be accessed by either ordinal index or the node's `name` or `id` attributes.

Note: Collections in the HTML DOM are assumed to be *live* [p.119] meaning that they are automatically updated when the underlying document is changed.

IDL Definition

```
interface HTMLCollection {
  readonly attribute unsigned long   length;
  Node               item(in unsigned long index);
  Node               namedItem(in DOMString name);
};
```

Attributes

`length` of type `unsigned long`, readonly

This attribute specifies the length or *size* of the list.

Methods

`item`

This method retrieves a node specified by ordinal index. Nodes are numbered in tree order (depth-first traversal order).

Parameters

`index` of type `unsigned long`

The index of the node to be fetched. The index origin is 0.

Return Value

`Node` The `Node` at the corresponding position upon success. A value of `null` is returned if the index is out of range.

No Exceptions

`namedItem`

This method retrieves a `Node` using a name. It first searches for a `Node` with a matching `id` attribute. If it doesn't find one, it then searches for a `Node` with a matching `name` attribute, but only on those elements that are allowed a name attribute. This method is case insensitive in HTML documents and case sensitive in XHTML documents.

Parameters

`name` of type `DOMString`

The name of the `Node` to be fetched.

Return Value

`Node` The `Node` with a `name` or `id` attribute whose value corresponds to the specified string. Upon failure (e.g., no node with this name exists), returns `null`.

No Exceptions

1.5. Objects related to HTML documents

Interface *HTMLDocument*

An `HTMLDocument` is the root of the HTML hierarchy and holds the entire content. Besides providing access to the hierarchy, it also provides some *convenience* [p.119] methods for accessing certain sets of information from the document.

The following properties have been deprecated in favor of the corresponding ones for the `BODY` element:

- alinkColor
- background
- bgColor
- fgColor
- linkColor
- vlinkColor

Note: In DOM Level 2, the method `getElementById` is inherited from the `Document` interface where it was moved to.

IDL Definition

```
interface HTMLDocument : Document {
           attribute DOMString       title;
  readonly attribute DOMString       referrer;
  readonly attribute DOMString       domain;
  readonly attribute DOMString       URL;
           attribute HTMLElement     body;
  readonly attribute HTMLCollection  images;
  readonly attribute HTMLCollection  applets;
  readonly attribute HTMLCollection  links;
  readonly attribute HTMLCollection  forms;
  readonly attribute HTMLCollection  anchors;
           attribute DOMString       cookie;
                                        // raises(DOMException) on setting

  void               open();
  void               close();
  void               write(in DOMString text);
  void               writeln(in DOMString text);
  NodeList           getElementsByName(in DOMString elementName);
};
```

Attributes

`URL` of type `DOMString`, readonly

The absolute URI [RFC2396] of the document.

`anchors` of type `HTMLCollection` [p.12] , readonly
: A collection of all the anchor (`A`) elements in a document with a value for the `name` attribute.

  Note: For reasons of backwards compatibility, the returned set of anchors only contains those anchors created with the `name` attribute, not those created with the `id` attribute.

`applets` of type `HTMLCollection` [p.12] , readonly
: A collection of all the `OBJECT` elements that include applets and `APPLET` (*deprecated*) elements in a document.

`body` of type `HTMLElement` [p.17]
: The element that contains the content for the document. In documents with `BODY` contents, returns the `BODY` element. In frameset documents, this returns the outermost `FRAMESET` element.

`cookie` of type `DOMString`
: This mutable string attribute denotes persistent state information that (1) is associated with the current frame or document and (2) is composed of information described by the `cookies` non-terminal of [RFC2965], Section 4.2.2.
  If no persistent state information is available for the current frame or document document, then this property's value is an empty string.
  When this attribute is read, all cookies are returned as a single string, with each cookie's name-value pair concatenated into a list of name-value pairs, each list item being separated by a ';' (semicolon).
  When this attribute is set, the value it is set to should be a string that adheres to the `cookie` non-terminal of [RFC2965]; that is, it should be a single name-value pair followed by zero or more cookie attribute values. If no domain attribute is specified, then the domain attribute for the new value defaults to the host portion of an absolute URI [RFC2396] of the current frame or document. If no path attribute is specified, then the path attribute for the new value defaults to the absolute path portion of the URI [RFC2396] of the current frame or document. If no max-age attribute is specified, then the max-age attribute for the new value defaults to a user agent defined value. If a cookie with the specified name is already associated with the current frame or document, then the new value as well as the new attributes replace the old value and attributes. If a max-age attribute of 0 is specified for the new value, then any existing cookies of the specified name are removed from the cookie storage.

  Note: See [RFC2965] for the semantics of persistent state item attribute value pairs.

  Note: The precise nature of a user agent session is not defined by this specification.

  Exceptions on setting

`DOMException`	SYNTAX_ERR: If the new value does not adhere to the cookie syntax specified by [RFC2965].

`domain` of type `DOMString`, readonly
: The domain name of the server that served the document, or `null` if the server cannot be identified by a domain name.

`forms` of type `HTMLCollection` [p.12] , readonly
: A collection of all the forms of a document.

`images` of type `HTMLCollection` [p.12] , readonly
: A collection of all the `IMG` elements in a document. The behavior is limited to `IMG` elements for backwards compatibility.

`links` of type `HTMLCollection` [p.12] , readonly
: A collection of all `AREA` elements and anchor (`A`) elements in a document with a value for the `href` attribute.

`referrer` of type `DOMString`, readonly
: Returns the URI [RFC2396] of the page that linked to this page. The value is an empty string if the user navigated to the page directly (not through a link, but, for example, via a bookmark).

`title` of type `DOMString`
: The title of a document as specified by the `TITLE` element in the head of the document.

Methods

`close`
: Closes a document stream opened by `open()` and forces rendering.

No Parameters

No Return Value

No Exceptions

`getElementsByName`
: Returns the (possibly empty) collection of elements whose `name` value is given by `elementName`. This method is case sensitive.

Parameters

`elementName` of type `DOMString`
: The `name` attribute value for an element.

Return Value

`NodeList`	The matching elements.

No Exceptions

`open`
: Open a document stream for writing. If a document exists in the target, this method clears it.

Note: This method and the ones following allow a user to add to or replace the structure model of a document using strings of unparsed HTML. At the time of writing alternate methods for providing similar functionality for both HTML and XML documents were being considered (see [DOM Level 3 Abstract Schemas and Load and Save]). The following methods may be deprecated at some point in the future in favor of a more general-purpose mechanism.

No Parameters
No Return Value
No Exceptions

`write`
Write a string of text to a document stream opened by `open()`. The text is parsed into the document's structure model.
Parameters
`text` of type `DOMString`
The string to be parsed into some structure in the document structure model.
No Return Value
No Exceptions

`writeln`
Write a string of text followed by a newline character to a document stream opened by `open()`. The text is parsed into the document's structure model.
Parameters
`text` of type `DOMString`
The string to be parsed into some structure in the document structure model.
No Return Value
No Exceptions

1.6. HTML Elements

1.6.1. Property Attributes

HTML attributes are exposed as properties on the element object. The DOM naming conventions always determine the name of the exposed property, and is independent of the case of the attribute in the source document. The data type of the property is in general determined by the type of the attribute as determined by the HTML 4.0 (transitional and frameset) and XHTML 1.0 DTDs. The attributes have the semantics (including case-sensitivity) given in the HTML 4.0 and XHTML 1.0 specifications.

The attributes are exposed as properties for compatibility with *DOM Level 0* [p.119] . This usage is deprecated because it can not be generalized to all possible attribute names, as is required both for XML and potentially for future versions of HTML. We recommend the use of generic methods on the core `Element` interface for setting, getting and removing attributes.

DTD Data Type	***Object Model Data* Type**
CDATA	DOMString
Value list (e.g., (left | right | center))	DOMString
one-value Value list (e.g., (disabled))	boolean
Number	long int

In an HTML document the return value of an attribute that has a data type that is a value list is normalized to lower case (independent of the case of the value in the source document).

For example, if the value of the align attribute on a P element is "Left" (which is not a valid value in XHTML due to the case sensitivity of XHTML) then the value is returned as "left". For attributes with the `CDATA` data type, the case of the return value is that given in the source document.

The return value of an attribute that is unspecified and does not have a default value is the empty string if the return type is a DOMString, false if the return type is a boolean and 0 if the return type is a number.

1.6.2. Naming Exceptions

To avoid namespace conflicts, an attribute with the same name as a keyword in one of our chosen *binding languages* [p.119] is prefixed. For HTML, the prefix used is "html". For example, the `for` attribute of the `LABEL` element collides with loop construct naming conventions and is renamed `htmlFor`.

1.6.3. Exposing Element Type Names (`tagName`, (`nodeName`))

If the document is a HTML 4.0 document the element type names exposed through a property are in uppercase. For example, the body element type name is exposed through the `tagName` property as `BODY`. If the document is an XHTML 1.0 or above document the element name is exposed as it is written in the XHTML file. This means that the element type names are exposed in lower case for XHTML documents since the XHTML 1.0 DTD defines element type names as lower case, and XHTML, being derived from XML, is case sensitive.

1.6.4. The HTMLElement interface

Interface *HTMLElement*

All HTML element interfaces derive from this class. Elements that only expose the HTML core attributes are represented by the base `HTMLElement` interface. These elements are as follows:

- HEAD
- special: SUB, SUP, SPAN, BDO
- font: TT, I, B, U, S, STRIKE, BIG, SMALL
- phrase: EM, STRONG, DFN, CODE, SAMP, KBD, VAR, CITE, ACRONYM, ABBR
- list: DD, DT
- NOFRAMES, NOSCRIPT
- ADDRESS, CENTER

Note: The `style` attribute of an HTML element is accessible through the `ElementCSSInlineStyle` interface which is defined in the CSS module [DOM Level 2 Style Sheets and CSS].

IDL Definition

```
interface HTMLElement : Element {
           attribute DOMString       id;
           attribute DOMString       title;
           attribute DOMString       lang;
           attribute DOMString       dir;
           attribute DOMString       className;
};
```

Attributes

`className` of type `DOMString`
: The class attribute of the element. This attribute has been renamed due to conflicts with the "class" keyword exposed by many languages. See the class attribute definition in HTML 4.0.

`dir` of type `DOMString`
: Specifies the base direction of directionally neutral text and the directionality of tables. See the dir attribute definition in HTML 4.0.

`id` of type `DOMString`
: The element's identifier. See the id attribute definition in HTML 4.0.

`lang` of type `DOMString`
: Language code defined in RFC 1766. See the lang attribute definition in HTML 4.0.

`title` of type `DOMString`
: The element's advisory title. See the title attribute definition in HTML 4.0.

1.6.5. Object definitions

Interface *HTMLHtmlElement*

Root of an HTML document. See the HTML element definition in HTML 4.0.

IDL Definition

```
interface HTMLHtmlElement : HTMLElement {
           attribute DOMString       version;
};
```

Attributes

`version` of type `DOMString`
: Version information about the document's DTD. See the version attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLHeadElement*

Document head information. See the HEAD element definition in HTML 4.0.

IDL Definition

```
interface HTMLHeadElement : HTMLElement {
           attribute DOMString       profile;
};
```

Attributes

`profile` of type `DOMString`

URI [RFC2396] designating a metadata profile. See the profile attribute definition in HTML 4.0.

Interface *HTMLLinkElement*

The `LINK` element specifies a link to an external resource, and defines this document's relationship to that resource (or vice versa). See the LINK element definition in HTML 4.0 (see also the `LinkStyle` interface in the StyleSheet module [DOM Level 2 Style Sheets and CSS]).

IDL Definition

```
interface HTMLLinkElement : HTMLElement {
           attribute boolean         disabled;
           attribute DOMString       charset;
           attribute DOMString       href;
           attribute DOMString       hreflang;
           attribute DOMString       media;
           attribute DOMString       rel;
           attribute DOMString       rev;
           attribute DOMString       target;
           attribute DOMString       type;
};
```

Attributes

`charset` of type `DOMString`

The character encoding of the resource being linked to. See the charset attribute definition in HTML 4.0.

`disabled` of type `boolean`

Enables/disables the link. This is currently only used for style sheet links, and may be used to activate or deactivate style sheets.

`href` of type `DOMString`

The URI [RFC2396] of the linked resource. See the href attribute definition in HTML 4.0.

`hreflang` of type `DOMString`

Language code of the linked resource. See the hreflang attribute definition in HTML 4.0.

`media` of type `DOMString`

Designed for use with one or more target media. See the media attribute definition in HTML 4.0.

`rel` of type `DOMString`

Forward link type. See the rel attribute definition in HTML 4.0.

`rev` of type `DOMString`

Reverse link type. See the rev attribute definition in HTML 4.0.

`target` of type `DOMString`

Frame to render the resource in. See the target attribute definition in HTML 4.0.

`type` of type `DOMString`

Advisory content type. See the type attribute definition in HTML 4.0.

Interface *HTMLTitleElement*

The document title. See the TITLE element definition in HTML 4.0.

IDL Definition

```
interface HTMLTitleElement : HTMLElement {
           attribute DOMString       text;
};
```

Attributes

`text` of type `DOMString`
: The specified title as a string.

Interface *HTMLMetaElement*

This contains generic meta-information about the document. See the META element definition in HTML 4.0.

IDL Definition

```
interface HTMLMetaElement : HTMLElement {
           attribute DOMString       content;
           attribute DOMString       httpEquiv;
           attribute DOMString       name;
           attribute DOMString       scheme;
};
```

Attributes

`content` of type `DOMString`
: Associated information. See the content attribute definition in HTML 4.0.

`httpEquiv` of type `DOMString`
: HTTP response header name [RFC2616]. See the http-equiv attribute definition in HTML 4.0.

`name` of type `DOMString`
: Meta information name. See the name attribute definition in HTML 4.0.

`scheme` of type `DOMString`
: Select form of content. See the scheme attribute definition in HTML 4.0.

Interface *HTMLBaseElement*

Document base URI [RFC2396]. See the BASE element definition in HTML 4.0.

IDL Definition

```
interface HTMLBaseElement : HTMLElement {
           attribute DOMString       href;
           attribute DOMString       target;
};
```

Attributes

`href` of type `DOMString`
: The base URI [RFC2396]. See the href attribute definition in HTML 4.0.

`target` of type `DOMString`
: The default target frame. See the target attribute definition in HTML 4.0.

Interface *HTMLIsIndexElement*

This element is used for single-line text input. See the ISINDEX element definition in HTML 4.0. This element is deprecated in HTML 4.0.

IDL Definition

```
interface HTMLIsIndexElement : HTMLElement {
  readonly attribute HTMLFormElement form;
           attribute DOMString       prompt;
};
```

Attributes

`form` of type `HTMLFormElement` [p.22] , readonly
: Returns the `FORM` element containing this control. Returns `null` if this control is not within the context of a form.

`prompt` of type `DOMString`
: The prompt message. See the prompt attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLStyleElement*

Style information. See the STYLE element definition in HTML 4.0, the CSS module [DOM Level 2 Style Sheets and CSS] and the `LinkStyle` interface in the StyleSheets module [DOM Level 2 Style Sheets and CSS].

IDL Definition

```
interface HTMLStyleElement : HTMLElement {
           attribute boolean         disabled;
           attribute DOMString       media;
           attribute DOMString       type;
};
```

Attributes

`disabled` of type `boolean`
: Enables/disables the style sheet.

`media` of type `DOMString`
: Designed for use with one or more target media. See the media attribute definition in HTML 4.0.

`type` of type `DOMString`
: The content type of the style sheet language. See the type attribute definition in HTML 4.0.

Interface *HTMLBodyElement*

The HTML document body. This element is always present in the DOM API, even if the tags are not present in the source document. See the BODY element definition in HTML 4.0.

IDL Definition

```
interface HTMLBodyElement : HTMLElement {
           attribute DOMString       aLink;
           attribute DOMString       background;
           attribute DOMString       bgColor;
           attribute DOMString       link;
           attribute DOMString       text;
           attribute DOMString       vLink;
};
```

Attributes

`aLink` of type `DOMString`
: Color of active links (after mouse-button down, but before mouse-button up). See the alink attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`background` of type `DOMString`
: URI [RFC2396] of the background texture tile image. See the background attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`bgColor` of type `DOMString`
: Document background color. See the bgcolor attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`link` of type `DOMString`
: Color of links that are not active and unvisited. See the link attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`text` of type `DOMString`
: Document text color. See the text attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`vLink` of type `DOMString`
: Color of links that have been visited by the user. See the vlink attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLFormElement*

The `FORM` element encompasses behavior similar to a collection and an element. It provides direct access to the contained form controls as well as the attributes of the form element. See the FORM element definition in HTML 4.0.

IDL Definition

```
interface HTMLFormElement : HTMLElement {
  readonly attribute HTMLCollection  elements;
  readonly attribute long            length;
           attribute DOMString       name;
           attribute DOMString       acceptCharset;
           attribute DOMString       action;
           attribute DOMString       enctype;
           attribute DOMString       method;
           attribute DOMString       target;
  void               submit();
  void               reset();
};
```

Attributes

`acceptCharset` of type `DOMString`
: List of character sets supported by the server. See the accept-charset attribute definition in HTML 4.0.

`action` of type `DOMString`
: Server-side form handler. See the action attribute definition in HTML 4.0.

`elements` of type `HTMLCollection` [p.12] , readonly
: Returns a collection of all form control elements in the form.

`enctype` of type `DOMString`
: The content type of the submitted form, generally "application/x-www-form-urlencoded". See the enctype attribute definition in HTML 4.0.

`length` of type `long`, readonly
: The number of form controls in the form.

`method` of type `DOMString`
: HTTP method [RFC2616] used to submit form. See the method attribute definition in HTML 4.0.

`name` of type `DOMString`
: Names the form.

`target` of type `DOMString`
: Frame to render the resource in. See the target attribute definition in HTML 4.0.

Methods

`reset`
: Restores a form element's default values. It performs the same action as a reset button.
No Parameters
No Return Value
No Exceptions

`submit`
: Submits the form. It performs the same action as a submit button.
No Parameters
No Return Value
No Exceptions

Interface *HTMLSelectElement*

The select element allows the selection of an option. The contained options can be directly accessed through the select element as a collection. See the SELECT element definition in HTML 4.0.

IDL Definition

```
interface HTMLSelectElement : HTMLElement {
  readonly attribute DOMString       type;
           attribute long            selectedIndex;
           attribute DOMString       value;
  readonly attribute long            length;
  readonly attribute HTMLFormElement form;
  readonly attribute HTMLCollection  options;
           attribute boolean         disabled;
           attribute boolean         multiple;
           attribute DOMString       name;
           attribute long            size;
```

```
           attribute long            tabIndex;
  void               add(in HTMLElement element,
                         in HTMLElement before)
                                        raises(DOMException);
  void               remove(in long index);
  void               blur();
  void               focus();
};
```

Attributes

`disabled` of type `boolean`
: The control is unavailable in this context. See the disabled attribute definition in HTML 4.0.

`form` of type `HTMLFormElement` [p.22] , readonly
: Returns the `FORM` element containing this control. Returns `null` if this control is not within the context of a form.

`length` of type `long`, readonly
: The number of options in this `SELECT`.

`multiple` of type `boolean`
: If true, multiple `OPTION` elements may be selected in this `SELECT`. See the multiple attribute definition in HTML 4.0.

`name` of type `DOMString`
: Form control or object name when submitted with a form. See the name attribute definition in HTML 4.0.

`options` of type `HTMLCollection` [p.12] , readonly
: The collection of `OPTION` elements contained by this element.

`selectedIndex` of type `long`
: The ordinal index of the selected option, starting from 0. The value -1 is returned if no element is selected. If multiple options are selected, the index of the first selected option is returned.

`size` of type `long`
: Number of visible rows. See the size attribute definition in HTML 4.0.

`tabIndex` of type `long`
: Index that represents the element's position in the tabbing order. See the tabindex attribute definition in HTML 4.0.

`type` of type `DOMString`, readonly
: The type of this form control. This is the string "select-multiple" when the multiple attribute is `true` and the string "select-one" when `false`.

`value` of type `DOMString`
: The current form control value (i.e. the value of the currently selected option), if multiple options are selected this is the value of the first selected option.

Methods

`add`
: Add a new element to the collection of `OPTION` elements for this `SELECT`. This method is the equivalent of the `appendChild` method of the `Node` interface if the `before` parameter is `null`. It is equivalent to the `insertBefore` method on the parent of `before` in all other cases.

  Parameters

`element` of type `HTMLElement` [p.17]
The element to add.
`before` of type `HTMLElement`
The element to insert before, or `null` for the tail of the list.
Exceptions

`DOMException`	NOT_FOUND_ERR: Raised if `before` is not a descendant of the `SELECT` element.

No Return Value

`blur`
Removes keyboard focus from this element.
No Parameters
No Return Value
No Exceptions

`focus`
Gives keyboard focus to this element.
No Parameters
No Return Value
No Exceptions

`remove`
Remove an element from the collection of `OPTION` elements for this `SELECT`. Does nothing if no element has the given index.
Parameters
`index` of type `long`
The index of the item to remove, starting from 0.
No Return Value
No Exceptions

Interface *HTMLOptGroupElement*

Group options together in logical subdivisions. See the OPTGROUP element definition in HTML 4.0.

IDL Definition

```
interface HTMLOptGroupElement : HTMLElement {
           attribute boolean         disabled;
           attribute DOMString       label;
};
```

Attributes

`disabled` of type `boolean`
The control is unavailable in this context. See the disabled attribute definition in HTML 4.0.
`label` of type `DOMString`
Assigns a label to this option group. See the label attribute definition in HTML 4.0.

Interface *HTMLOptionElement*

A selectable choice. See the OPTION element definition in HTML 4.0.

IDL Definition

```
interface HTMLOptionElement : HTMLElement {
  readonly attribute HTMLFormElement form;
           attribute boolean         defaultSelected;
  readonly attribute DOMString       text;
  readonly attribute long            index;
           attribute boolean         disabled;
           attribute DOMString       label;
           attribute boolean         selected;
           attribute DOMString       value;
};
```

Attributes

`defaultSelected` of type `boolean`
: Represents the value of the HTML selected attribute. The value of this attribute does not change if the state of the corresponding form control, in an interactive user agent, changes. See the selected attribute definition in HTML 4.0.

`disabled` of type `boolean`
: The control is unavailable in this context. See the disabled attribute definition in HTML 4.0.

`form` of type `HTMLFormElement` [p.22] , readonly
: Returns the `FORM` element containing this control. Returns `null` if this control is not within the context of a form.

`index` of type `long`, readonly
: The index of this `OPTION` in its parent `SELECT`, starting from 0.

`label` of type `DOMString`
: Option label for use in hierarchical menus. See the label attribute definition in HTML 4.0.

`selected` of type `boolean`
: Represents the current state of the corresponding form control, in an interactive user agent. Changing this attribute changes the state of the form control, but does not change the value of the HTML selected attribute of the element.

`text` of type `DOMString`, readonly
: The text contained within the option element.

`value` of type `DOMString`
: The current form control value. See the value attribute definition in HTML 4.0.

Interface *HTMLInputElement*

Form control.

Note: Depending upon the environment in which the page is being viewed, the value property may be read-only for the file upload input type. For the "password" input type, the actual value returned may be masked to prevent unauthorized use. See the INPUT element definition in [HTML4.0].

IDL Definition

```
interface HTMLInputElement : HTMLElement {
           attribute DOMString       defaultValue;
           attribute boolean         defaultChecked;
  readonly attribute HTMLFormElement form;
           attribute DOMString       accept;
           attribute DOMString       accessKey;
           attribute DOMString       align;
           attribute DOMString       alt;
           attribute boolean         checked;
           attribute boolean         disabled;
           attribute long            maxLength;
           attribute DOMString       name;
           attribute boolean         readOnly;
           attribute DOMString       size;
           attribute DOMString       src;
           attribute long            tabIndex;
  // Modified in DOM Level 2:
           attribute DOMString       type;
           attribute DOMString       useMap;
           attribute DOMString       value;
  void               blur();
  void               focus();
  void               select();
  void               click();
};
```

Attributes

`accept` of type `DOMString`
: A comma-separated list of content types that a server processing this form will handle correctly. See the accept attribute definition in HTML 4.0.

`accessKey` of type `DOMString`
: A single character access key to give access to the form control. See the accesskey attribute definition in HTML 4.0.

`align` of type `DOMString`
: Aligns this object (vertically or horizontally) with respect to its surrounding text. See the align attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`alt` of type `DOMString`
: Alternate text for user agents not rendering the normal content of this element. See the alt attribute definition in HTML 4.0.

`checked` of type `boolean`
: When the `type` attribute of the element has the value "radio" or "checkbox", this represents the current state of the form control, in an interactive user agent. Changes to this attribute change the state of the form control, but do not change the value of the HTML value attribute of the element.

`defaultChecked` of type `boolean`
: When `type` has the value "radio" or "checkbox", this represents the HTML checked attribute of the element. The value of this attribute does not change if the state of the corresponding form control, in an interactive user agent, changes. See the checked attribute definition in HTML 4.0.

`defaultValue` of type `DOMString`
: When the `type` attribute of the element has the value "text", "file" or "password", this represents the HTML value attribute of the element. The value of this attribute does not change if the contents of the corresponding form control, in an interactive user agent, changes. See the value attribute definition in HTML 4.0.

`disabled` of type `boolean`
: The control is unavailable in this context. See the disabled attribute definition in HTML 4.0.

`form` of type `HTMLFormElement` [p.22] , readonly
: Returns the `FORM` element containing this control. Returns `null` if this control is not within the context of a form.

`maxLength` of type `long`
: Maximum number of characters for text fields, when `type` has the value "text" or "password". See the maxlength attribute definition in HTML 4.0.

`name` of type `DOMString`
: Form control or object name when submitted with a form. See the name attribute definition in HTML 4.0.

`readOnly` of type `boolean`
: This control is read-only. Relevant only when `type` has the value "text" or "password". See the readonly attribute definition in HTML 4.0.

`size` of type `DOMString`
: Size information. The precise meaning is specific to each type of field. See the size attribute definition in HTML 4.0.

`src` of type `DOMString`
: When the `type` attribute has the value "image", this attribute specifies the location of the image to be used to decorate the graphical submit button. See the src attribute definition in HTML 4.0.

`tabIndex` of type `long`
: Index that represents the element's position in the tabbing order. See the tabindex attribute definition in HTML 4.0.

`type` of type `DOMString`, modified in **DOM Level 2**
: The type of control created (all lower case). See the type attribute definition in HTML 4.0.

`useMap` of type `DOMString`
: Use client-side image map. See the usemap attribute definition in HTML 4.0.

`value` of type `DOMString`
: When the `type` attribute of the element has the value "text", "file" or "password", this represents the current contents of the corresponding form control, in an interactive user agent. Changing this attribute changes the contents of the form control, but does not change the value of the HTML value attribute of the element. When the `type` attribute of the element has the value "button", "hidden", "submit", "reset", "image", "checkbox" or "radio", this represents the HTML value attribute of the element. See the value attribute definition in HTML 4.0.

Methods

`blur`
: Removes keyboard focus from this element.
No Parameters

No Return Value

No Exceptions

`click`

Simulate a mouse-click. For `INPUT` elements whose `type` attribute has one of the following values: "button", "checkbox", "radio", "reset", or "submit".

No Parameters

No Return Value

No Exceptions

`focus`

Gives keyboard focus to this element.

No Parameters

No Return Value

No Exceptions

`select`

Select the contents of the text area. For `INPUT` elements whose `type` attribute has one of the following values: "text", "file", or "password".

No Parameters

No Return Value

No Exceptions

Interface *HTMLTextAreaElement*

Multi-line text field. See the TEXTAREA element definition in HTML 4.0.

IDL Definition

```
interface HTMLTextAreaElement : HTMLElement {
           attribute DOMString       defaultValue;
  readonly attribute HTMLFormElement form;
           attribute DOMString       accessKey;
           attribute long            cols;
           attribute boolean         disabled;
           attribute DOMString       name;
           attribute boolean         readOnly;
           attribute long            rows;
           attribute long            tabIndex;
  readonly attribute DOMString       type;
           attribute DOMString       value;
  void               blur();
  void               focus();
  void               select();
};
```

Attributes

`accessKey` of type `DOMString`

A single character access key to give access to the form control. See the accesskey attribute definition in HTML 4.0.

`cols` of type `long`

Width of control (in characters). See the cols attribute definition in HTML 4.0.

`defaultValue` of type `DOMString`
: Represents the contents of the element. The value of this attribute does not change if the contents of the corresponding form control, in an interactive user agent, changes.

`disabled` of type `boolean`
: The control is unavailable in this context. See the disabled attribute definition in HTML 4.0.

`form` of type `HTMLFormElement` [p.22] , readonly
: Returns the `FORM` element containing this control. Returns `null` if this control is not within the context of a form.

`name` of type `DOMString`
: Form control or object name when submitted with a form. See the name attribute definition in HTML 4.0.

`readOnly` of type `boolean`
: This control is read-only. See the readonly attribute definition in HTML 4.0.

`rows` of type `long`
: Number of text rows. See the rows attribute definition in HTML 4.0.

`tabIndex` of type `long`
: Index that represents the element's position in the tabbing order. See the tabindex attribute definition in HTML 4.0.

`type` of type `DOMString`, readonly
: The type of this form control. This the string "textarea".

`value` of type `DOMString`
: Represents the current contents of the corresponding form control, in an interactive user agent. Changing this attribute changes the contents of the form control, but does not change the contents of the element. If the entirety of the data can not fit into a single `DOMString`, the implementation may truncate the data.

Methods

`blur`
: Removes keyboard focus from this element.
No Parameters
No Return Value
No Exceptions

`focus`
: Gives keyboard focus to this element.
No Parameters
No Return Value
No Exceptions

`select`
: Select the contents of the `TEXTAREA`.
No Parameters
No Return Value
No Exceptions

Interface *HTMLButtonElement*

Push button. See the BUTTON element definition in HTML 4.0.

IDL Definition

```
interface HTMLButtonElement : HTMLElement {
  readonly attribute HTMLFormElement form;
           attribute DOMString       accessKey;
           attribute boolean         disabled;
           attribute DOMString       name;
           attribute long            tabIndex;
  readonly attribute DOMString       type;
           attribute DOMString       value;
};
```

Attributes

`accessKey` of type `DOMString`
: A single character access key to give access to the form control. See the accesskey attribute definition in HTML 4.0.

`disabled` of type `boolean`
: The control is unavailable in this context. See the disabled attribute definition in HTML 4.0.

`form` of type `HTMLFormElement` [p.22] , readonly
: Returns the `FORM` element containing this control. Returns `null` if this control is not within the context of a form.

`name` of type `DOMString`
: Form control or object name when submitted with a form. See the name attribute definition in HTML 4.0.

`tabIndex` of type `long`
: Index that represents the element's position in the tabbing order. See the tabindex attribute definition in HTML 4.0.

`type` of type `DOMString`, readonly
: The type of button (all lower case). See the type attribute definition in HTML 4.0.

`value` of type `DOMString`
: The current form control value. See the value attribute definition in HTML 4.0.

Interface *HTMLLabelElement*

Form field label text. See the LABEL element definition in HTML 4.0.

IDL Definition

```
interface HTMLLabelElement : HTMLElement {
  readonly attribute HTMLFormElement form;
           attribute DOMString       accessKey;
           attribute DOMString       htmlFor;
};
```

Attributes

`accessKey` of type `DOMString`
: A single character access key to give access to the form control. See the accesskey attribute definition in HTML 4.0.

`form` of type `HTMLFormElement` [p.22] , readonly
: Returns the `FORM` element containing this control. Returns `null` if this control is not within the context of a form.

`htmlFor` of type `DOMString`
: This attribute links this label with another form control by `id` attribute. See the for attribute definition in HTML 4.0.

Interface *HTMLFieldSetElement*

Organizes form controls into logical groups. See the FIELDSET element definition in HTML 4.0.

IDL Definition

```
interface HTMLFieldSetElement : HTMLElement {
  readonly attribute HTMLFormElement form;
};
```

Attributes

`form` of type `HTMLFormElement` [p.22] , readonly
: Returns the `FORM` element containing this control. Returns `null` if this control is not within the context of a form.

Interface *HTMLLegendElement*

Provides a caption for a `FIELDSET` grouping. See the LEGEND element definition in HTML 4.0.

IDL Definition

```
interface HTMLLegendElement : HTMLElement {
  readonly attribute HTMLFormElement form;
           attribute DOMString       accessKey;
           attribute DOMString       align;
};
```

Attributes

`accessKey` of type `DOMString`
: A single character access key to give access to the form control. See the accesskey attribute definition in HTML 4.0.

`align` of type `DOMString`
: Text alignment relative to `FIELDSET`. See the align attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`form` of type `HTMLFormElement` [p.22] , readonly
: Returns the `FORM` element containing this control. Returns `null` if this control is not within the context of a form.

Interface *HTMLUListElement*

Unordered list. See the UL element definition in HTML 4.0.

IDL Definition

```
interface HTMLUListElement : HTMLElement {
           attribute boolean         compact;
           attribute DOMString       type;
};
```

Attributes

`compact` of type `boolean`

Reduce spacing between list items. See the compact attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`type` of type `DOMString`

Bullet style. See the type attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLOListElement*

Ordered list. See the OL element definition in HTML 4.0.

IDL Definition

```
interface HTMLOListElement : HTMLElement {
           attribute boolean         compact;
           attribute long            start;
           attribute DOMString       type;
};
```

Attributes

`compact` of type `boolean`

Reduce spacing between list items. See the compact attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`start` of type `long`

Starting sequence number. See the start attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`type` of type `DOMString`

Numbering style. See the type attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLDListElement*

Definition list. See the DL element definition in HTML 4.0.

IDL Definition

```
interface HTMLDListElement : HTMLElement {
           attribute boolean         compact;
};
```

Attributes

`compact` of type `boolean`

Reduce spacing between list items. See the compact attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLDirectoryElement*

Directory list. See the DIR element definition in HTML 4.0. This element is deprecated in HTML 4.0.

IDL Definition

```
interface HTMLDirectoryElement : HTMLElement {
           attribute boolean         compact;
};
```

Attributes

`compact` of type `boolean`

Reduce spacing between list items. See the compact attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLMenuElement*

Menu list. See the MENU element definition in HTML 4.0. This element is deprecated in HTML 4.0.

IDL Definition

```
interface HTMLMenuElement : HTMLElement {
           attribute boolean         compact;
};
```

Attributes

`compact` of type `boolean`

Reduce spacing between list items. See the compact attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLLIElement*

List item. See the LI element definition in HTML 4.0.

IDL Definition

```
interface HTMLLIElement : HTMLElement {
           attribute DOMString       type;
           attribute long            value;
};
```

Attributes

`type` of type `DOMString`

List item bullet style. See the type attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`value` of type `long`

Reset sequence number when used in `OL`. See the value attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLDivElement*

Generic block container. See the DIV element definition in HTML 4.0.

IDL Definition

```
interface HTMLDivElement : HTMLElement {
           attribute DOMString       align;
};
```

Attributes

`align` of type `DOMString`

Horizontal text alignment. See the align attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLParagraphElement*

Paragraphs. See the P element definition in HTML 4.0.

IDL Definition

```
interface HTMLParagraphElement : HTMLElement {
           attribute DOMString       align;
};
```

Attributes

`align` of type `DOMString`

Horizontal text alignment. See the align attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLHeadingElement*

For the `H1` to `H6` elements. See the H1 element definition in HTML 4.0.

IDL Definition

```
interface HTMLHeadingElement : HTMLElement {
           attribute DOMString       align;
};
```

Attributes

`align` of type `DOMString`

Horizontal text alignment. See the align attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLQuoteElement*

For the `Q` and `BLOCKQUOTE` elements. See the Q element definition in HTML 4.0.

IDL Definition

```
interface HTMLQuoteElement : HTMLElement {
           attribute DOMString       cite;
};
```

Attributes

`cite` of type `DOMString`

A URI [RFC2396] designating a source document or message. See the cite attribute definition in HTML 4.0.

Interface *HTMLPreElement*

Preformatted text. See the PRE element definition in HTML 4.0.

IDL Definition

```
interface HTMLPreElement : HTMLElement {
           attribute long            width;
};
```

Attributes

`width` of type `long`

Fixed width for content. See the width attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLBRElement*

Force a line break. See the BR element definition in HTML 4.0.

IDL Definition

```
interface HTMLBRElement : HTMLElement {
           attribute DOMString       clear;
};
```

Attributes

`clear` of type `DOMString`

Control flow of text around floats. See the clear attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLBaseFontElement*

Base font. See the BASEFONT element definition in HTML 4.0. This element is deprecated in HTML 4.0.

IDL Definition

```
interface HTMLBaseFontElement : HTMLElement {
           attribute DOMString       color;
           attribute DOMString       face;
           attribute DOMString       size;
};
```

Attributes

`color` of type `DOMString`

Font color. See the color attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`face` of type `DOMString`

Font face identifier. See the face attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`size` of type `DOMString`

Font size. See the size attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLFontElement*

Local change to font. See the FONT element definition in HTML 4.0. This element is deprecated in HTML 4.0.

IDL Definition

```
interface HTMLFontElement : HTMLElement {
           attribute DOMString       color;
           attribute DOMString       face;
           attribute DOMString       size;
};
```

Attributes

`color` of type `DOMString`
: Font color. See the color attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`face` of type `DOMString`
: Font face identifier. See the face attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`size` of type `DOMString`
: Font size. See the size attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLHRElement*

Create a horizontal rule. See the HR element definition in HTML 4.0.

IDL Definition

```
interface HTMLHRElement : HTMLElement {
           attribute DOMString       align;
           attribute boolean         noShade;
           attribute DOMString       size;
           attribute DOMString       width;
};
```

Attributes

`align` of type `DOMString`
: Align the rule on the page. See the align attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`noShade` of type `boolean`
: Indicates to the user agent that there should be no shading in the rendering of this element. See the noshade attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`size` of type `DOMString`
: The height of the rule. See the size attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`width` of type `DOMString`
: The width of the rule. See the width attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLModElement*

Notice of modification to part of a document. See the INS and DEL element definitions in HTML 4.0.

IDL Definition

```
interface HTMLModElement : HTMLElement {
           attribute DOMString       cite;
           attribute DOMString       dateTime;
};
```

Attributes

`cite` of type `DOMString`
: A URI [RFC2396] designating a document that describes the reason for the change. See the cite attribute definition in HTML 4.0.

`dateTime` of type `DOMString`
: The date and time of the change. See the datetime attribute definition in HTML 4.0.

Interface *HTMLAnchorElement*

The anchor element. See the A element definition in HTML 4.0.

IDL Definition

```
interface HTMLAnchorElement : HTMLElement {
           attribute DOMString       accessKey;
           attribute DOMString       charset;
           attribute DOMString       coords;
           attribute DOMString       href;
           attribute DOMString       hreflang;
           attribute DOMString       name;
           attribute DOMString       rel;
           attribute DOMString       rev;
           attribute DOMString       shape;
           attribute long            tabIndex;
           attribute DOMString       target;
           attribute DOMString       type;
  void               blur();
  void               focus();
};
```

Attributes

`accessKey` of type `DOMString`
: A single character access key to give access to the form control. See the accesskey attribute definition in HTML 4.0.

`charset` of type `DOMString`
: The character encoding of the linked resource. See the charset attribute definition in HTML 4.0.

`coords` of type `DOMString`
: Comma-separated list of lengths, defining an active region geometry. See also `shape` for the shape of the region. See the coords attribute definition in HTML 4.0.

`href` of type `DOMString`
: The absolute URI [RFC2396] of the linked resource. See the href attribute definition in HTML 4.0.

`hreflang` of type `DOMString`
: Language code of the linked resource. See the hreflang attribute definition in HTML 4.0.

`name` of type `DOMString`
: Anchor name. See the name attribute definition in HTML 4.0.

`rel` of type `DOMString`
: Forward link type. See the rel attribute definition in HTML 4.0.

`rev` of type `DOMString`
: Reverse link type. See the rev attribute definition in HTML 4.0.

`shape` of type `DOMString`
: The shape of the active area. The coordinates are given by `coords`. See the shape attribute definition in HTML 4.0.

`tabIndex` of type `long`
: Index that represents the element's position in the tabbing order. See the tabindex attribute definition in HTML 4.0.

`target` of type `DOMString`
: Frame to render the resource in. See the target attribute definition in HTML 4.0.

`type` of type `DOMString`
: Advisory content type. See the type attribute definition in HTML 4.0.

Methods

`blur`
: Removes keyboard focus from this element.
No Parameters
No Return Value
No Exceptions

`focus`
: Gives keyboard focus to this element.
No Parameters
No Return Value
No Exceptions

Interface *HTMLImageElement*

Embedded image. See the IMG element definition in HTML 4.0.

IDL Definition

```
interface HTMLImageElement : HTMLElement {
           attribute DOMString       name;
           attribute DOMString       align;
           attribute DOMString       alt;
           attribute long            border;
           attribute long            height;
           attribute long            hspace;
           attribute boolean         isMap;
           attribute DOMString       longDesc;
           attribute DOMString       src;
```

```
            attribute DOMString       useMap;
            attribute long            vspace;
            attribute long            width;
};
```

Attributes

`align` of type `DOMString`
: Aligns this object (vertically or horizontally) with respect to its surrounding text. See the align attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`alt` of type `DOMString`
: Alternate text for user agents not rendering the normal content of this element. See the alt attribute definition in HTML 4.0.

`border` of type `long`
: Width of border around image in pixels. See the border attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0. Note that the type of this attribute was `DOMString` in DOM Level 1 HTML [DOM Level 1].

`height` of type `long`
: Height of the image in pixels. See the height attribute definition in HTML 4.0. Note that the type of this attribute was `DOMString` in DOM Level 1 HTML [DOM Level 1].

`hspace` of type `long`
: Horizontal space to the left and right of this image in pixels. See the hspace attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0. Note that the type of this attribute was `DOMString` in DOM Level 1 HTML [DOM Level 1].

`isMap` of type `boolean`
: Use server-side image map. See the ismap attribute definition in HTML 4.0.

`longDesc` of type `DOMString`
: URI [RFC2396] designating a long description of this image or frame. See the longdesc attribute definition in HTML 4.0.

`name` of type `DOMString`
: The name of the element (for backwards compatibility).

`src` of type `DOMString`
: URI [RFC2396] designating the source of this image. See the src attribute definition in HTML 4.0.

`useMap` of type `DOMString`
: Use client-side image map. See the usemap attribute definition in HTML 4.0.

`vspace` of type `long`
: Vertical space above and below this image in pixels. See the vspace attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0. Note that the type of this attribute was "DOMString" in DOM Level 1 HTML [DOM Level 1].

`width` of type `long`
: The width of the image in pixels. See the width attribute definition in HTML 4.0. Note that the type of this attribute was `DOMString` in DOM Level 1 HTML [DOM Level 1].

Interface *HTMLObjectElement*

Generic embedded object.

Note: In principle, all properties on the object element are read-write but in some environments some properties may be read-only once the underlying object is instantiated. See the OBJECT element definition in [HTML4.0].

IDL Definition

```
interface HTMLObjectElement : HTMLElement {
           attribute DOMString       code;
           attribute DOMString       align;
           attribute DOMString       archive;
           attribute DOMString       border;
           attribute DOMString       codeBase;
           attribute DOMString       codeType;
           attribute DOMString       data;
           attribute boolean         declare;
           attribute DOMString       height;
           attribute DOMString       hspace;
           attribute DOMString       name;
           attribute DOMString       standby;
           attribute long            tabIndex;
           attribute DOMString       type;
           attribute DOMString       useMap;
           attribute DOMString       vspace;
           attribute DOMString       width;
  // Introduced in DOM Level 2:
  readonly attribute Document        contentDocument;
};
```

Attributes

`align` of type `DOMString`
: Aligns this object (vertically or horizontally) with respect to its surrounding text. See the align attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`archive` of type `DOMString`
: Space-separated list of archives. See the archive attribute definition in HTML 4.0.

`border` of type `DOMString`
: Width of border around the object. See the border attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`code` of type `DOMString`
: Applet class file. See the `code` attribute for HTMLAppletElement.

`codeBase` of type `DOMString`
: Base URI [RFC2396] for `classid`, `data`, and `archive` attributes. See the codebase attribute definition in HTML 4.0.

`codeType` of type `DOMString`
: Content type for data downloaded via `classid` attribute. See the codetype attribute definition in HTML 4.0.

`contentDocument` of type `Document`, readonly, introduced in **DOM Level 2**
: The document this object contains, if there is any and it is available, or `null` otherwise.

`data` of type `DOMString`
: A URI [RFC2396] specifying the location of the object's data. See the data attribute definition in HTML 4.0.

`declare` of type `boolean`
: Declare (for future reference), but do not instantiate, this object. See the declare attribute definition in HTML 4.0.

`height` of type `DOMString`
: Override height. See the height attribute definition in HTML 4.0.

`hspace` of type `DOMString`
: Horizontal space to the left and right of this image, applet, or object. See the hspace attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`name` of type `DOMString`
: Form control or object name when submitted with a form. See the name attribute definition in HTML 4.0.

`standby` of type `DOMString`
: Message to render while loading the object. See the standby attribute definition in HTML 4.0.

`tabIndex` of type `long`
: Index that represents the element's position in the tabbing order. See the tabindex attribute definition in HTML 4.0.

`type` of type `DOMString`
: Content type for data downloaded via `data` attribute. See the type attribute definition in HTML 4.0.

`useMap` of type `DOMString`
: Use client-side image map. See the usemap attribute definition in HTML 4.0.

`vspace` of type `DOMString`
: Vertical space above and below this image, applet, or object. See the vspace attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`width` of type `DOMString`
: Override width. See the width attribute definition in HTML 4.0.

Interface *HTMLParamElement*

Parameters fed to the `OBJECT` element. See the PARAM element definition in HTML 4.0.

IDL Definition

```
interface HTMLParamElement : HTMLElement {
           attribute DOMString       name;
           attribute DOMString       type;
           attribute DOMString       value;
           attribute DOMString       valueType;
};
```

Attributes

`name` of type `DOMString`
: The name of a run-time parameter. See the name attribute definition in HTML 4.0.

`type` of type `DOMString`
: Content type for the `value` attribute when `valuetype` has the value "ref". See the type attribute definition in HTML 4.0.

`value` of type `DOMString`
: The value of a run-time parameter. See the value attribute definition in HTML 4.0.

`valueType` of type `DOMString`
: Information about the meaning of the `value` attribute value. See the valuetype attribute definition in HTML 4.0.

Interface *HTMLAppletElement*

An embedded Java applet. See the APPLET element definition in HTML 4.0. This element is deprecated in HTML 4.0.

IDL Definition

```
interface HTMLAppletElement : HTMLElement {
           attribute DOMString       align;
           attribute DOMString       alt;
           attribute DOMString       archive;
           attribute DOMString       code;
           attribute DOMString       codeBase;
           attribute DOMString       height;
           attribute DOMString       hspace;
           attribute DOMString       name;
           attribute DOMString       object;
           attribute DOMString       vspace;
           attribute DOMString       width;
};
```

Attributes

`align` of type `DOMString`
: Aligns this object (vertically or horizontally) with respect to its surrounding text. See the align attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`alt` of type `DOMString`
: Alternate text for user agents not rendering the normal content of this element. See the alt attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`archive` of type `DOMString`
: Comma-separated archive list. See the archive attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`code` of type `DOMString`
: Applet class file. See the code attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`codeBase` of type `DOMString`
: Optional base URI [RFC2396] for applet. See the codebase attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`height` of type `DOMString`
: Override height. See the height attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`hspace` of type `DOMString`
: Horizontal space to the left and right of this image, applet, or object. See the hspace attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`name` of type `DOMString`
: The name of the applet. See the name attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`object` of type `DOMString`
: Serialized applet file. See the object attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`vspace` of type `DOMString`
: Vertical space above and below this image, applet, or object. See the vspace attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`width` of type `DOMString`
: Override width. See the width attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLMapElement*

Client-side image map. See the MAP element definition in HTML 4.0.

IDL Definition

```
interface HTMLMapElement : HTMLElement {
  readonly attribute HTMLCollection  areas;
           attribute DOMString       name;
};
```

Attributes

`areas` of type `HTMLCollection` [p.12] , readonly
: The list of areas defined for the image map.

`name` of type `DOMString`
: Names the map (for use with `usemap`). See the name attribute definition in HTML 4.0.

Interface *HTMLAreaElement*

Client-side image map area definition. See the AREA element definition in HTML 4.0.

IDL Definition

```
interface HTMLAreaElement : HTMLElement {
           attribute DOMString       accessKey;
           attribute DOMString       alt;
           attribute DOMString       coords;
           attribute DOMString       href;
           attribute boolean         noHref;
           attribute DOMString       shape;
           attribute long            tabIndex;
           attribute DOMString       target;
};
```

Attributes

`accessKey` of type `DOMString`
: A single character access key to give access to the form control. See the accesskey attribute definition in HTML 4.0.

`alt` of type `DOMString`
: Alternate text for user agents not rendering the normal content of this element. See the alt attribute definition in HTML 4.0.

`coords` of type `DOMString`
: Comma-separated list of lengths, defining an active region geometry. See also `shape` for the shape of the region. See the coords attribute definition in HTML 4.0.

`href` of type `DOMString`
: The URI [RFC2396] of the linked resource. See the href attribute definition in HTML 4.0.

`noHref` of type `boolean`
: Specifies that this area is inactive, i.e., has no associated action. See the nohref attribute definition in HTML 4.0.

`shape` of type `DOMString`
: The shape of the active area. The coordinates are given by `coords`. See the shape attribute definition in HTML 4.0.

`tabIndex` of type `long`
: Index that represents the element's position in the tabbing order. See the tabindex attribute definition in HTML 4.0.

`target` of type `DOMString`
: Frame to render the resource in. See the target attribute definition in HTML 4.0.

Interface *HTMLScriptElement*

Script statements. See the SCRIPT element definition in HTML 4.0.

IDL Definition

```
interface HTMLScriptElement : HTMLElement {
           attribute DOMString       text;
           attribute DOMString       htmlFor;
           attribute DOMString       event;
           attribute DOMString       charset;
           attribute boolean         defer;
           attribute DOMString       src;
           attribute DOMString       type;
};
```

Attributes

`charset` of type `DOMString`
: The character encoding of the linked resource. See the charset attribute definition in HTML 4.0.

`defer` of type `boolean`
: Indicates that the user agent can defer processing of the script. See the defer attribute definition in HTML 4.0.

`event` of type `DOMString`
: *Reserved for future use.*

`htmlFor` of type `DOMString`
: *Reserved for future use.*

`src` of type `DOMString`
: URI [RFC2396] designating an external script. See the src attribute definition in HTML 4.0.

`text` of type `DOMString`
: The script content of the element.

`type` of type `DOMString`
: The content type of the script language. See the type attribute definition in HTML 4.0.

Interface *HTMLTableElement*

The create* and delete* methods on the table allow authors to construct and modify tables. HTML 4.0 specifies that only one of each of the `CAPTION`, `THEAD`, and `TFOOT` elements may exist in a table. Therefore, if one exists, and the createTHead() or createTFoot() method is called, the method returns the existing THead or TFoot element. See the TABLE element definition in HTML 4.0.

IDL Definition

```
interface HTMLTableElement : HTMLElement {
           attribute HTMLTableCaptionElement caption;
           attribute HTMLTableSectionElement tHead;
           attribute HTMLTableSectionElement tFoot;
  readonly attribute HTMLCollection  rows;
  readonly attribute HTMLCollection  tBodies;
           attribute DOMString       align;
           attribute DOMString       bgColor;
           attribute DOMString       border;
           attribute DOMString       cellPadding;
           attribute DOMString       cellSpacing;
           attribute DOMString       frame;
           attribute DOMString       rules;
           attribute DOMString       summary;
           attribute DOMString       width;
  HTMLElement        createTHead();
  void               deleteTHead();
  HTMLElement        createTFoot();
  void               deleteTFoot();
  HTMLElement        createCaption();
  void               deleteCaption();
  HTMLElement        insertRow(in long index)
                                    raises(DOMException);
  void               deleteRow(in long index)
                                    raises(DOMException);
};
```

Attributes

`align` of type `DOMString`
: Specifies the table's position with respect to the rest of the document. See the align attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`bgColor` of type `DOMString`
: Cell background color. See the bgcolor attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`border` of type `DOMString`
: The width of the border around the table. See the border attribute definition in HTML 4.0.

`caption` of type `HTMLTableCaptionElement` [p.49]
: Returns the table's `CAPTION`, or void if none exists.

`cellPadding` of type `DOMString`
: Specifies the horizontal and vertical space between cell content and cell borders. See the cellpadding attribute definition in HTML 4.0.

`cellSpacing` of type `DOMString`
: Specifies the horizontal and vertical separation between cells. See the cellspacing attribute definition in HTML 4.0.

`frame` of type `DOMString`
: Specifies which external table borders to render. See the frame attribute definition in HTML 4.0.

`rows` of type `HTMLCollection` [p.12] , readonly
: Returns a collection of all the rows in the table, including all in `THEAD`, `TFOOT`, all `TBODY` elements.

`rules` of type `DOMString`
: Specifies which internal table borders to render. See the rules attribute definition in HTML 4.0.

`summary` of type `DOMString`
: Description about the purpose or structure of a table. See the summary attribute definition in HTML 4.0.

`tBodies` of type `HTMLCollection` [p.12] , readonly
: Returns a collection of the defined table bodies.

`tFoot` of type `HTMLTableSectionElement` [p.50]
: Returns the table's `TFOOT`, or `null` if none exists.

`tHead` of type `HTMLTableSectionElement` [p.50]
: Returns the table's `THEAD`, or `null` if none exists.

`width` of type `DOMString`
: Specifies the desired table width. See the width attribute definition in HTML 4.0.

Methods

`createCaption`
: Create a new table caption object or return an existing one.

Return Value

`HTMLElement` [p.17]	A `CAPTION` element.

No Parameters

No Exceptions

`createTFoot`
: Create a table footer row or return an existing one.

Return Value

`HTMLElement` [p.17]	A footer element (`TFOOT`).

No Parameters

No Exceptions

`createTHead`
Create a table header row or return an existing one.
Return Value

`HTMLElement` [p.17]	A new table header element (`THEAD`).

No Parameters
No Exceptions
`deleteCaption`
Delete the table caption, if one exists.
No Parameters
No Return Value
No Exceptions
`deleteRow`
Delete a table row.
Parameters
`index` of type `long`
The index of the row to be deleted. This index starts from 0 and is relative to all the rows contained inside the table, regardless of section parentage.
Exceptions

`DOMException`	INDEX_SIZE_ERR: Raised if the specified index is greater than or equal to the number of rows or if the index is negative.

No Return Value
`deleteTFoot`
Delete the footer from the table, if one exists.
No Parameters
No Return Value
No Exceptions
`deleteTHead`
Delete the header from the table, if one exists.
No Parameters
No Return Value
No Exceptions
`insertRow`
Insert a new empty row in the table. The new row is inserted immediately before and in the same section as the current `index`th row in the table. If `index` is equal to the number of rows, the new row is appended. In addition, when the table is empty the row is inserted into a `TBODY` which is created and inserted into the table.

Note: A table row cannot be empty according to [HTML4.0].

Parameters

`index` of type `long`
The row number where to insert a new row. This index starts from 0 and is relative to all the rows contained inside the table, regardless of section parentage.

Return Value

`HTMLElement` [p.17]	The newly created row.

Exceptions

`DOMException`	INDEX_SIZE_ERR: Raised if the specified index is greater than the number of rows or if the index is negative.

Interface *HTMLTableCaptionElement*

Table caption See the CAPTION element definition in HTML 4.0.

IDL Definition

```
interface HTMLTableCaptionElement : HTMLElement {
           attribute DOMString       align;
};
```

Attributes

`align` of type `DOMString`
Caption alignment with respect to the table. See the align attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLTableColElement*

Regroups the `COL` and `COLGROUP` elements. See the COL element definition in HTML 4.0.

IDL Definition

```
interface HTMLTableColElement : HTMLElement {
           attribute DOMString       align;
           attribute DOMString       ch;
           attribute DOMString       chOff;
           attribute long            span;
           attribute DOMString       vAlign;
           attribute DOMString       width;
};
```

Attributes

`align` of type `DOMString`
Horizontal alignment of cell data in column. See the align attribute definition in HTML 4.0.

`ch` of type `DOMString`
Alignment character for cells in a column. See the char attribute definition in HTML 4.0.

`chOff` of type `DOMString`
: Offset of alignment character. See the charoff attribute definition in HTML 4.0.

`span` of type `long`
: Indicates the number of columns in a group or affected by a grouping. See the span attribute definition in HTML 4.0.

`vAlign` of type `DOMString`
: Vertical alignment of cell data in column. See the valign attribute definition in HTML 4.0.

`width` of type `DOMString`
: Default column width. See the width attribute definition in HTML 4.0.

Interface *HTMLTableSectionElement*

The `THEAD`, `TFOOT`, and `TBODY` elements.

IDL Definition

```
interface HTMLTableSectionElement : HTMLElement {
           attribute DOMString       align;
           attribute DOMString       ch;
           attribute DOMString       chOff;
           attribute DOMString       vAlign;
  readonly attribute HTMLCollection  rows;
  HTMLElement        insertRow(in long index)
                                       raises(DOMException);
  void               deleteRow(in long index)
                                       raises(DOMException);
};
```

Attributes

`align` of type `DOMString`
: Horizontal alignment of data in cells. See the `align` attribute for HTMLTheadElement for details.

`ch` of type `DOMString`
: Alignment character for cells in a column. See the char attribute definition in HTML 4.0.

`chOff` of type `DOMString`
: Offset of alignment character. See the charoff attribute definition in HTML 4.0.

`rows` of type `HTMLCollection` [p.12] , readonly
: The collection of rows in this table section.

`vAlign` of type `DOMString`
: Vertical alignment of data in cells. See the `valign` attribute for HTMLTheadElement for details.

Methods

`deleteRow`
: Delete a row from this section.

Parameters

`index` of type `long`
: The index of the row to be deleted. This index starts from 0 and is relative only to the rows contained inside this section, not all the rows in the table.

Exceptions

`DOMException`	INDEX_SIZE_ERR: Raised if the specified index is greater than or equal to the number of rows or if the index is negative.

No Return Value

`insertRow`

Insert a row into this section. The new row is inserted immediately before the current `index`th row in this section. If `index` is equal to the number of rows in this section, the new row is appended.

Parameters

`index` of type `long`

The row number where to insert a new row. This index starts from 0 and is relative only to the rows contained inside this section, not all the rows in the table.

Return Value

`HTMLElement` [p.17]	The newly created row.

Exceptions

`DOMException`	INDEX_SIZE_ERR: Raised if the specified index is greater than the number of rows of if the index is negative.

Interface *HTMLTableRowElement*

A row in a table. See the TR element definition in HTML 4.0.

IDL Definition

```
interface HTMLTableRowElement : HTMLElement {
  readonly attribute long            rowIndex;
  readonly attribute long            sectionRowIndex;
  readonly attribute HTMLCollection  cells;
           attribute DOMString       align;
           attribute DOMString       bgColor;
           attribute DOMString       ch;
           attribute DOMString       chOff;
           attribute DOMString       vAlign;
  HTMLElement        insertCell(in long index)
                                       raises(DOMException);
  void               deleteCell(in long index)
                                       raises(DOMException);
};
```

Attributes

`align` of type `DOMString`

Horizontal alignment of data within cells of this row. See the align attribute definition in HTML 4.0.

`bgColor` of type `DOMString`
: Background color for rows. See the bgcolor attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`cells` of type `HTMLCollection` [p.12] , readonly
: The collection of cells in this row.

`ch` of type `DOMString`
: Alignment character for cells in a column. See the char attribute definition in HTML 4.0.

`chOff` of type `DOMString`
: Offset of alignment character. See the charoff attribute definition in HTML 4.0.

`rowIndex` of type `long`, readonly
: The index of this row, relative to the entire table, starting from 0. This is in document tree order and not display order. The `rowIndex` does not take into account sections (`THEAD`, `TFOOT`, or `TBODY`) within the table.

`sectionRowIndex` of type `long`, readonly
: The index of this row, relative to the current section (`THEAD`, `TFOOT`, or `TBODY`), starting from 0.

`vAlign` of type `DOMString`
: Vertical alignment of data within cells of this row. See the valign attribute definition in HTML 4.0.

Methods

`deleteCell`
: Delete a cell from the current row.

Parameters

`index` of type `long`
: The index of the cell to delete, starting from 0.

Exceptions

`DOMException`	INDEX_SIZE_ERR: Raised if the specified `index` is greater than or equal to the number of cells or if the index is negative.

No Return Value

`insertCell`
: Insert an empty `TD` cell into this row. If `index` is equal to the number of cells, the new cell is appended.

Parameters

`index` of type `long`
: The place to insert the cell, starting from 0.

Return Value

`HTMLElement` [p.17]	The newly created cell.

Exceptions

`DOMException`	INDEX_SIZE_ERR: Raised if the specified `index` is greater than the number of cells or if the index is negative.

Interface *HTMLTableCellElement*

The object used to represent the `TH` and `TD` elements. See the TD element definition in HTML 4.0.

IDL Definition

```
interface HTMLTableCellElement : HTMLElement {
  readonly attribute long            cellIndex;
           attribute DOMString       abbr;
           attribute DOMString       align;
           attribute DOMString       axis;
           attribute DOMString       bgColor;
           attribute DOMString       ch;
           attribute DOMString       chOff;
           attribute long            colSpan;
           attribute DOMString       headers;
           attribute DOMString       height;
           attribute boolean         noWrap;
           attribute long            rowSpan;
           attribute DOMString       scope;
           attribute DOMString       vAlign;
           attribute DOMString       width;
};
```

Attributes

`abbr` of type `DOMString`
: Abbreviation for header cells. See the abbr attribute definition in HTML 4.0.

`align` of type `DOMString`
: Horizontal alignment of data in cell. See the align attribute definition in HTML 4.0.

`axis` of type `DOMString`
: Names group of related headers. See the axis attribute definition in HTML 4.0.

`bgColor` of type `DOMString`
: Cell background color. See the bgcolor attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`cellIndex` of type `long`, readonly
: The index of this cell in the row, starting from 0. This index is in document tree order and not display order.

`ch` of type `DOMString`
: Alignment character for cells in a column. See the char attribute definition in HTML 4.0.

`chOff` of type `DOMString`
: Offset of alignment character. See the charoff attribute definition in HTML 4.0.

`colSpan` of type `long`
: Number of columns spanned by cell. See the colspan attribute definition in HTML 4.0.

`headers` of type `DOMString`
: List of `id` attribute values for header cells. See the headers attribute definition in HTML 4.0.

`height` of type `DOMString`
: Cell height. See the height attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`noWrap` of type `boolean`
: Suppress word wrapping. See the nowrap attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`rowSpan` of type `long`
: Number of rows spanned by cell. See the rowspan attribute definition in HTML 4.0.

`scope` of type `DOMString`
: Scope covered by header cells. See the scope attribute definition in HTML 4.0.

`vAlign` of type `DOMString`
: Vertical alignment of data in cell. See the valign attribute definition in HTML 4.0.

`width` of type `DOMString`
: Cell width. See the width attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

Interface *HTMLFrameSetElement*

Create a grid of frames. See the FRAMESET element definition in HTML 4.0.

IDL Definition

```
interface HTMLFrameSetElement : HTMLElement {
           attribute DOMString       cols;
           attribute DOMString       rows;
};
```

Attributes

`cols` of type `DOMString`
: The number of columns of frames in the frameset. See the cols attribute definition in HTML 4.0.

`rows` of type `DOMString`
: The number of rows of frames in the frameset. See the rows attribute definition in HTML 4.0.

Interface *HTMLFrameElement*

Create a frame. See the FRAME element definition in HTML 4.0.

IDL Definition

```
interface HTMLFrameElement : HTMLElement {
           attribute DOMString       frameBorder;
           attribute DOMString       longDesc;
           attribute DOMString       marginHeight;
           attribute DOMString       marginWidth;
           attribute DOMString       name;
           attribute boolean         noResize;
           attribute DOMString       scrolling;
           attribute DOMString       src;
  // Introduced in DOM Level 2:
  readonly attribute Document        contentDocument;
};
```

Attributes

`contentDocument` of type `Document`, readonly, introduced in **DOM Level 2**
: The document this frame contains, if there is any and it is available, or `null` otherwise.

`frameBorder` of type `DOMString`
: Request frame borders. See the frameborder attribute definition in HTML 4.0.

`longDesc` of type `DOMString`
: URI [RFC2396] designating a long description of this image or frame. See the longdesc attribute definition in HTML 4.0.

`marginHeight` of type `DOMString`
: Frame margin height, in pixels. See the marginheight attribute definition in HTML 4.0.

`marginWidth` of type `DOMString`
: Frame margin width, in pixels. See the marginwidth attribute definition in HTML 4.0.

`name` of type `DOMString`
: The frame name (object of the `target` attribute). See the name attribute definition in HTML 4.0.

`noResize` of type `boolean`
: When true, forbid user from resizing frame. See the noresize attribute definition in HTML 4.0.

`scrolling` of type `DOMString`
: Specify whether or not the frame should have scrollbars. See the scrolling attribute definition in HTML 4.0.

`src` of type `DOMString`
: A URI [RFC2396] designating the initial frame contents. See the src attribute definition in HTML 4.0.

Interface *HTMLIFrameElement*

Inline subwindows. See the IFRAME element definition in HTML 4.0.

IDL Definition

```
interface HTMLIFrameElement : HTMLElement {
           attribute DOMString       align;
           attribute DOMString       frameBorder;
           attribute DOMString       height;
           attribute DOMString       longDesc;
           attribute DOMString       marginHeight;
           attribute DOMString       marginWidth;
           attribute DOMString       name;
           attribute DOMString       scrolling;
           attribute DOMString       src;
           attribute DOMString       width;
  // Introduced in DOM Level 2:
  readonly attribute Document        contentDocument;
};
```

Attributes

`align` of type `DOMString`
: Aligns this object (vertically or horizontally) with respect to its surrounding text. See the align attribute definition in HTML 4.0. This attribute is deprecated in HTML 4.0.

`contentDocument` of type `Document`, readonly, introduced in **DOM Level 2**
: The document this frame contains, if there is any and it is available, or `null` otherwise.

`frameBorder` of type `DOMString`
: Request frame borders. See the frameborder attribute definition in HTML 4.0.

`height` of type `DOMString`
: Frame height. See the height attribute definition in HTML 4.0.

`longDesc` of type `DOMString`
: URI [RFC2396] designating a long description of this image or frame. See the longdesc attribute definition in HTML 4.0.

`marginHeight` of type `DOMString`
: Frame margin height, in pixels. See the marginheight attribute definition in HTML 4.0.

`marginWidth` of type `DOMString`
: Frame margin width, in pixels. See the marginwidth attribute definition in HTML 4.0.

`name` of type `DOMString`
: The frame name (object of the `target` attribute). See the name attribute definition in HTML 4.0.

`scrolling` of type `DOMString`
: Specify whether or not the frame should have scrollbars. See the scrolling attribute definition in HTML 4.0.

`src` of type `DOMString`
: A URI [RFC2396] designating the initial frame contents. See the src attribute definition in HTML 4.0.

`width` of type `DOMString`
: Frame width. See the width attribute definition in HTML 4.0.

Appendix A: Changes

Editor:
Philippe Le Hégaret, W3C

A.1: Changes between DOM Level 1 and DOM Level 2

Note: The DOM Level 2 HTML module is not backward compatible with the DOM Level 1 HTML module.

OMG IDL
The DOM Level 2 specifications are now using Corba 2.3.1 instead of Corba 2.2.

XHTML 1.0 support
The DOM Level 2 HTML module supports HTML 4 as well as XHTML 1.0 documents. Therefore, case sensitivity in methods depends on the `Document` supports the feature "`XML`" as well as "`HTML`".

A.1.1: Changes to DOM Level 1 interfaces and exceptions

Interface `HTMLDocument` [p.13]
the method `getElementById` is now inherited from the `Document` interface [DOM Level 2 Core] where it was moved.

Interface `HTMLFrameElement` [p.54]
the attribute `contentDocument` was added.

Interface `HTMLIFrameElement` [p.55]
the attribute `contentDocument` was added.

Interface `HTMLInputElement` [p.26]
the attribute `type` is no longer read only.

Interface `HTMLObjectElement` [p.40]
The attribute `contentDocument` was added.
The attribute `form` was removed.

Interface `HTMLOptionElement` [p.26]
Changing the `defaultSelected` attribute does not reset the state of the form control.

Interface `HTMLTextAreaElement` [p.29]
Changing the `defaultValue` attribute does not reset the contents of the form control.

Interface `HTMLImageElement` [p.39]
The `lowSrc` attribute was removed.
The types of the attributes `border`, `height`, `hspace`, `width` and `vspace` were changed from `DOMString` to `long`.

A.1.2: New Interfaces

Interface `HTMLDOMImplementation` [p.11]
The `HTMLDOMImplementation` [p.11] interface was added to the HTML module.

A.1.2: New Interfaces

Appendix B: IDL Definitions

This appendix contains the complete OMG IDL [OMGIDL] for the Level 2 Document Object Model HTML definitions.

Unfortunately the OMG IDL in this appendix is not conformant due to problems in the validator that was used to validate Level 1. The `readOnly` attribute on the `HTMLInputElement` [p.26] and `HTMLTextAreaElement` [p.29] interfaces, as well as the `object` attribute on the `HTMLAppletElement` [p.43] interface, are not conformant with OMG IDL 2.2. The `valueType` attribute on the `HTMLParamElement` [p.42] interface is not conformant with OMG IDL 2.3.1, which hadn't been released when DOM Level 1 [DOM Level 1] was published.

The IDL files are also available as:
http://www.w3.org/TR/2001/WD-DOM-Level-2-HTML-20011207/idl.zip

html.idl:

```
// File: html.idl

#ifndef _HTML_IDL_
#define _HTML_IDL_

#include "dom.idl"

#pragma prefix "dom.w3c.org"
module html
{

  typedef dom::DOMString DOMString;
  typedef dom::Node Node;
  typedef dom::DOMImplementation DOMImplementation;
  typedef dom::Document Document;
  typedef dom::NodeList NodeList;
  typedef dom::Element Element;

  interface HTMLDocument;
  interface HTMLElement;
  interface HTMLFormElement;
  interface HTMLTableCaptionElement;
  interface HTMLTableSectionElement;

  interface HTMLCollection {
    readonly attribute unsigned long   length;
    Node               item(in unsigned long index);
    Node               namedItem(in DOMString name);
  };

  // Introduced in DOM Level 2:
  interface HTMLDOMImplementation : DOMImplementation {
    HTMLDocument       createHTMLDocument(in DOMString title);
  };

  interface HTMLDocument : Document {
```

```
           attribute DOMString       title;
  readonly attribute DOMString       referrer;
  readonly attribute DOMString       domain;
  readonly attribute DOMString       URL;
           attribute HTMLElement     body;
  readonly attribute HTMLCollection  images;
  readonly attribute HTMLCollection  applets;
  readonly attribute HTMLCollection  links;
  readonly attribute HTMLCollection  forms;
  readonly attribute HTMLCollection  anchors;
           attribute DOMString       cookie;
                                      // raises(dom::DOMException) on setting

  void               open();
  void               close();
  void               write(in DOMString text);
  void               writeln(in DOMString text);
  NodeList           getElementsByName(in DOMString elementName);
};

interface HTMLElement : Element {
           attribute DOMString       id;
           attribute DOMString       title;
           attribute DOMString       lang;
           attribute DOMString       dir;
           attribute DOMString       className;
};

interface HTMLHtmlElement : HTMLElement {
           attribute DOMString       version;
};

interface HTMLHeadElement : HTMLElement {
           attribute DOMString       profile;
};

interface HTMLLinkElement : HTMLElement {
           attribute boolean         disabled;
           attribute DOMString       charset;
           attribute DOMString       href;
           attribute DOMString       hreflang;
           attribute DOMString       media;
           attribute DOMString       rel;
           attribute DOMString       rev;
           attribute DOMString       target;
           attribute DOMString       type;
};

interface HTMLTitleElement : HTMLElement {
           attribute DOMString       text;
};

interface HTMLMetaElement : HTMLElement {
           attribute DOMString       content;
           attribute DOMString       httpEquiv;
           attribute DOMString       name;
           attribute DOMString       scheme;
```

```
};

interface HTMLBaseElement : HTMLElement {
           attribute DOMString       href;
           attribute DOMString       target;
};

interface HTMLIsIndexElement : HTMLElement {
  readonly attribute HTMLFormElement form;
           attribute DOMString       prompt;
};

interface HTMLStyleElement : HTMLElement {
           attribute boolean         disabled;
           attribute DOMString       media;
           attribute DOMString       type;
};

interface HTMLBodyElement : HTMLElement {
           attribute DOMString       aLink;
           attribute DOMString       background;
           attribute DOMString       bgColor;
           attribute DOMString       link;
           attribute DOMString       text;
           attribute DOMString       vLink;
};

interface HTMLFormElement : HTMLElement {
  readonly attribute HTMLCollection  elements;
  readonly attribute long            length;
           attribute DOMString       name;
           attribute DOMString       acceptCharset;
           attribute DOMString       action;
           attribute DOMString       enctype;
           attribute DOMString       method;
           attribute DOMString       target;
  void               submit();
  void               reset();
};

interface HTMLSelectElement : HTMLElement {
  readonly attribute DOMString       type;
           attribute long            selectedIndex;
           attribute DOMString       value;
  readonly attribute long            length;
  readonly attribute HTMLFormElement form;
  readonly attribute HTMLCollection  options;
           attribute boolean         disabled;
           attribute boolean         multiple;
           attribute DOMString       name;
           attribute long            size;
           attribute long            tabIndex;
  void               add(in HTMLElement element,
                         in HTMLElement before)
                                      raises(dom::DOMException);
  void               remove(in long index);
  void               blur();
```

```
  void               focus();
};

interface HTMLOptGroupElement : HTMLElement {
           attribute boolean         disabled;
           attribute DOMString       label;
};

interface HTMLOptionElement : HTMLElement {
  readonly attribute HTMLFormElement form;
           attribute boolean         defaultSelected;
  readonly attribute DOMString       text;
  readonly attribute long            index;
           attribute boolean         disabled;
           attribute DOMString       label;
           attribute boolean         selected;
           attribute DOMString       value;
};

interface HTMLInputElement : HTMLElement {
           attribute DOMString       defaultValue;
           attribute boolean         defaultChecked;
  readonly attribute HTMLFormElement form;
           attribute DOMString       accept;
           attribute DOMString       accessKey;
           attribute DOMString       align;
           attribute DOMString       alt;
           attribute boolean         checked;
           attribute boolean         disabled;
           attribute long            maxLength;
           attribute DOMString       name;
           attribute boolean         readOnly;
           attribute DOMString       size;
           attribute DOMString       src;
           attribute long            tabIndex;
  // Modified in DOM Level 2:
           attribute DOMString       type;
           attribute DOMString       useMap;
           attribute DOMString       value;
  void               blur();
  void               focus();
  void               select();
  void               click();
};

interface HTMLTextAreaElement : HTMLElement {
           attribute DOMString       defaultValue;
  readonly attribute HTMLFormElement form;
           attribute DOMString       accessKey;
           attribute long            cols;
           attribute boolean         disabled;
           attribute DOMString       name;
           attribute boolean         readOnly;
           attribute long            rows;
           attribute long            tabIndex;
  readonly attribute DOMString       type;
           attribute DOMString       value;
```

```
  void               blur();
  void               focus();
  void               select();
};

interface HTMLButtonElement : HTMLElement {
  readonly attribute HTMLFormElement form;
           attribute DOMString       accessKey;
           attribute boolean         disabled;
           attribute DOMString       name;
           attribute long            tabIndex;
  readonly attribute DOMString       type;
           attribute DOMString       value;
};

interface HTMLLabelElement : HTMLElement {
  readonly attribute HTMLFormElement form;
           attribute DOMString       accessKey;
           attribute DOMString       htmlFor;
};

interface HTMLFieldSetElement : HTMLElement {
  readonly attribute HTMLFormElement form;
};

interface HTMLLegendElement : HTMLElement {
  readonly attribute HTMLFormElement form;
           attribute DOMString       accessKey;
           attribute DOMString       align;
};

interface HTMLUListElement : HTMLElement {
           attribute boolean         compact;
           attribute DOMString       type;
};

interface HTMLOListElement : HTMLElement {
           attribute boolean         compact;
           attribute long            start;
           attribute DOMString       type;
};

interface HTMLDListElement : HTMLElement {
           attribute boolean         compact;
};

interface HTMLDirectoryElement : HTMLElement {
           attribute boolean         compact;
};

interface HTMLMenuElement : HTMLElement {
           attribute boolean         compact;
};

interface HTMLLIElement : HTMLElement {
           attribute DOMString       type;
           attribute long            value;
```

```
};

interface HTMLDivElement : HTMLElement {
           attribute DOMString       align;
};

interface HTMLParagraphElement : HTMLElement {
           attribute DOMString       align;
};

interface HTMLHeadingElement : HTMLElement {
           attribute DOMString       align;
};

interface HTMLQuoteElement : HTMLElement {
           attribute DOMString       cite;
};

interface HTMLPreElement : HTMLElement {
           attribute long            width;
};

interface HTMLBRElement : HTMLElement {
           attribute DOMString       clear;
};

interface HTMLBaseFontElement : HTMLElement {
           attribute DOMString       color;
           attribute DOMString       face;
           attribute DOMString       size;
};

interface HTMLFontElement : HTMLElement {
           attribute DOMString       color;
           attribute DOMString       face;
           attribute DOMString       size;
};

interface HTMLHRElement : HTMLElement {
           attribute DOMString       align;
           attribute boolean         noShade;
           attribute DOMString       size;
           attribute DOMString       width;
};

interface HTMLModElement : HTMLElement {
           attribute DOMString       cite;
           attribute DOMString       dateTime;
};

interface HTMLAnchorElement : HTMLElement {
           attribute DOMString       accessKey;
           attribute DOMString       charset;
           attribute DOMString       coords;
           attribute DOMString       href;
           attribute DOMString       hreflang;
           attribute DOMString       name;
```

```
            attribute DOMString       rel;
            attribute DOMString       rev;
            attribute DOMString       shape;
            attribute long            tabIndex;
            attribute DOMString       target;
            attribute DOMString       type;
  void              blur();
  void              focus();
};

interface HTMLImageElement : HTMLElement {
            attribute DOMString       name;
            attribute DOMString       align;
            attribute DOMString       alt;
            attribute long            border;
            attribute long            height;
            attribute long            hspace;
            attribute boolean         isMap;
            attribute DOMString       longDesc;
            attribute DOMString       src;
            attribute DOMString       useMap;
            attribute long            vspace;
            attribute long            width;
};

interface HTMLObjectElement : HTMLElement {
            attribute DOMString       code;
            attribute DOMString       align;
            attribute DOMString       archive;
            attribute DOMString       border;
            attribute DOMString       codeBase;
            attribute DOMString       codeType;
            attribute DOMString       data;
            attribute boolean         declare;
            attribute DOMString       height;
            attribute DOMString       hspace;
            attribute DOMString       name;
            attribute DOMString       standby;
            attribute long            tabIndex;
            attribute DOMString       type;
            attribute DOMString       useMap;
            attribute DOMString       vspace;
            attribute DOMString       width;
  // Introduced in DOM Level 2:
  readonly attribute Document        contentDocument;
};

interface HTMLParamElement : HTMLElement {
            attribute DOMString       name;
            attribute DOMString       type;
            attribute DOMString       value;
            attribute DOMString       valueType;
};

interface HTMLAppletElement : HTMLElement {
            attribute DOMString       align;
            attribute DOMString       alt;
```

```
           attribute DOMString       archive;
           attribute DOMString       code;
           attribute DOMString       codeBase;
           attribute DOMString       height;
           attribute DOMString       hspace;
           attribute DOMString       name;
           attribute DOMString       object;
           attribute DOMString       vspace;
           attribute DOMString       width;
};

interface HTMLMapElement : HTMLElement {
  readonly attribute HTMLCollection  areas;
           attribute DOMString       name;
};

interface HTMLAreaElement : HTMLElement {
           attribute DOMString       accessKey;
           attribute DOMString       alt;
           attribute DOMString       coords;
           attribute DOMString       href;
           attribute boolean         noHref;
           attribute DOMString       shape;
           attribute long            tabIndex;
           attribute DOMString       target;
};

interface HTMLScriptElement : HTMLElement {
           attribute DOMString       text;
           attribute DOMString       htmlFor;
           attribute DOMString       event;
           attribute DOMString       charset;
           attribute boolean         defer;
           attribute DOMString       src;
           attribute DOMString       type;
};

interface HTMLTableElement : HTMLElement {
           attribute HTMLTableCaptionElement caption;
           attribute HTMLTableSectionElement tHead;
           attribute HTMLTableSectionElement tFoot;
  readonly attribute HTMLCollection  rows;
  readonly attribute HTMLCollection  tBodies;
           attribute DOMString       align;
           attribute DOMString       bgColor;
           attribute DOMString       border;
           attribute DOMString       cellPadding;
           attribute DOMString       cellSpacing;
           attribute DOMString       frame;
           attribute DOMString       rules;
           attribute DOMString       summary;
           attribute DOMString       width;
  HTMLElement        createTHead();
  void               deleteTHead();
  HTMLElement        createTFoot();
  void               deleteTFoot();
  HTMLElement        createCaption();
```

```
  void               deleteCaption();
  HTMLElement        insertRow(in long index)
                                   raises(dom::DOMException);
  void               deleteRow(in long index)
                                   raises(dom::DOMException);
};

interface HTMLTableCaptionElement : HTMLElement {
           attribute DOMString       align;
};

interface HTMLTableColElement : HTMLElement {
           attribute DOMString       align;
           attribute DOMString       ch;
           attribute DOMString       chOff;
           attribute long            span;
           attribute DOMString       vAlign;
           attribute DOMString       width;
};

interface HTMLTableSectionElement : HTMLElement {
           attribute DOMString       align;
           attribute DOMString       ch;
           attribute DOMString       chOff;
           attribute DOMString       vAlign;
  readonly attribute HTMLCollection  rows;
  HTMLElement        insertRow(in long index)
                                   raises(dom::DOMException);
  void               deleteRow(in long index)
                                   raises(dom::DOMException);
};

interface HTMLTableRowElement : HTMLElement {
  readonly attribute long            rowIndex;
  readonly attribute long            sectionRowIndex;
  readonly attribute HTMLCollection  cells;
           attribute DOMString       align;
           attribute DOMString       bgColor;
           attribute DOMString       ch;
           attribute DOMString       chOff;
           attribute DOMString       vAlign;
  HTMLElement        insertCell(in long index)
                                   raises(dom::DOMException);
  void               deleteCell(in long index)
                                   raises(dom::DOMException);
};

interface HTMLTableCellElement : HTMLElement {
  readonly attribute long            cellIndex;
           attribute DOMString       abbr;
           attribute DOMString       align;
           attribute DOMString       axis;
           attribute DOMString       bgColor;
           attribute DOMString       ch;
           attribute DOMString       chOff;
           attribute long            colSpan;
           attribute DOMString       headers;
```

```
            attribute DOMString       height;
            attribute boolean         noWrap;
            attribute long            rowSpan;
            attribute DOMString       scope;
            attribute DOMString       vAlign;
            attribute DOMString       width;
  };

  interface HTMLFrameSetElement : HTMLElement {
            attribute DOMString       cols;
            attribute DOMString       rows;
  };

  interface HTMLFrameElement : HTMLElement {
            attribute DOMString       frameBorder;
            attribute DOMString       longDesc;
            attribute DOMString       marginHeight;
            attribute DOMString       marginWidth;
            attribute DOMString       name;
            attribute boolean         noResize;
            attribute DOMString       scrolling;
            attribute DOMString       src;
    // Introduced in DOM Level 2:
    readonly attribute Document        contentDocument;
  };

  interface HTMLIFrameElement : HTMLElement {
            attribute DOMString       align;
            attribute DOMString       frameBorder;
            attribute DOMString       height;
            attribute DOMString       longDesc;
            attribute DOMString       marginHeight;
            attribute DOMString       marginWidth;
            attribute DOMString       name;
            attribute DOMString       scrolling;
            attribute DOMString       src;
            attribute DOMString       width;
    // Introduced in DOM Level 2:
    readonly attribute Document        contentDocument;
  };
};

#endif // _HTML_IDL_
```

Appendix C: Java Language Binding

This appendix contains the complete Java Language [Java] binding for the Level 2 Document Object Model HTML.

The Java files are also available as
http://www.w3.org/TR/2001/WD-DOM-Level-2-HTML-20011207/java-binding.zip

org/w3c/dom/html/HTMLDOMImplementation.java:

```
package org.w3c.dom.html;

import org.w3c.dom.DOMImplementation;

public interface HTMLDOMImplementation extends DOMImplementation {
    public HTMLDocument createHTMLDocument(String title);

}
```

org/w3c/dom/html/HTMLCollection.java:

```
package org.w3c.dom.html;

import org.w3c.dom.Node;

public interface HTMLCollection {
    public int getLength();

    public Node item(int index);

    public Node namedItem(String name);

}
```

org/w3c/dom/html/HTMLDocument.java:

```
package org.w3c.dom.html;

import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.DOMException;

public interface HTMLDocument extends Document {
    public String getTitle();
    public void setTitle(String title);

    public String getReferrer();

    public String getDomain();

    public String getURL();

    public HTMLElement getBody();
```

```
    public void setBody(HTMLElement body);

    public HTMLCollection getImages();

    public HTMLCollection getApplets();

    public HTMLCollection getLinks();

    public HTMLCollection getForms();

    public HTMLCollection getAnchors();

    public String getCookie();
    public void setCookie(String cookie)
                          throws DOMException;

    public void open();

    public void close();

    public void write(String text);

    public void writeln(String text);

    public NodeList getElementsByName(String elementName);

}
```

org/w3c/dom/html/HTMLElement.java:

```
package org.w3c.dom.html;

import org.w3c.dom.Element;

public interface HTMLElement extends Element {
    public String getId();
    public void setId(String id);

    public String getTitle();
    public void setTitle(String title);

    public String getLang();
    public void setLang(String lang);

    public String getDir();
    public void setDir(String dir);

    public String getClassName();
    public void setClassName(String className);

}
```

org/w3c/dom/html/HTMLHtmlElement.java:

```
package org.w3c.dom.html;

public interface HTMLHtmlElement extends HTMLElement {
    public String getVersion();
    public void setVersion(String version);

}
```

org/w3c/dom/html/HTMLHeadElement.java:

```
package org.w3c.dom.html;

public interface HTMLHeadElement extends HTMLElement {
    public String getProfile();
    public void setProfile(String profile);

}
```

org/w3c/dom/html/HTMLLinkElement.java:

```
package org.w3c.dom.html;

public interface HTMLLinkElement extends HTMLElement {
    public boolean getDisabled();
    public void setDisabled(boolean disabled);

    public String getCharset();
    public void setCharset(String charset);

    public String getHref();
    public void setHref(String href);

    public String getHreflang();
    public void setHreflang(String hreflang);

    public String getMedia();
    public void setMedia(String media);

    public String getRel();
    public void setRel(String rel);

    public String getRev();
    public void setRev(String rev);

    public String getTarget();
    public void setTarget(String target);

    public String getType();
    public void setType(String type);

}
```

org/w3c/dom/html/HTMLTitleElement.java:

```
package org.w3c.dom.html;

public interface HTMLTitleElement extends HTMLElement {
    public String getText();
    public void setText(String text);

}
```

org/w3c/dom/html/HTMLMetaElement.java:

```
package org.w3c.dom.html;

public interface HTMLMetaElement extends HTMLElement {
    public String getContent();
    public void setContent(String content);

    public String getHttpEquiv();
    public void setHttpEquiv(String httpEquiv);

    public String getName();
    public void setName(String name);

    public String getScheme();
    public void setScheme(String scheme);

}
```

org/w3c/dom/html/HTMLBaseElement.java:

```
package org.w3c.dom.html;

public interface HTMLBaseElement extends HTMLElement {
    public String getHref();
    public void setHref(String href);

    public String getTarget();
    public void setTarget(String target);

}
```

org/w3c/dom/html/HTMLIsIndexElement.java:

```
package org.w3c.dom.html;

public interface HTMLIsIndexElement extends HTMLElement {
    public HTMLFormElement getForm();

    public String getPrompt();
    public void setPrompt(String prompt);

}
```

org/w3c/dom/html/HTMLStyleElement.java:

```
package org.w3c.dom.html;

public interface HTMLStyleElement extends HTMLElement {
    public boolean getDisabled();
    public void setDisabled(boolean disabled);

    public String getMedia();
    public void setMedia(String media);

    public String getType();
    public void setType(String type);

}
```

org/w3c/dom/html/HTMLBodyElement.java:

```
package org.w3c.dom.html;

public interface HTMLBodyElement extends HTMLElement {
    public String getALink();
    public void setALink(String aLink);

    public String getBackground();
    public void setBackground(String background);

    public String getBgColor();
    public void setBgColor(String bgColor);

    public String getLink();
    public void setLink(String link);

    public String getText();
    public void setText(String text);

    public String getVLink();
    public void setVLink(String vLink);

}
```

org/w3c/dom/html/HTMLFormElement.java:

```
package org.w3c.dom.html;

public interface HTMLFormElement extends HTMLElement {
    public HTMLCollection getElements();

    public int getLength();

    public String getName();
    public void setName(String name);

    public String getAcceptCharset();
    public void setAcceptCharset(String acceptCharset);
```

```
    public String getAction();
    public void setAction(String action);

    public String getEnctype();
    public void setEnctype(String enctype);

    public String getMethod();
    public void setMethod(String method);

    public String getTarget();
    public void setTarget(String target);

    public void submit();

    public void reset();

}
```

org/w3c/dom/html/HTMLSelectElement.java:

```
package org.w3c.dom.html;

import org.w3c.dom.DOMException;

public interface HTMLSelectElement extends HTMLElement {
    public String getType();

    public int getSelectedIndex();
    public void setSelectedIndex(int selectedIndex);

    public String getValue();
    public void setValue(String value);

    public int getLength();

    public HTMLFormElement getForm();

    public HTMLCollection getOptions();

    public boolean getDisabled();
    public void setDisabled(boolean disabled);

    public boolean getMultiple();
    public void setMultiple(boolean multiple);

    public String getName();
    public void setName(String name);

    public int getSize();
    public void setSize(int size);

    public int getTabIndex();
    public void setTabIndex(int tabIndex);

    public void add(HTMLElement element,
```

```
                    HTMLElement before)
                    throws DOMException;

    public void remove(int index);

    public void blur();

    public void focus();

}
```

org/w3c/dom/html/HTMLOptGroupElement.java:

```
package org.w3c.dom.html;

public interface HTMLOptGroupElement extends HTMLElement {
    public boolean getDisabled();
    public void setDisabled(boolean disabled);

    public String getLabel();
    public void setLabel(String label);

}
```

org/w3c/dom/html/HTMLOptionElement.java:

```
package org.w3c.dom.html;

public interface HTMLOptionElement extends HTMLElement {
    public HTMLFormElement getForm();

    public boolean getDefaultSelected();
    public void setDefaultSelected(boolean defaultSelected);

    public String getText();

    public int getIndex();

    public boolean getDisabled();
    public void setDisabled(boolean disabled);

    public String getLabel();
    public void setLabel(String label);

    public boolean getSelected();
    public void setSelected(boolean selected);

    public String getValue();
    public void setValue(String value);

}
```

org/w3c/dom/html/HTMLInputElement.java:

```
package org.w3c.dom.html;

public interface HTMLInputElement extends HTMLElement {
    public String getDefaultValue();
    public void setDefaultValue(String defaultValue);

    public boolean getDefaultChecked();
    public void setDefaultChecked(boolean defaultChecked);

    public HTMLFormElement getForm();

    public String getAccept();
    public void setAccept(String accept);

    public String getAccessKey();
    public void setAccessKey(String accessKey);

    public String getAlign();
    public void setAlign(String align);

    public String getAlt();
    public void setAlt(String alt);

    public boolean getChecked();
    public void setChecked(boolean checked);

    public boolean getDisabled();
    public void setDisabled(boolean disabled);

    public int getMaxLength();
    public void setMaxLength(int maxLength);

    public String getName();
    public void setName(String name);

    public boolean getReadOnly();
    public void setReadOnly(boolean readOnly);

    public String getSize();
    public void setSize(String size);

    public String getSrc();
    public void setSrc(String src);

    public int getTabIndex();
    public void setTabIndex(int tabIndex);

    public String getType();
    public void setType(String type);

    public String getUseMap();
    public void setUseMap(String useMap);

    public String getValue();
```

```
    public void setValue(String value);

    public void blur();

    public void focus();

    public void select();

    public void click();

}
```

org/w3c/dom/html/HTMLTextAreaElement.java:

```
package org.w3c.dom.html;

public interface HTMLTextAreaElement extends HTMLElement {
    public String getDefaultValue();
    public void setDefaultValue(String defaultValue);

    public HTMLFormElement getForm();

    public String getAccessKey();
    public void setAccessKey(String accessKey);

    public int getCols();
    public void setCols(int cols);

    public boolean getDisabled();
    public void setDisabled(boolean disabled);

    public String getName();
    public void setName(String name);

    public boolean getReadOnly();
    public void setReadOnly(boolean readOnly);

    public int getRows();
    public void setRows(int rows);

    public int getTabIndex();
    public void setTabIndex(int tabIndex);

    public String getType();

    public String getValue();
    public void setValue(String value);

    public void blur();

    public void focus();

    public void select();

}
```

org/w3c/dom/html/HTMLButtonElement.java:

```
package org.w3c.dom.html;

public interface HTMLButtonElement extends HTMLElement {
    public HTMLFormElement getForm();

    public String getAccessKey();
    public void setAccessKey(String accessKey);

    public boolean getDisabled();
    public void setDisabled(boolean disabled);

    public String getName();
    public void setName(String name);

    public int getTabIndex();
    public void setTabIndex(int tabIndex);

    public String getType();

    public String getValue();
    public void setValue(String value);

}
```

org/w3c/dom/html/HTMLLabelElement.java:

```
package org.w3c.dom.html;

public interface HTMLLabelElement extends HTMLElement {
    public HTMLFormElement getForm();

    public String getAccessKey();
    public void setAccessKey(String accessKey);

    public String getHtmlFor();
    public void setHtmlFor(String htmlFor);

}
```

org/w3c/dom/html/HTMLFieldSetElement.java:

```
package org.w3c.dom.html;

public interface HTMLFieldSetElement extends HTMLElement {
    public HTMLFormElement getForm();

}
```

org/w3c/dom/html/HTMLLegendElement.java:

```
package org.w3c.dom.html;

public interface HTMLLegendElement extends HTMLElement {
    public HTMLFormElement getForm();

    public String getAccessKey();
    public void setAccessKey(String accessKey);

    public String getAlign();
    public void setAlign(String align);

}
```

org/w3c/dom/html/HTMLUListElement.java:

```
package org.w3c.dom.html;

public interface HTMLUListElement extends HTMLElement {
    public boolean getCompact();
    public void setCompact(boolean compact);

    public String getType();
    public void setType(String type);

}
```

org/w3c/dom/html/HTMLOListElement.java:

```
package org.w3c.dom.html;

public interface HTMLOListElement extends HTMLElement {
    public boolean getCompact();
    public void setCompact(boolean compact);

    public int getStart();
    public void setStart(int start);

    public String getType();
    public void setType(String type);

}
```

org/w3c/dom/html/HTMLDListElement.java:

```
package org.w3c.dom.html;

public interface HTMLDListElement extends HTMLElement {
    public boolean getCompact();
    public void setCompact(boolean compact);

}
```

org/w3c/dom/html/HTMLDirectoryElement.java:

```
package org.w3c.dom.html;

public interface HTMLDirectoryElement extends HTMLElement {
    public boolean getCompact();
    public void setCompact(boolean compact);

}
```

org/w3c/dom/html/HTMLMenuElement.java:

```
package org.w3c.dom.html;

public interface HTMLMenuElement extends HTMLElement {
    public boolean getCompact();
    public void setCompact(boolean compact);

}
```

org/w3c/dom/html/HTMLLIElement.java:

```
package org.w3c.dom.html;

public interface HTMLLIElement extends HTMLElement {
    public String getType();
    public void setType(String type);

    public int getValue();
    public void setValue(int value);

}
```

org/w3c/dom/html/HTMLDivElement.java:

```
package org.w3c.dom.html;

public interface HTMLDivElement extends HTMLElement {
    public String getAlign();
    public void setAlign(String align);

}
```

org/w3c/dom/html/HTMLParagraphElement.java:

```
package org.w3c.dom.html;

public interface HTMLParagraphElement extends HTMLElement {
    public String getAlign();
    public void setAlign(String align);

}
```

org/w3c/dom/html/HTMLHeadingElement.java:

```
package org.w3c.dom.html;

public interface HTMLHeadingElement extends HTMLElement {
    public String getAlign();
    public void setAlign(String align);

}
```

org/w3c/dom/html/HTMLQuoteElement.java:

```
package org.w3c.dom.html;

public interface HTMLQuoteElement extends HTMLElement {
    public String getCite();
    public void setCite(String cite);

}
```

org/w3c/dom/html/HTMLPreElement.java:

```
package org.w3c.dom.html;

public interface HTMLPreElement extends HTMLElement {
    public int getWidth();
    public void setWidth(int width);

}
```

org/w3c/dom/html/HTMLBRElement.java:

```
package org.w3c.dom.html;

public interface HTMLBRElement extends HTMLElement {
    public String getClear();
    public void setClear(String clear);

}
```

org/w3c/dom/html/HTMLBaseFontElement.java:

```
package org.w3c.dom.html;

public interface HTMLBaseFontElement extends HTMLElement {
    public String getColor();
    public void setColor(String color);

    public String getFace();
    public void setFace(String face);
```

```
    public String getSize();
    public void setSize(String size);

}
```

org/w3c/dom/html/HTMLFontElement.java:

```
package org.w3c.dom.html;

public interface HTMLFontElement extends HTMLElement {
    public String getColor();
    public void setColor(String color);

    public String getFace();
    public void setFace(String face);

    public String getSize();
    public void setSize(String size);

}
```

org/w3c/dom/html/HTMLHRElement.java:

```
package org.w3c.dom.html;

public interface HTMLHRElement extends HTMLElement {
    public String getAlign();
    public void setAlign(String align);

    public boolean getNoShade();
    public void setNoShade(boolean noShade);

    public String getSize();
    public void setSize(String size);

    public String getWidth();
    public void setWidth(String width);

}
```

org/w3c/dom/html/HTMLModElement.java:

```
package org.w3c.dom.html;

public interface HTMLModElement extends HTMLElement {
    public String getCite();
    public void setCite(String cite);

    public String getDateTime();
    public void setDateTime(String dateTime);

}
```

org/w3c/dom/html/HTMLAnchorElement.java:

```
package org.w3c.dom.html;

public interface HTMLAnchorElement extends HTMLElement {
    public String getAccessKey();
    public void setAccessKey(String accessKey);

    public String getCharset();
    public void setCharset(String charset);

    public String getCoords();
    public void setCoords(String coords);

    public String getHref();
    public void setHref(String href);

    public String getHreflang();
    public void setHreflang(String hreflang);

    public String getName();
    public void setName(String name);

    public String getRel();
    public void setRel(String rel);

    public String getRev();
    public void setRev(String rev);

    public String getShape();
    public void setShape(String shape);

    public int getTabIndex();
    public void setTabIndex(int tabIndex);

    public String getTarget();
    public void setTarget(String target);

    public String getType();
    public void setType(String type);

    public void blur();

    public void focus();

}
```

org/w3c/dom/html/HTMLImageElement.java:

```
package org.w3c.dom.html;

public interface HTMLImageElement extends HTMLElement {
    public String getName();
    public void setName(String name);

    public String getAlign();
```

```
    public void setAlign(String align);

    public String getAlt();
    public void setAlt(String alt);

    public int getBorder();
    public void setBorder(int border);

    public int getHeight();
    public void setHeight(int height);

    public int getHspace();
    public void setHspace(int hspace);

    public boolean getIsMap();
    public void setIsMap(boolean isMap);

    public String getLongDesc();
    public void setLongDesc(String longDesc);

    public String getSrc();
    public void setSrc(String src);

    public String getUseMap();
    public void setUseMap(String useMap);

    public int getVspace();
    public void setVspace(int vspace);

    public int getWidth();
    public void setWidth(int width);

}
```

org/w3c/dom/html/HTMLObjectElement.java:

```
package org.w3c.dom.html;

import org.w3c.dom.Document;

public interface HTMLObjectElement extends HTMLElement {
    public String getCode();
    public void setCode(String code);

    public String getAlign();
    public void setAlign(String align);

    public String getArchive();
    public void setArchive(String archive);

    public String getBorder();
    public void setBorder(String border);

    public String getCodeBase();
    public void setCodeBase(String codeBase);
```

```
    public String getCodeType();
    public void setCodeType(String codeType);

    public String getData();
    public void setData(String data);

    public boolean getDeclare();
    public void setDeclare(boolean declare);

    public String getHeight();
    public void setHeight(String height);

    public String getHspace();
    public void setHspace(String hspace);

    public String getName();
    public void setName(String name);

    public String getStandby();
    public void setStandby(String standby);

    public int getTabIndex();
    public void setTabIndex(int tabIndex);

    public String getType();
    public void setType(String type);

    public String getUseMap();
    public void setUseMap(String useMap);

    public String getVspace();
    public void setVspace(String vspace);

    public String getWidth();
    public void setWidth(String width);

    public Document getContentDocument();

}
```

org/w3c/dom/html/HTMLParamElement.java:

```
package org.w3c.dom.html;

public interface HTMLParamElement extends HTMLElement {
    public String getName();
    public void setName(String name);

    public String getType();
    public void setType(String type);

    public String getValue();
    public void setValue(String value);
```

```
    public String getValueType();
    public void setValueType(String valueType);

}
```

org/w3c/dom/html/HTMLAppletElement.java:

```
package org.w3c.dom.html;

public interface HTMLAppletElement extends HTMLElement {
    public String getAlign();
    public void setAlign(String align);

    public String getAlt();
    public void setAlt(String alt);

    public String getArchive();
    public void setArchive(String archive);

    public String getCode();
    public void setCode(String code);

    public String getCodeBase();
    public void setCodeBase(String codeBase);

    public String getHeight();
    public void setHeight(String height);

    public String getHspace();
    public void setHspace(String hspace);

    public String getName();
    public void setName(String name);

    public String getObject();
    public void setObject(String object);

    public String getVspace();
    public void setVspace(String vspace);

    public String getWidth();
    public void setWidth(String width);

}
```

org/w3c/dom/html/HTMLMapElement.java:

```
package org.w3c.dom.html;

public interface HTMLMapElement extends HTMLElement {
    public HTMLCollection getAreas();

    public String getName();
    public void setName(String name);

}
```

org/w3c/dom/html/HTMLAreaElement.java:

```
package org.w3c.dom.html;

public interface HTMLAreaElement extends HTMLElement {
    public String getAccessKey();
    public void setAccessKey(String accessKey);

    public String getAlt();
    public void setAlt(String alt);

    public String getCoords();
    public void setCoords(String coords);

    public String getHref();
    public void setHref(String href);

    public boolean getNoHref();
    public void setNoHref(boolean noHref);

    public String getShape();
    public void setShape(String shape);

    public int getTabIndex();
    public void setTabIndex(int tabIndex);

    public String getTarget();
    public void setTarget(String target);

}
```

org/w3c/dom/html/HTMLScriptElement.java:

```
package org.w3c.dom.html;

public interface HTMLScriptElement extends HTMLElement {
    public String getText();
    public void setText(String text);

    public String getHtmlFor();
    public void setHtmlFor(String htmlFor);

    public String getEvent();
    public void setEvent(String event);

    public String getCharset();
    public void setCharset(String charset);

    public boolean getDefer();
    public void setDefer(boolean defer);

    public String getSrc();
    public void setSrc(String src);
```

```
    public String getType();
    public void setType(String type);

}
```

org/w3c/dom/html/HTMLTableElement.java:

```
package org.w3c.dom.html;

import org.w3c.dom.DOMException;

public interface HTMLTableElement extends HTMLElement {
    public HTMLTableCaptionElement getCaption();
    public void setCaption(HTMLTableCaptionElement caption);

    public HTMLTableSectionElement getTHead();
    public void setTHead(HTMLTableSectionElement tHead);

    public HTMLTableSectionElement getTFoot();
    public void setTFoot(HTMLTableSectionElement tFoot);

    public HTMLCollection getRows();

    public HTMLCollection getTBodies();

    public String getAlign();
    public void setAlign(String align);

    public String getBgColor();
    public void setBgColor(String bgColor);

    public String getBorder();
    public void setBorder(String border);

    public String getCellPadding();
    public void setCellPadding(String cellPadding);

    public String getCellSpacing();
    public void setCellSpacing(String cellSpacing);

    public String getFrame();
    public void setFrame(String frame);

    public String getRules();
    public void setRules(String rules);

    public String getSummary();
    public void setSummary(String summary);

    public String getWidth();
    public void setWidth(String width);

    public HTMLElement createTHead();

    public void deleteTHead();
```

```
    public HTMLElement createTFoot();

    public void deleteTFoot();

    public HTMLElement createCaption();

    public void deleteCaption();

    public HTMLElement insertRow(int index)
                                 throws DOMException;

    public void deleteRow(int index)
                          throws DOMException;

}
```

org/w3c/dom/html/HTMLTableCaptionElement.java:

```
package org.w3c.dom.html;

public interface HTMLTableCaptionElement extends HTMLElement {
    public String getAlign();
    public void setAlign(String align);

}
```

org/w3c/dom/html/HTMLTableColElement.java:

```
package org.w3c.dom.html;

public interface HTMLTableColElement extends HTMLElement {
    public String getAlign();
    public void setAlign(String align);

    public String getCh();
    public void setCh(String ch);

    public String getChOff();
    public void setChOff(String chOff);

    public int getSpan();
    public void setSpan(int span);

    public String getVAlign();
    public void setVAlign(String vAlign);

    public String getWidth();
    public void setWidth(String width);

}
```

org/w3c/dom/html/HTMLTableSectionElement.java:

```
package org.w3c.dom.html;

import org.w3c.dom.DOMException;

public interface HTMLTableSectionElement extends HTMLElement {
    public String getAlign();
    public void setAlign(String align);

    public String getCh();
    public void setCh(String ch);

    public String getChOff();
    public void setChOff(String chOff);

    public String getVAlign();
    public void setVAlign(String vAlign);

    public HTMLCollection getRows();

    public HTMLElement insertRow(int index)
                                 throws DOMException;

    public void deleteRow(int index)
                          throws DOMException;

}
```

org/w3c/dom/html/HTMLTableRowElement.java:

```
package org.w3c.dom.html;

import org.w3c.dom.DOMException;

public interface HTMLTableRowElement extends HTMLElement {
    public int getRowIndex();

    public int getSectionRowIndex();

    public HTMLCollection getCells();

    public String getAlign();
    public void setAlign(String align);

    public String getBgColor();
    public void setBgColor(String bgColor);

    public String getCh();
    public void setCh(String ch);

    public String getChOff();
    public void setChOff(String chOff);

    public String getVAlign();
    public void setVAlign(String vAlign);
```

```
    public HTMLElement insertCell(int index)
                                  throws DOMException;

    public void deleteCell(int index)
                           throws DOMException;

}
```

org/w3c/dom/html/HTMLTableCellElement.java:

```
package org.w3c.dom.html;

public interface HTMLTableCellElement extends HTMLElement {
    public int getCellIndex();

    public String getAbbr();
    public void setAbbr(String abbr);

    public String getAlign();
    public void setAlign(String align);

    public String getAxis();
    public void setAxis(String axis);

    public String getBgColor();
    public void setBgColor(String bgColor);

    public String getCh();
    public void setCh(String ch);

    public String getChOff();
    public void setChOff(String chOff);

    public int getColSpan();
    public void setColSpan(int colSpan);

    public String getHeaders();
    public void setHeaders(String headers);

    public String getHeight();
    public void setHeight(String height);

    public boolean getNoWrap();
    public void setNoWrap(boolean noWrap);

    public int getRowSpan();
    public void setRowSpan(int rowSpan);

    public String getScope();
    public void setScope(String scope);

    public String getVAlign();
    public void setVAlign(String vAlign);
```

```
    public String getWidth();
    public void setWidth(String width);

}
```

org/w3c/dom/html/HTMLFrameSetElement.java:

```
package org.w3c.dom.html;

public interface HTMLFrameSetElement extends HTMLElement {
    public String getCols();
    public void setCols(String cols);

    public String getRows();
    public void setRows(String rows);

}
```

org/w3c/dom/html/HTMLFrameElement.java:

```
package org.w3c.dom.html;

import org.w3c.dom.Document;

public interface HTMLFrameElement extends HTMLElement {
    public String getFrameBorder();
    public void setFrameBorder(String frameBorder);

    public String getLongDesc();
    public void setLongDesc(String longDesc);

    public String getMarginHeight();
    public void setMarginHeight(String marginHeight);

    public String getMarginWidth();
    public void setMarginWidth(String marginWidth);

    public String getName();
    public void setName(String name);

    public boolean getNoResize();
    public void setNoResize(boolean noResize);

    public String getScrolling();
    public void setScrolling(String scrolling);

    public String getSrc();
    public void setSrc(String src);

    public Document getContentDocument();

}
```

org/w3c/dom/html/HTMLIFrameElement.java:

```
package org.w3c.dom.html;

import org.w3c.dom.Document;

public interface HTMLIFrameElement extends HTMLElement {
    public String getAlign();
    public void setAlign(String align);

    public String getFrameBorder();
    public void setFrameBorder(String frameBorder);

    public String getHeight();
    public void setHeight(String height);

    public String getLongDesc();
    public void setLongDesc(String longDesc);

    public String getMarginHeight();
    public void setMarginHeight(String marginHeight);

    public String getMarginWidth();
    public void setMarginWidth(String marginWidth);

    public String getName();
    public void setName(String name);

    public String getScrolling();
    public void setScrolling(String scrolling);

    public String getSrc();
    public void setSrc(String src);

    public String getWidth();
    public void setWidth(String width);

    public Document getContentDocument();

}
```

org/w3c/dom/html/HTMLIFrameElement.java:

Appendix D: ECMAScript Language Binding

This appendix contains the complete ECMAScript [ECMAScript] binding for the Level 2 Document Object Model HTML definitions.

Note: Exceptions handling is only supported by ECMAScript implementation conformant with the Standard ECMA-262 3rd. Edition ([ECMAScript]).

Objects that implement the **HTMLDOMImplementation** interface:

Objects that implement the **HTMLDOMImplementation** interface have all properties and functions of the **DOMImplementation** interface as well as the properties and functions defined below.

Functions of objects that implement the **HTMLDOMImplementation** interface:

createHTMLDocument(title)

This function returns an object that implements the **HTMLDocument** interface.

The **title** parameter is a **String**.

Objects that implement the **HTMLCollection** interface:

Properties of objects that implement the **HTMLCollection** interface:

length

This read-only property is a **Number**.

Functions of objects that implement the **HTMLCollection** interface:

item(index)

This function returns an object that implements the **Node** interface.

The **index** parameter is a **Number**.

Note: This object can also be dereferenced using square bracket notation (e.g. obj[1]). Dereferencing with an integer **index** is equivalent to invoking the **item** function with that index.

namedItem(name)

This function returns an object that implements the **Node** interface.

The **name** parameter is a **String**.

Note: This object can also be dereferenced using square bracket notation (e.g. obj["foo"]). Dereferencing using a string index is equivalent to invoking the **namedItem** function with that index.

Objects that implement the **HTMLDocument** interface:

Objects that implement the **HTMLDocument** interface have all properties and functions of the **Document** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLDocument** interface:

title

This property is a **String**.

referrer

This read-only property is a **String**.

domain

This read-only property is a **String**.

URL

This read-only property is a **String**.

body
This property is an object that implements the **HTMLElement** interface.
images
This read-only property is an object that implements the **HTMLCollection** interface.
applets
This read-only property is an object that implements the **HTMLCollection** interface.
links
This read-only property is an object that implements the **HTMLCollection** interface.
forms
This read-only property is an object that implements the **HTMLCollection** interface.
anchors
This read-only property is an object that implements the **HTMLCollection** interface.
cookie
This property is a **String** and can raise an objewct that implements **DOMException** interface on setting.

Functions of objects that implement the **HTMLDocument** interface:

open()
This function has no return value.
close()
This function has no return value.
write(text)
This function has no return value.
The **text** parameter is a **String**.
writeln(text)
This function has no return value.
The **text** parameter is a **String**.
getElementsByName(elementName)
This function returns an object that implements the **NodeList** interface.
The **elementName** parameter is a **String**.

Objects that implement the **HTMLElement** interface:

Objects that implement the **HTMLElement** interface have all properties and functions of the **Element** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLElement** interface:

id
This property is a **String**.
title
This property is a **String**.
lang
This property is a **String**.
dir
This property is a **String**.
className
This property is a **String**.

Objects that implement the **HTMLHtmlElement** interface:

Objects that implement the **HTMLHtmlElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLHtmlElement** interface:

version

This property is a **String**.

Objects that implement the **HTMLHeadElement** interface:

Objects that implement the **HTMLHeadElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLHeadElement** interface:

profile

This property is a **String**.

Objects that implement the **HTMLLinkElement** interface:

Objects that implement the **HTMLLinkElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLLinkElement** interface:

disabled

This property is a **Boolean**.

charset

This property is a **String**.

href

This property is a **String**.

hreflang

This property is a **String**.

media

This property is a **String**.

rel

This property is a **String**.

rev

This property is a **String**.

target

This property is a **String**.

type

This property is a **String**.

Objects that implement the **HTMLTitleElement** interface:

Objects that implement the **HTMLTitleElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLTitleElement** interface:

text

This property is a **String**.

Objects that implement the **HTMLMetaElement** interface:

Objects that implement the **HTMLMetaElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLMetaElement** interface:

content

This property is a **String**.

httpEquiv

This property is a **String**.

name
This property is a **String**.
scheme
This property is a **String**.

Objects that implement the **HTMLBaseElement** interface:
Objects that implement the **HTMLBaseElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.
Properties of objects that implement the **HTMLBaseElement** interface:
href
This property is a **String**.
target
This property is a **String**.

Objects that implement the **HTMLIsIndexElement** interface:
Objects that implement the **HTMLIsIndexElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.
Properties of objects that implement the **HTMLIsIndexElement** interface:
form
This read-only property is an object that implements the **HTMLFormElement** interface.
prompt
This property is a **String**.

Objects that implement the **HTMLStyleElement** interface:
Objects that implement the **HTMLStyleElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.
Properties of objects that implement the **HTMLStyleElement** interface:
disabled
This property is a **Boolean**.
media
This property is a **String**.
type
This property is a **String**.

Objects that implement the **HTMLBodyElement** interface:
Objects that implement the **HTMLBodyElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.
Properties of objects that implement the **HTMLBodyElement** interface:
aLink
This property is a **String**.
background
This property is a **String**.
bgColor
This property is a **String**.
link
This property is a **String**.
text
This property is a **String**.
vLink
This property is a **String**.

Objects that implement the **HTMLFormElement** interface:

Objects that implement the **HTMLFormElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLFormElement** interface:

elements

This read-only property is an object that implements the **HTMLCollection** interface.

length

This read-only property is a **Number**.

name

This property is a **String**.

acceptCharset

This property is a **String**.

action

This property is a **String**.

enctype

This property is a **String**.

method

This property is a **String**.

target

This property is a **String**.

Functions of objects that implement the **HTMLFormElement** interface:

submit()

This function has no return value.

reset()

This function has no return value.

Objects that implement the **HTMLSelectElement** interface:

Objects that implement the **HTMLSelectElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLSelectElement** interface:

type

This read-only property is a **String**.

selectedIndex

This property is a **Number**.

value

This property is a **String**.

length

This read-only property is a **Number**.

form

This read-only property is an object that implements the **HTMLFormElement** interface.

options

This read-only property is an object that implements the **HTMLCollection** interface.

disabled

This property is a **Boolean**.

multiple

This property is a **Boolean**.

name
This property is a **String**.
size
This property is a **Number**.
tabIndex
This property is a **Number**.

Functions of objects that implement the **HTMLSelectElement** interface:

add(element, before)
This function has no return value.
The **element** parameter is an object that implements the **HTMLElement** interface.
The **before** parameter is an object that implements the **HTMLElement** interface.
This function can raise an object that implements the **DOMException** interface.
remove(index)
This function has no return value.
The **index** parameter is a **Number**.
blur()
This function has no return value.
focus()
This function has no return value.

Objects that implement the **HTMLOptGroupElement** interface:

Objects that implement the **HTMLOptGroupElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLOptGroupElement** interface:

disabled
This property is a **Boolean**.
label
This property is a **String**.

Objects that implement the **HTMLOptionElement** interface:

Objects that implement the **HTMLOptionElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLOptionElement** interface:

form
This read-only property is an object that implements the **HTMLFormElement** interface.
defaultSelected
This property is a **Boolean**.
text
This read-only property is a **String**.
index
This read-only property is a **Number**.
disabled
This property is a **Boolean**.
label
This property is a **String**.
selected
This property is a **Boolean**.

value
This property is a **String**.

Objects that implement the **HTMLInputElement** interface:

Objects that implement the **HTMLInputElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLInputElement** interface:

defaultValue
This property is a **String**.

defaultChecked
This property is a **Boolean**.

form
This read-only property is an object that implements the **HTMLFormElement** interface.

accept
This property is a **String**.

accessKey
This property is a **String**.

align
This property is a **String**.

alt
This property is a **String**.

checked
This property is a **Boolean**.

disabled
This property is a **Boolean**.

maxLength
This property is a **Number**.

name
This property is a **String**.

readOnly
This property is a **Boolean**.

size
This property is a **String**.

src
This property is a **String**.

tabIndex
This property is a **Number**.

type
This property is a **String**.

useMap
This property is a **String**.

value
This property is a **String**.

Functions of objects that implement the **HTMLInputElement** interface:

blur()
This function has no return value.

focus()
This function has no return value.
select()
This function has no return value.
click()
This function has no return value.

Objects that implement the **HTMLTextAreaElement** interface:

Objects that implement the **HTMLTextAreaElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLTextAreaElement** interface:

defaultValue
This property is a **String**.
form
This read-only property is an object that implements the **HTMLFormElement** interface.
accessKey
This property is a **String**.
cols
This property is a **Number**.
disabled
This property is a **Boolean**.
name
This property is a **String**.
readOnly
This property is a **Boolean**.
rows
This property is a **Number**.
tabIndex
This property is a **Number**.
type
This read-only property is a **String**.
value
This property is a **String**.

Functions of objects that implement the **HTMLTextAreaElement** interface:

blur()
This function has no return value.
focus()
This function has no return value.
select()
This function has no return value.

Objects that implement the **HTMLButtonElement** interface:

Objects that implement the **HTMLButtonElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLButtonElement** interface:

form
This read-only property is an object that implements the **HTMLFormElement** interface.

accessKey
This property is a **String**.
disabled
This property is a **Boolean**.
name
This property is a **String**.
tabIndex
This property is a **Number**.
type
This read-only property is a **String**.
value
This property is a **String**.

Objects that implement the **HTMLLabelElement** interface:

Objects that implement the **HTMLLabelElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLLabelElement** interface:

form
This read-only property is an object that implements the **HTMLFormElement** interface.
accessKey
This property is a **String**.
htmlFor
This property is a **String**.

Objects that implement the **HTMLFieldSetElement** interface:

Objects that implement the **HTMLFieldSetElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLFieldSetElement** interface:

form
This read-only property is an object that implements the **HTMLFormElement** interface.

Objects that implement the **HTMLLegendElement** interface:

Objects that implement the **HTMLLegendElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLLegendElement** interface:

form
This read-only property is an object that implements the **HTMLFormElement** interface.
accessKey
This property is a **String**.
align
This property is a **String**.

Objects that implement the **HTMLUListElement** interface:

Objects that implement the **HTMLUListElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLUListElement** interface:

compact
This property is a **Boolean**.
type
This property is a **String**.

Objects that implement the **HTMLOListElement** interface:

Objects that implement the **HTMLOListElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLOListElement** interface:

compact

This property is a **Boolean**.

start

This property is a **Number**.

type

This property is a **String**.

Objects that implement the **HTMLDListElement** interface:

Objects that implement the **HTMLDListElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLDListElement** interface:

compact

This property is a **Boolean**.

Objects that implement the **HTMLDirectoryElement** interface:

Objects that implement the **HTMLDirectoryElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLDirectoryElement** interface:

compact

This property is a **Boolean**.

Objects that implement the **HTMLMenuElement** interface:

Objects that implement the **HTMLMenuElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLMenuElement** interface:

compact

This property is a **Boolean**.

Objects that implement the **HTMLLIElement** interface:

Objects that implement the **HTMLLIElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLLIElement** interface:

type

This property is a **String**.

value

This property is a **Number**.

Objects that implement the **HTMLDivElement** interface:

Objects that implement the **HTMLDivElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLDivElement** interface:

align

This property is a **String**.

Objects that implement the **HTMLParagraphElement** interface:

Objects that implement the **HTMLParagraphElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLParagraphElement** interface:

align

This property is a **String**.

Objects that implement the **HTMLHeadingElement** interface:

Objects that implement the **HTMLHeadingElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLHeadingElement** interface:

align

This property is a **String**.

Objects that implement the **HTMLQuoteElement** interface:

Objects that implement the **HTMLQuoteElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLQuoteElement** interface:

cite

This property is a **String**.

Objects that implement the **HTMLPreElement** interface:

Objects that implement the **HTMLPreElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLPreElement** interface:

width

This property is a **Number**.

Objects that implement the **HTMLBRElement** interface:

Objects that implement the **HTMLBRElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLBRElement** interface:

clear

This property is a **String**.

Objects that implement the **HTMLBaseFontElement** interface:

Objects that implement the **HTMLBaseFontElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLBaseFontElement** interface:

color

This property is a **String**.

face

This property is a **String**.

size

This property is a **String**.

Objects that implement the **HTMLFontElement** interface:

Objects that implement the **HTMLFontElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLFontElement** interface:

color

This property is a **String**.

face

This property is a **String**.

size
This property is a **String**.

Objects that implement the **HTMLHRElement** interface:
Objects that implement the **HTMLHRElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.
Properties of objects that implement the **HTMLHRElement** interface:

align
This property is a **String**.

noShade
This property is a **Boolean**.

size
This property is a **String**.

width
This property is a **String**.

Objects that implement the **HTMLModElement** interface:
Objects that implement the **HTMLModElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.
Properties of objects that implement the **HTMLModElement** interface:

cite
This property is a **String**.

dateTime
This property is a **String**.

Objects that implement the **HTMLAnchorElement** interface:
Objects that implement the **HTMLAnchorElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.
Properties of objects that implement the **HTMLAnchorElement** interface:

accessKey
This property is a **String**.

charset
This property is a **String**.

coords
This property is a **String**.

href
This property is a **String**.

hreflang
This property is a **String**.

name
This property is a **String**.

rel
This property is a **String**.

rev
This property is a **String**.

shape
This property is a **String**.

tabIndex
This property is a **Number**.

target
This property is a **String**.
type
This property is a **String**.
Functions of objects that implement the **HTMLAnchorElement** interface:
blur()
This function has no return value.
focus()
This function has no return value.
Objects that implement the **HTMLImageElement** interface:
Objects that implement the **HTMLImageElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.
Properties of objects that implement the **HTMLImageElement** interface:
name
This property is a **String**.
align
This property is a **String**.
alt
This property is a **String**.
border
This property is a **Number**.
height
This property is a **Number**.
hspace
This property is a **Number**.
isMap
This property is a **Boolean**.
longDesc
This property is a **String**.
src
This property is a **String**.
useMap
This property is a **String**.
vspace
This property is a **Number**.
width
This property is a **Number**.
Objects that implement the **HTMLObjectElement** interface:
Objects that implement the **HTMLObjectElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.
Properties of objects that implement the **HTMLObjectElement** interface:
code
This property is a **String**.
align
This property is a **String**.

archive
This property is a **String**.
border
This property is a **String**.
codeBase
This property is a **String**.
codeType
This property is a **String**.
data
This property is a **String**.
declare
This property is a **Boolean**.
height
This property is a **String**.
hspace
This property is a **String**.
name
This property is a **String**.
standby
This property is a **String**.
tabIndex
This property is a **Number**.
type
This property is a **String**.
useMap
This property is a **String**.
vspace
This property is a **String**.
width
This property is a **String**.
contentDocument
This read-only property is an object that implements the **Document** interface.

Objects that implement the **HTMLParamElement** interface:

Objects that implement the **HTMLParamElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLParamElement** interface:

name
This property is a **String**.
type
This property is a **String**.
value
This property is a **String**.
valueType
This property is a **String**.

Objects that implement the **HTMLAppletElement** interface:

Objects that implement the **HTMLAppletElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLAppletElement** interface:

align
: This property is a **String**.

alt
: This property is a **String**.

archive
: This property is a **String**.

code
: This property is a **String**.

codeBase
: This property is a **String**.

height
: This property is a **String**.

hspace
: This property is a **String**.

name
: This property is a **String**.

object
: This property is a **String**.

vspace
: This property is a **String**.

width
: This property is a **String**.

Objects that implement the **HTMLMapElement** interface:

Objects that implement the **HTMLMapElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLMapElement** interface:

areas
: This read-only property is an object that implements the **HTMLCollection** interface.

name
: This property is a **String**.

Objects that implement the **HTMLAreaElement** interface:

Objects that implement the **HTMLAreaElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLAreaElement** interface:

accessKey
: This property is a **String**.

alt
: This property is a **String**.

coords
: This property is a **String**.

href
: This property is a **String**.

noHref
This property is a **Boolean**.
shape
This property is a **String**.
tabIndex
This property is a **Number**.
target
This property is a **String**.

Objects that implement the **HTMLScriptElement** interface:

Objects that implement the **HTMLScriptElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLScriptElement** interface:

text
This property is a **String**.
htmlFor
This property is a **String**.
event
This property is a **String**.
charset
This property is a **String**.
defer
This property is a **Boolean**.
src
This property is a **String**.
type
This property is a **String**.

Objects that implement the **HTMLTableElement** interface:

Objects that implement the **HTMLTableElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLTableElement** interface:

caption
This property is an object that implements the **HTMLTableCaptionElement** interface.
tHead
This property is an object that implements the **HTMLTableSectionElement** interface.
tFoot
This property is an object that implements the **HTMLTableSectionElement** interface.
rows
This read-only property is an object that implements the **HTMLCollection** interface.
tBodies
This read-only property is an object that implements the **HTMLCollection** interface.
align
This property is a **String**.
bgColor
This property is a **String**.
border
This property is a **String**.

cellPadding
This property is a **String**.
cellSpacing
This property is a **String**.
frame
This property is a **String**.
rules
This property is a **String**.
summary
This property is a **String**.
width
This property is a **String**.

Functions of objects that implement the **HTMLTableElement** interface:

createTHead()
This function returns an object that implements the **HTMLElement** interface.
deleteTHead()
This function has no return value.
createTFoot()
This function returns an object that implements the **HTMLElement** interface.
deleteTFoot()
This function has no return value.
createCaption()
This function returns an object that implements the **HTMLElement** interface.
deleteCaption()
This function has no return value.
insertRow(index)
This function returns an object that implements the **HTMLElement** interface.
The **index** parameter is a **Number**.
This function can raise an object that implements the **DOMException** interface.
deleteRow(index)
This function has no return value.
The **index** parameter is a **Number**.
This function can raise an object that implements the **DOMException** interface.

Objects that implement the **HTMLTableCaptionElement** interface:

Objects that implement the **HTMLTableCaptionElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLTableCaptionElement** interface:

align
This property is a **String**.

Objects that implement the **HTMLTableColElement** interface:

Objects that implement the **HTMLTableColElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLTableColElement** interface:

align
This property is a **String**.

ch
This property is a **String**.
chOff
This property is a **String**.
span
This property is a **Number**.
vAlign
This property is a **String**.
width
This property is a **String**.

Objects that implement the **HTMLTableSectionElement** interface:

Objects that implement the **HTMLTableSectionElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLTableSectionElement** interface:

align
This property is a **String**.
ch
This property is a **String**.
chOff
This property is a **String**.
vAlign
This property is a **String**.
rows
This read-only property is an object that implements the **HTMLCollection** interface.

Functions of objects that implement the **HTMLTableSectionElement** interface:

insertRow(index)
This function returns an object that implements the **HTMLElement** interface.
The **index** parameter is a **Number**.
This function can raise an object that implements the **DOMException** interface.
deleteRow(index)
This function has no return value.
The **index** parameter is a **Number**.
This function can raise an object that implements the **DOMException** interface.

Objects that implement the **HTMLTableRowElement** interface:

Objects that implement the **HTMLTableRowElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLTableRowElement** interface:

rowIndex
This read-only property is a **Number**.
sectionRowIndex
This read-only property is a **Number**.
cells
This read-only property is an object that implements the **HTMLCollection** interface.
align
This property is a **String**.

bgColor
This property is a **String**.
ch
This property is a **String**.
chOff
This property is a **String**.
vAlign
This property is a **String**.

Functions of objects that implement the **HTMLTableRowElement** interface:

insertCell(index)
This function returns an object that implements the **HTMLElement** interface.
The **index** parameter is a **Number**.
This function can raise an object that implements the **DOMException** interface.
deleteCell(index)
This function has no return value.
The **index** parameter is a **Number**.
This function can raise an object that implements the **DOMException** interface.

Objects that implement the **HTMLTableCellElement** interface:

Objects that implement the **HTMLTableCellElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.

Properties of objects that implement the **HTMLTableCellElement** interface:

cellIndex
This read-only property is a **Number**.
abbr
This property is a **String**.
align
This property is a **String**.
axis
This property is a **String**.
bgColor
This property is a **String**.
ch
This property is a **String**.
chOff
This property is a **String**.
colSpan
This property is a **Number**.
headers
This property is a **String**.
height
This property is a **String**.
noWrap
This property is a **Boolean**.
rowSpan
This property is a **Number**.

scope
This property is a **String**.
vAlign
This property is a **String**.
width
This property is a **String**.

Objects that implement the **HTMLFrameSetElement** interface:
Objects that implement the **HTMLFrameSetElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.
Properties of objects that implement the **HTMLFrameSetElement** interface:
cols
This property is a **String**.
rows
This property is a **String**.

Objects that implement the **HTMLFrameElement** interface:
Objects that implement the **HTMLFrameElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.
Properties of objects that implement the **HTMLFrameElement** interface:
frameBorder
This property is a **String**.
longDesc
This property is a **String**.
marginHeight
This property is a **String**.
marginWidth
This property is a **String**.
name
This property is a **String**.
noResize
This property is a **Boolean**.
scrolling
This property is a **String**.
src
This property is a **String**.
contentDocument
This read-only property is an object that implements the **Document** interface.

Objects that implement the **HTMLIFrameElement** interface:
Objects that implement the **HTMLIFrameElement** interface have all properties and functions of the **HTMLElement** interface as well as the properties and functions defined below.
Properties of objects that implement the **HTMLIFrameElement** interface:
align
This property is a **String**.
frameBorder
This property is a **String**.
height
This property is a **String**.

longDesc
This property is a **String**.
marginHeight
This property is a **String**.
marginWidth
This property is a **String**.
name
This property is a **String**.
scrolling
This property is a **String**.
src
This property is a **String**.
width
This property is a **String**.
contentDocument
This read-only property is an object that implements the **Document** interface.

HTMLAnchorElement objects in `String` expressions are evaluated as `HTMLAnchorElement.href` [p.39] . For example:

```
alert(" The absolute URI is " + myAnchorElement);
```

is equivalent to

```
alert(" The absolute URI is " + myAnchorElement.href);
```

Appendix E: Acknowledgements

Many people contributed to this specification, including members of the DOM Working Group and the DOM Interest Group. We especially thank the following:

Lauren Wood (SoftQuad Software Inc., *chair*), Andrew Watson (Object Management Group), Andy Heninger (IBM), Arnaud Le Hors (W3C and IBM), Ben Chang (Oracle), Bill Smith (Sun), Bill Shea (Merrill Lynch), Bob Sutor (IBM), Chris Lovett (Microsoft), Chris Wilson (Microsoft), David Brownell (Sun), David Singer (IBM), Don Park (invited), Eric Vasilik (Microsoft), Gavin Nicol (INSO), Glenn Adams (ATSC), Ian Jacobs (W3C), James Clark (invited), James Davidson (Sun), Jared Sorensen (Novell), Joe Kesselman (IBM), Joe Lapp (webMethods), Joe Marini (Macromedia), Johnny Stenback (Netscape), Jonathan Marsh (Microsoft), Jonathan Robie (Texcel Research and Software AG), Kim Adamson-Sharpe (SoftQuad Software Inc.), Laurence Cable (Sun), Mark Davis (IBM), Mark Scardina (Oracle), Martin Dürst (W3C), Mick Goulish (Software AG), Mike Champion (Arbortext and Software AG), Miles Sabin (Cromwell Media), Patti Lutsky (Arbortext), Paul Grosso (Arbortext), Peter Sharpe (SoftQuad Software Inc.), Phil Karlton (Netscape), Philippe Le Hégaret (W3C, *W3C team contact*), Ramesh Lekshmynarayanan (Merrill Lynch), Ray Whitmer (iMall, Excite@Home and Netscape), Rich Rollman (Microsoft), Rick Gessner (Netscape), Scott Isaacs (Microsoft), Sharon Adler (INSO), Steve Byrne (JavaSoft), Tim Bray (invited), Tom Pixley (Netscape), Vidur Apparao (Netscape), Vinod Anupam (Lucent).

Thanks to all those who have helped to improve this specification by sending suggestions and corrections.

E.1: Production Systems

This specification was written in XML. The HTML, OMG IDL, Java and ECMA Script bindings were all produced automatically.

Thanks to Joe English, author of cost, which was used as the basis for producing DOM Level 1. Thanks also to Gavin Nicol, who wrote the scripts which run on top of cost. Arnaud Le Hors and Philippe Le Hégaret maintained the scripts.

For DOM Level 2, we used Xerces as the basis DOM implementation and wish to thank the authors. Philippe Le Hégaret and Arnaud Le Hors wrote the Java programs which are the DOM application.

Thanks also to Jan Kärrman, author of html2ps, which we use in creating the PostScript version of the specification.

Glossary

Editors:
Arnaud Le Hors, IBM
Lauren Wood, SoftQuad Software Inc.
Robert S. Sutor, IBM (for DOM Level 1)

Several of the following term definitions have been borrowed or modified from similar definitions in other W3C or standards documents. See the links within the definitions for more information.

convenience
A *convenience method* is an operation on an object that could be accomplished by a program consisting of more basic operations on the object. Convenience *methods* are usually provided to make the API easier and simpler to use or to allow specific programs to create more optimized implementations for common operations. A similar definition holds for a *convenience property*.

data model
A *data model* is a collection of descriptions of data structures and their contained fields, together with the operations or functions that manipulate them.

DOM Level 0
The term "*DOM Level 0*" refers to a mix (not formally specified) of HTML document functionalities offered by Netscape Navigator version 3.0 and Microsoft Internet Explorer version 3.0. In some cases, attributes or *methods* have been included for reasons of backward compatibility with "DOM Level 0".

HTML
The HyperText Markup Language (*HTML*) is a simple markup language used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of applications. [HTML4.0]

language binding
A programming *language binding* for an IDL specification is an implementation of the interfaces in the specification for the given language. For example, a Java language binding for the Document Object Model IDL specification would implement the concrete Java classes that provide the functionality exposed by the interfaces.

live
An object is *live* if any change to the underlying document structure is reflected in the object.

tokenized
The description given to various information items (for example, attribute values of various types, but not including the StringType CDATA) after having been processed by the XML processor. The process includes stripping leading and trailing white space, and replacing multiple space characters by one. See the definition of tokenized type.

References

For the latest version of any W3C specification please consult the list of W3C Technical Reports available at http://www.w3.org/TR.

G.1: Normative references

DOM Level 2 Core
W3C (World Wide Web Consortium) Document Object Model Level 2 Core Specification, November 2000. Available at http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113

ECMAScript
ECMA (European Computer Manufacturers Association) ECMAScript Language Specification. Available at http://www.ecma.ch/ecma1/STAND/ECMA-262.HTM

HTML4.0
W3C (World Wide Web Consortium) HTML 4.0 Specification, April 1998. Available at http://www.w3.org/TR/1998/REC-html40-19980424

XHTML 1.0
W3C (World Wide Web Consortium) XHTML 1.0: Extensible HyperText Markup Language, A Reformulation of HTML 4.0 in XML 1.0. Available at http://www.w3.org/TR/2000/REC-xhtml1-20000126

Java
Sun Microsystems Inc. The Java Language Specification, James Gosling, Bill Joy, and Guy Steele, September 1996. Available at http://java.sun.com/docs/books/jls

OMGIDL
OMG (Object Management Group) IDL (Interface Definition Language) defined in The Common Object Request Broker: Architecture and Specification, version 2.3.1, October 1999. Available from http://www.omg.org

DOM Level 2 Style Sheets and CSS
W3C (World Wide Web Consortium) Document Object Model Level 2 Style Sheets and CSS Specification, November 2000. Available at http://www.w3.org/TR/2000/REC-DOM-Level-2-Style-20001113

RFC2396
IETF (Internet Engineering Task Force) RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax, eds. T. Berners-Lee, R. Fielding, L. Masinter. August 1998. Available at http://www.ietf.org/rfc/rfc2396.txt

RFC2965
IETF (Internet Engineering Task Force) RFC 2965: HTTP State Management Mechanism, eds. D. Kristol and L. Montulli. October 2000. Available at http://www.ietf.org/rfc/rfc2965.txt

G.2: Informative references

DOM Level 1
W3C (World Wide Web Consortium) DOM Level 1 Specification, October 1998. Available at http://www.w3.org/TR/REC-DOM-Level-1

DOM Level 3 Abstract Schemas and Load and Save

W3C (World Wide Web Consortium) Document Object Model Level 3 Abstract Schemas and Load and Save Specification, June 2001. Available at http://www.w3.org/TR/DOM-Level-3-ASLS

RFC2616

IETF (Internet Engineering Task Force) RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1 , eds. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee. June 1999. Available at http://www.ietf.org/rfc/rfc2616.txt

Index

abbr

accept

acceptCharset

accessKey 27, 29, 31, 31, 32, 38, 44

action

add

align 27, 32, 35, 35, 35, 37, 40, 41, 43, 46, 49, 49, 50, 51, 53, 55

aLink

alt 27, 40, 43, 45

anchors

applets

archive 41, 43

areas

axis

background

bgColor 22, 46, 52, 53

blur 25, 28, 30, 39

body

border 40, 41, 46

caption

cellIndex

cellPadding

cells

cellSpacing

ch 49, 50, 52, 53

charset 19, 38, 45

checked

chOff 50, 50, 52, 53

cite 35, 38

className

clear

click

close

code 41, 43

codeBase 41, 43

codeType

color 36, 37

cols 29, 54

colSpan

compact 33, 33, 33, 34, 34

content

contentDocument 41, 55, 56

convenience 9, 13, 119

cookie

coords 38, 45

createCaption

createHTMLDocument

createTFoot

createTHead

data

data model 16, 119

dateTime

declare

defaultChecked

defaultSelected

defaultValue 28, 30

defer

deleteCaption

deleteCell

deleteRow 48, 50

deleteTFoot

deleteTHead
dir
disabled 19, 21, 24, 25, 26, 28, 30, 31
DOM Level 0 9, 10, 16, 119
DOM Level 1 9, 11, 40, 40, 40, 40, 121
DOM Level 2 Core 9, 121
DOM Level 2 Style Sheets and CSS 17, 19, 21, 121
DOM Level 3 Abstract Schemas and Load and Save 15, 122
domain

ECMAScript
elements
enctype
event

face 36, 37
focus 25, 29, 30, 39
form 21, 24, 26, 28, 30, 31, 32, 32, 32
forms
frame
frameBorder 55, 56

getElementsByName

headers
height 40, 42, 43, 53, 56
href 19, 20, 39, 45
hreflang 19, 39
hspace 40, 42, 43
HTML 9, 119
HTML4.0 9, 11, 26, 40, 48, 119, 121
HTMLAnchorElement
HTMLAppletElement
HTMLAreaElement
HTMLBaseElement
HTMLBaseFontElement
HTMLBodyElement
HTMLBRElement
HTMLButtonElement
HTMLCollection
HTMLDirectoryElement
HTMLDivElement
HTMLDListElement
HTMLDocument
HTMLDOMImplementation
HTMLElement
HTMLFieldSetElement
HTMLFontElement
htmlFor 32, 45
HTMLFormElement
HTMLFrameElement
HTMLFrameSetElement
HTMLHeadElement
HTMLHeadingElement
HTMLHRElement
HTMLHtmlElement
HTMLIFrameElement
HTMLImageElement
HTMLInputElement
HTMLIsIndexElement

HTMLLabelElement
HTMLLegendElement
HTMLLIElement
HTMLLinkElement
HTMLMapElement
HTMLMenuElement
HTMLMetaElement
HTMLModElement
HTMLObjectElement
HTMLOListElement
HTMLOptGroupElement
HTMLOptionElement
HTMLParagraphElement
HTMLParamElement
HTMLPreElement
HTMLQuoteElement
HTMLScriptElement
HTMLSelectElement
HTMLStyleElement
HTMLTableCaptionElement
HTMLTableCellElement
HTMLTableColElement
HTMLTableElement
HTMLTableRowElement
HTMLTableSectionElement
HTMLTextAreaElement
HTMLTitleElement
HTMLUListElement
httpEquiv

id
images
index
insertCell
insertRow 48, 51
isMap
item

Java

label 25, 26
lang
language binding 17, 119
length 12, 23, 24
link
links
live 12, 119
longDesc 40, 55, 56

marginHeight 55, 56
marginWidth 55, 56
maxLength
media 19, 21
method
multiple

name 20, 23, 24, 28, 30, 31, 39, 40, 42, 42, 44, 44, 55, 56
namedItem
noHref
noResize
noShade
noWrap

object
OMGIDL
open
options

profile
prompt

readOnly 28, 30
referrer
rel 19, 39
remove
reset
rev 19, 39
RFC2396 15, 13, 14, 19, 19, 20, 20, 22, 35, 38, 39, 40, 40, 41, 41, 43, 45, 45, 55, 55, 56, 56, 121
RFC2616 20, 23, 122
RFC2965 14, 121
rowIndex
rows 30, 47, 50, 54
rowSpan
rules

scheme
scope
scrolling 55, 56
sectionRowIndex
select 29, 30
selected
selectedIndex
shape 39, 45
size 24, 28, 36, 37, 37
span
src 28, 40, 45, 55, 56
standby
start
submit
summary

tabIndex 24, 28, 30, 31, 39, 42, 45
target 19, 21, 23, 39, 45
tBodies
text 20, 22, 26, 46
tFoot
tHead
title 15, 18
tokenized
type 19, 21, 24, 28, 30, 31, 33, 33, 34, 39, 42, 42, 46

URL
useMap 28, 40, 42

vAlign 50, 50, 52, 54
value 24, 26, 28, 30, 31, 34, 43
valueType
version
vLink
vspace 40, 42, 44

width 36, 37, 40, 42, 44, 47, 50, 54, 56
write
writeln

XHTML 1.0 9, 11, 121