[MS-DOM1]:
Internet Explorer Document Object Model (DOM) Level 1 Standards Support Document

---

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
03/17/2010	0.1	New	Released new document.
03/26/2010	1.0	None	Introduced no new technical or language changes.
05/26/2010	1.2	None	Introduced no new technical or language changes.
09/08/2010	1.3	Major	Significantly changed the technical content.
02/10/2011	2.0	No change	Introduced no new technical or language changes.
02/22/2012	3.0	Major	Significantly changed the technical content.
07/25/2012	3.1	Minor	Clarified the meaning of the technical content.

Table of Contents

1 Introduction

This document describes the level of support provided by Windows® Internet Explorer® for the *Document Object Model (DOM) Level 1 Specification Version 1.0* [DOM Level 1], W3C Recommendation 1 October, 1998. Internet Explorer displays webpages written in HTML.

The [DOM Level 1] specification may contain guidance for authors of HTML and XML documents, browser users and user agents (browser applications). Statements found in this document apply only to normative requirements in the specification targeted to user agents, not those targeted to authors.

1.1 Glossary

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

1.2 References

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

1.2.1 Normative References

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

[DOM Level 1] Worldwide Web Consortium (W3C), "Document Object Model (DOM) Level 1 Specification", Version 1.0, W3C Recommendation, October 1998, http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/

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

1.2.2 Informative References

None.

1.3 Microsoft Implementations

The following Microsoft products implement some portion of the [DOM Level 1] specification:

- Windows® Internet Explorer® 7
- Windows® Internet Explorer® 8
- Windows® Internet Explorer® 9
- Windows® Internet Explorer® 10

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

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

Throughout this document, the document mode appears first followed by the browser version in parentheses. Only those document modes and versions of Internet Explorer for which there is a variation note will be listed. If the document mode is not listed, conformance to the specification can be assumed.

Note "Standards mode" in Internet Explorer 7 and "IE7 mode" in Internet Explorer 8 refer to the same document mode. "IE7 mode" is the preferred way of referring to this document mode across all versions of the browser.

1.4 Standards Support Requirements

To conform to [DOM Level 1], a user agent must implement all required portions of the specification. Any optional portions that have been implemented must also be implemented as described by the specification. Normative language is usually used to define both required and optional portions. (For more information, see [RFC2119].)

The following table lists the sections of [DOM Level 1] and whether they are considered normative or informative.

Sections	Normative/Informative
1-2	Normative
Appendix A	Informative
Appendices B-E	Normative

1.5 Notation

The following notations are used in this document to differentiate between notes of clarification, variation from the specification, and extension points.

Notation	Explanation
C####	This identifies a clarification of ambiguity in the target specification. This includes imprecise statements, omitted information, discrepancies, and errata. This does not include data formatting clarifications.
V####	This identifies an intended point of variability in the target specification such as the use of MAY, SHOULD, or RECOMMENDED. (See [RFC2119].) This does not include extensibility points.
E####	Because the use of extensibility points (such as optional implementation-specific data) can impair interoperability, this profile identifies such points in the target specification.

2 Standards Support Statements

This section contains a full list of variations, clarifications, and extension points in the Microsoft implementation of [DOM Level 1].

- Section 2.1 includes only those variations that violate a MUST requirement in the target specification.
- Section 2.2 describes further variations from MAY and SHOULD requirements.
- Section 2.3 identifies variations in error handling.
- Section 2.4 identifies variations that impact security

2.1 Normative Variations

The following subsections detail the normative variations from MUST requirements in [DOM Level 1].

2.1.1 [DOM Level 1] Section 1.2, Fundamental Interfaces

V0001:

The specification states:

```
IDL Definition
exception DOMException {
  unsigned short   code;
};
```

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

The **DOMException** interface is not supported.

V0002:

The specification states:

```
While it is true that a Document object could fulfil this role, a Document object
can potentially be a heavyweight object, depending on the underlying
implementation. What is really needed for this is a very lightweight object.
DocumentFragment is such an object.
```

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

The **DocumentFragment** interface is derived from the **Document** interface. The **createDocumentFragment** method returns a full **Document** object.

V0003:

The specification states:

```
IDL Definition
interface DocumentFragment : Node {
```

```
};
```

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

The **DocumentFragment** interface is derived from the **Document** interface. The **Document** interface is derived from the **Node** interface.

V0004:

The specification states:

```
Method
createElement
Creates an element of the type specified. Note that the instance returned
implements the Element interface, so attributes can be specified directly on the
returned object.

Parameters
tagName  The name of the element type to instantiate. For XML, this is case-sensitive.
For HTML, the tagName parameter may be provided in any case, but it must be mapped
to the canonical uppercase form by the DOM implementation.

Return Value
A new Element object.

Exceptions
DOMException
INVALID_CHARACTER_ERR: Raised if the specified name contains an invalid character.
```

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

The following variation apply:

- The **createElement** method is overloaded with one that takes no parameters. When no parameters are given, this method returns an element with a **tagName** of null.
- The **createElement** method accepts full element declaration strings that contain otherwise invalid characters for the **tagName** parameter. A parameter string such as `"<div id='div1'>"` would return a **div** element with an **id** of `div1`. An INVALID_CHARACTER_ERR exception is not raised in this case.

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

- When an element that contains an XMLNS declaration (such as `<html XMLNS:mns='http://www.contoso.com'>`) is specified for the **tagName** parameter, the value of the **tagUrn** property for the new element is set to the specified URI.

V0005:

The specification states:

```
interface Document : Node

Method
createCDATASection
Creates a CDATASection node whose value is the specified string.
```

```
Parameters
data
The data for the CDATASection contents.

Return Value
The new CDATASection object.

Exceptions
DOMException  NOT_SUPPORTED_ERR: Raised if this document is an HTML document.
```

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

The **createCDATASection** method of the **Document** interface is not supported.

V0006:

The specification states:

```
interface Document : Node

Method
createProcessingInstruction
Creates a ProcessingInstruction node given the specified name and data strings.

Parameters
target
The target part of the processing instruction. data  The data for the node.

Return Value
The new ProcessingInstruction object.

Exceptions
DOMException  INVALID_CHARACTER_ERR: Raised if an invalid character is specified.
NOT_SUPPORTED_ERR: Raised if this document is an HTML document.
```

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

The **createProcessingInstruction** method of the **Document** interface is not supported.

V0008:

The specification states:

```
IDL Definition
interface Node {
  // NodeType
  const unsigned short      ELEMENT_NODE       = 1;
  const unsigned short      ATTRIBUTE_NODE     = 2;
  const unsigned short      TEXT_NODE          = 3;
  const unsigned short      CDATA_SECTION_NODE = 4;
  const unsigned short      ENTITY_REFERENCE_NODE = 5;
  const unsigned short      ENTITY_NODE        = 6;
  const unsigned short      PROCESSING_INSTRUCTION_NODE = 7;
  const unsigned short      COMMENT_NODE       = 8;
  const unsigned short      DOCUMENT_NODE      = 9;
```

```
    const unsigned short      DOCUMENT_TYPE_NODE = 10;
    const unsigned short      DOCUMENT_FRAGMENT_NODE = 11;
    const unsigned short      NOTATION_NODE      = 12;

    readonly attribute  DOMString           nodeName;
             attribute  DOMString           nodeValue;
                                            // raises(DOMException) on setting
                                            // raises(DOMException) on retrieval
    readonly attribute  unsigned short      nodeType;
    readonly attribute  Node                parentNode;
    readonly attribute  NodeList            childNodes;
    readonly attribute  Node                firstChild;
    readonly attribute  Node                lastChild;
    readonly attribute  Node                previousSibling;
    readonly attribute  Node                nextSibling;
    readonly attribute  NamedNodeMap        attributes;
    readonly attribute  Document            ownerDocument;
    Node                      insertBefore(in Node newChild,
                                           in Node refChild)
                                           raises(DOMException);
    Node                      replaceChild(in Node newChild,
                                           in Node oldChild)
                                           raises(DOMException);
    Node                      removeChild(in Node oldChild)
                                         raises(DOMException);
    Node                      appendChild(in Node newChild)
                                         raises(DOMException);
    boolean                   hasChildNodes();
    Node                      cloneNode(in boolean deep);
  };
```

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

The following constants are not accessible by name:

- ELEMENT_NODE
- ATTRIBUTE_NODE
- TEXT_NODE
- CDATA_SECTION_NODE
- ENTITY_REFERENCE_NODE
- ENTITY_NODE
- PROCESSING_INSTRUCTION_NODE
- COMMENT_NODE
- DOCUMENT_NODE
- DOCUMENT_TYPE_NODE
- DOCUMENT_FRAGMENT_NODE
- NOTATION_NODE

V0009:

The specification states:

```
Definition group
NodeType
An integer indicating which type of node this is.

Defined Constants
ELEMENT_NODE The node is a Element.
ATTRIBUTE_NODE The node is an Attr.
TEXT_NODE The node is a Text node.
CDATA_SECTION_NODE The node is a CDATASection.
ENTITY_REFERENCE_NODE The node is an EntityReference.
ENTITY_NODE The node is an Entity.
PROCESSING_INSTRUCTION_NODE The node is a ProcessingInstruction.
COMMENT_NODE The node is a Comment.
DOCUMENT_NODE The node is a Document.
DOCUMENT_TYPE_NODE The node is a DocumentType.
DOCUMENT_FRAGMENT_NODE The node is a DocumentFragment.
NOTATION_NODE The node is a Notation.

The values of nodeName, nodeValue, and attributes vary according to the node type
as follows:
                     nodeName, nodeValue, attributes
Element              tagName, null, NamedNodeMap
Attr                 name of attribute, value of attribute, null
Text                 #text, content of the text node, null
CDATASection         #cdata-section, content of the CDATA Section, null
EntityReference      name of entity referenced, null, null
Entity               entity name, null, null
ProcessingInstruction target, entire content excluding the target, null
Comment              #comment, content of the comment, null
Document             #document, null, null
DocumentType         document type name, null, null
DocumentFragment     #document-fragment, null, null
Notation             notation name, null, null
```

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

The following constants are not supported:

- ELEMENT_NODE
- ATTRIBUTE_NODE
- TEXT_NODE
- CDATA_SECTION_NODE
- ENTITY_REFERENCE_NODE
- ENTITY_NODE
- PROCESSING_INSTRUCTION_NODE
- COMMENT_NODE
- DOCUMENT_NODE

- DOCUMENT_TYPE_NODE
- DOCUMENT_FRAGMENT_NODE
- NOTATION_NODE

A document type declaration is treated as a **Comment** object instead of a **DocumentType** object

V0010:

The specification states:

```
Attribute
nodeName
The name of this node, depending on its type; see the table above.
```

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

The **nodeName** property returns uppercase values except for elements with names that resemble namespaces (such as <test:elementName>) when a proprietary namespace has been declared. In this case, the **nodeName** property drops the element prefixes and does not return uppercase values.

V0011:

The specification states:

```
Attribute
parentNode
The parent of this node. All nodes, except Document, DocumentFragment, and Attr may
have a parent. However, if a node has just been created and not yet added to the
tree, or if it has been removed from the tree, this is null.
```

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

When an element without a parent has child nodes, an **HTMLDocument** object is created and set as the parent of that element.

V0012:

The specification states:

```
Attribute
childNodes
A NodeList that contains all children of this node. If there are no children, this
is a NodeList containing no nodes. The content of the returned NodeList is "live"
in the sense that, for instance, changes to the children of the node object that it
was created from are immediately reflected in the nodes returned by the NodeList
accessors; it is not a static snapshot of the content of the node. This is true for
every NodeList, including the ones returned by the getElementsByTagName method.
```

All Document Modes (Internet Explorer 8)

Splitting multiple text nodes under an element using the **splitText** method can cause the **childNodes** collection update to be delayed. Other tree modifications cause the **childNodes** collection to synchronize again.

V0013:

The specification states:

```
Method
insertBefore Inserts the node newChild before the existing child node refChild. If
refChild is null, insert newChild at the end of the list of children. If newChild
is a DocumentFragment object, all of its children are inserted, in the same order,
before refChild. If the newChild is already in the tree, it is first removed.

Parameters
newChild  The node to insert.
refChild  The reference node, i.e., the node before which the new node must be
inserted.

Return Value
The node being inserted.

Exceptions
DOMException HIERARCHY_REQUEST_ERR: Raised if this node is of a type that does not
allow children of the type of the newChild node, or if the node to insert is one of
this node's ancestors.
WRONG_DOCUMENT_ERR: Raised if newChild was created from a different document than
the one that created this node.
NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
NOT_FOUND_ERR: Raised if refChild is not a child of this node.
```

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

The conditions to trigger the exceptions HIERARCHY_REQUEST_ERR and WRONG_DOCUMENT_ERR both result in an HRESULT `0x80070057`, which creates a JavaScript Error message "Invalid argument."

The following elements cause an exception when trying to dynamically insert or append new nodes:

- **APPLET**
- **AREA**
- **BASE**
- **BGSOUND**
- **BR**
- **COL**
- **COMMENT**
- **EMBED**
- **FRAME**
- **HR**

- **IFRAME**
- **IMG**
- **INPUT**
- **ISINDEX**
- **LINK**
- **META**
- **NEXTID**
- **NOEMBED**
- **NOFRAMES**
- **NOSCRIPT**
- **OBJECT**
- **PARAM**
- **SCRIPT**
- **STYLE**
- **WBR**

V0014:

The specification states:

```
Method
replaceChild Replaces the child node oldChild with newChild in the list of
children, and returns the oldChild node. If the newChild is already in the tree, it
is first removed.

Parameters
newChild  The new node to put in the child list.
oldChild  The node being replaced in the list.

Return Value
The node replaced.

Exceptions DOMException  HIERARCHY_REQUEST_ERR: Raised if this node is of a type
that does not allow children of the type of the newChild node, or it the node to
put in is one of this node's ancestors.
WRONG_DOCUMENT_ERR: Raised if newChild was created from a different document than
the one that created this node.
NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
NOT_FOUND_ERR: Raised if oldChild is not a child of this node.
```

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

The conditions to trigger HIERARCHY_REQUEST_ERR, WRONG_DOCUMENT_ERR, and NOT_FOUND_ERR all result in an HRESULT 0x80070057, which creates a JavaScript Error message "Invalid argument."

V0015:

The specification states:

```
Method
removeChild Removes the child node indicated by oldChild from the list of children,
and returns it.

Parameters
oldChild  The node being removed.

Return Value
The node removed.

Exceptions
DOMException  NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
NOT_FOUND_ERR: Raised if oldChild is not a child of this node.
```

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

If the child node indicated by the **oldChild** parameter is not from the list of child nodes, the **removeChild** method of the **Node** interface raises a **JSError** exception with an error message of "Invalid argument" and an HRESULT of 0x80070057.

V0016:

The specification states:

```
Method
appendChild Adds the node newChild to the end of the list of children of this node.
If the newChild is already in the tree, it is first removed.

Parameters
newChild  The node to add. If it is a DocumentFragment object, the entire contents
of the document fragment are moved into the child list of this node

Return Value
The node added.

Exceptions
DOMException  HIERARCHY_REQUEST_ERR: Raised if this node is of a type that does not
allow children of the type of the newChild node, or if the node to append is one of
this node's ancestors.
WRONG_DOCUMENT_ERR: Raised if newChild was created from a different document than
the one that created this node.
NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
```

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

The conditions to trigger HIERARCHY_REQUEST_ERR and WRONG_DOCUMENT_ERR both result in an HRESULT 0x80070057, which creates a JavaScript Error message "Invalid argument."

The following elements cause an exception when trying to dynamically insert or append new nodes:

- **APPLET**
- **AREA**
- **BASE**
- **BGSOUND**
- **BR**
- **COL**
- **COMMENT**
- **EMBED**
- **FRAME**
- **HR**
- **IFRAME**
- **IMG**
- **INPUT**
- **ISINDEX**
- **LINK**
- **META**
- **NEXTID**
- **NOEMBED**
- **NOFRAMES**
- **NOSCRIPT**
- **OBJECT**
- **PARAM**
- **SCRIPT**
- **STYLE**
- **WBR**

V0017:

The specification states:

```
Method
setNamedItem Adds a node using its nodeName attribute. As the nodeName attribute is
used to derive the name which the node must be stored under, multiple nodes of
```

```
certain types (those that have a "special" string value) cannot be stored as the
names would clash. This is seen as preferable to allowing nodes to be aliased.

Parameters
arg  A node to store in a named node map. The node will later be accessible using
the value of the nodeName attribute of the node. If a node with that name is
already present in the map, it is replaced by the new one.

Return Value
If the new Node replaces an existing node with the same name the previously
existing Node is returned, otherwise null is returned.

Exceptions
DOMException  WRONG_DOCUMENT_ERR: Raised if arg was created from a different
document than the one that created the NamedNodeMap.
NO_MODIFICATION_ALLOWED_ERR: Raised if this NamedNodeMap is readonly.
INUSE_ATTRIBUTE_ERR: Raised if arg is an Attr that is already an attribute of
another Element object. The DOM user must explicitly clone Attr nodes to re-use
them in other elements.
```

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

The following variations apply:

- The WRONG_DOCUMENT_ERR exception is not raised if the **arg** parameter was created from a document other than the one that created **NamedNodeMap**.
- Instead of the INUSE_ATTRIBUTE_ERR exception, if **arg** is an **Attr** that is already an attribute of another **Element** object, the **setNamedItem** method of the **Node** interface raises a JSError exception with an error message of "Invalid argument" and an HRESULT of `0x80070057`.

V0018:

The specification states:

```
Method
item Returns the indexth item in the map. If index is greater than or equal to the
number of nodes in the map, this returns null.

Parameters
index  Index into the map.

Return Value
The node at the indexth position in the NamedNodeMap, or null if that is not a
valid index.

This method raises no exceptions.
```

Quirks Mode and IE7 Mode (All Versions)

Instead of returning null when the **index** parameter is greater than the number of nodes in the map, the **item** method of the **Node** interface throws a JSError exception with an error message of `"Invalid argument"` and an HRESULT of `0x80070057`.

V0019:

The specification states:

```
Method
substringData Extracts a range of data from the node.

Parameters
offset  Start offset of substring to extract. count  The number of characters to
extract.

Return Value
The specified substring. If the sum of offset and count exceeds the length, then
all characters to the end of the data are returned.

Exceptions
DOMException  INDEX_SIZE_ERR: Raised if the specified offset is negative or greater
than the number of characters in data, or if the specified count is negative.
DOMSTRING_SIZE_ERR: Raised if the specified range of text does not fit into a DOMString.
```

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

The following variations apply:

- No exception is raised if the offset is greater than the number of 16-bit units in the data.
- Instead of the **INDEX_SIZE_ERR** DOMException, if the specified offset is negative or greater than the number of characters in the data or the specified count is negative, the **substringData** method of the **Node** interface raises a JScript Error exception with an error message of "Invalid argument" and an HRESULT of `0x80070057`.

V0020:

The specification states:

```
Attribute
specified
If this attribute was explicitly given a value in the original document, this is
true; otherwise, it is false. Note that the implementation is in charge of this
attribute, not the user. If the user changes the value of the attribute (even if it
ends up having the same value as the default value) then the specified flag is
automatically flipped to true. To re-specify the attribute as the default value
from the DTD, the user must delete the attribute. The implementation will then make
a new attribute available with specified set to false and the default value (if one
exists). In summary:
- If the attribute has an assigned value in the document then specified is true,
and the value is the assigned value.
- If the attribute has no assigned value in the document and has a default value in
the DTD, then specified is false, and the value is the default value in the DTD.
```

Quirks Mode and IE7 Mode (All Versions)

The value of the **specified** attribute is not automatically flipped when the associated attribute is changed.

V0021:

The specification states:

```
Elements may have attributes associated with them; since the Element interface
inherits from Node, the generic Node interface method getAttributes may be used to
retrieve the set of all attributes for an element. There are methods on the Element
interface to retrieve either an Attr object by name or an attribute value by name.
In XML, where an attribute value may contain entity references, an Attr object
should be retrieved to examine the possibly fairly complex sub-tree representing
the attribute value. On the other hand, in HTML, where all attributes have simple
string values, methods to directly access an attribute value can safely be used as
a convenience.
```

All Document Modes (All Versions)

The **getAttributes** method is not supported.

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

Attribute subtrees are not supported.

IE9 Mode and IE10 Mode (All Versions)

For any given attribute, only a single, immutable text node is supported as the subtree.

V0022:

The specification states:

```
Attribute
tagName
The name of the element. For example, in: <elementExample id="demo"> ...
</elementExample> ,tagName has the value "elementExample". Note that this is case-
preserving in XML, as are all of the operations of the DOM. The HTML DOM returns
the tagName of an HTML element in the canonical uppercase form, regardless of the
case in the source HTML document.
```

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

The **tagName** property returns uppercase values except for elements with names that resemble namespaces (such as <test:elementName>) when a proprietary namespace has been declared. In this case, the **tagName** property drops the element prefixes and does not return uppercase values.

V0023:

The specification states:

```
Method
getAttribute Retrieves an attribute value by name.

Parameters
name  The name of the attribute to retrieve.

Return Value The Attr value as a string, or the empty string if that attribute does
not have a specified or default value.

This method raises no exceptions.
```

Quirks Mode and IE7 Mode (All Versions)

The **getAttribute** method supports a second parameter called **iFlags**. The **iFlags** parameter controls case sensitivity and object interpolation. By default, **iFlags** is set to `0`, which indicates that the property search done by the **getAttribute** method is not case-sensitive and returns an interpolated value if the property is found.

V0024:

The specification states:

```
Method
setAttribute Adds a new attribute. If an attribute with that name is already
present in the element, its value is changed to be that of the value parameter.
This value is a simple string, it is not parsed as it is being set. So any markup
(such as syntax to be recognized as an entity reference) is treated as literal
text, and needs to be appropriately escaped by the implementation when it is
written out. In order to assign an attribute value that contains entity references,
the user must create an Attr node plus any Text and EntityReference nodes, build
the appropriate subtree, and use setAttributeNode to assign it as the value of an attribute.

Parameters
name  The name of the attribute to create or alter. value  Value to set in string form.

Exceptions DOMException  INVALID_CHARACTER_ERR: Raised if the specified name
contains an invalid character.
NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.

This method returns nothing.
```

Quirks Mode and IE7 Mode (All Versions)

The following variations apply:

- The **setAttribute** method assigns attributes in a case-sensitive manner (expected case-insensitive for HTML).
- The second parameter of the **setAttribute** method accepts only strings, not objects. An optional third parameter controls case sensitivity.
- Attributes that apply a boolean initial state to the associated DOM properties (for example, **value** and **checked**) are incorrectly associated with their live property rather than their default property. For example, the **setAttribute** method ('checked', 'checked') toggles the DOM **checked** property (the live view of a check box) rather than the **defaultChecked** property (initial value).
- The HTML **style** attribute and attributes that are event handlers do not apply their conditions when used with **setAttribute**.
- The **setAttribute** method requires DOM property names to apply effects for certain attribute names, such as **className** (instead of 'class'), **htmlFor** (instead of 'for'), and **httpEquiv** (instead of 'http-equiv').

V0025:

The specification states:

```
Method
removeAttribute Removes an attribute by name. If the removed attribute has a
default value it is immediately replaced.

Parameters
name  The name of the attribute to remove.

Exceptions DOMException  NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.

This method returns nothing.
```

Quirks Mode and IE7 Mode (All Versions)

The following variations apply:

- The **removeAttribute** method supports an additional parameter called **iCaseSensitive**. The **iCaseSensitive** parameter specifies whether to use a case-sensitive search to find the attribute. Removal of event handler attributes (such as **onClick**) or the style attribute does not cause the actual event handler to be removed, or the inline style to be removed.
- Default attributes are not re-created after the attribute is removed.

All Document Modes (All Versions)

- The following variations apply:
- The **removeAttribute** method of the **Element** interface exists on **CSSStyleDeclaration** objects that are used for inline styles, runtime styles, and style rules.
- The **removeAttribute** method returns a Boolean value that indicates whether the operation has succeeded or failed. The **removeAttribute** method is also available on the following objects: **element.currentStyle**, **element.runtimeStyle**, **element.style**, and **stylesheet.style**.

V0026:

The specification states:

```
Method
removeAttributeNode Removes the specified attribute.
Parameters
oldAttr  The Attr node to remove from the attribute list. If the removed Attr has a
default value it is immediately replaced.

Return Value The Attr node that was removed.

Exceptions DOMException  NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
NOT_FOUND_ERR: Raised if oldAttr is not an attribute of the element.
```

Quirks Mode and IE7 Mode (All Versions)

Default attributes are not re-created after an attribute is removed with the **removeAttributeNode** method. Removal of event handler attributes (such as **onclick**) or the **style** attribute does not cause the actual event handler or the inline style to be removed.

V0027:

The specification states:

```
Method
getElementsByTagName
Returns a NodeList of all descendant elements with a given tag name, in the order
in which they would be encountered in a preorder traversal of the Element tree.

Parameters
name  The name of the tag to match on. The special value "*" matches all tags.

Return Value
A list of matching Element nodes.

This method raises no exceptions.
```

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

The following variations apply:

- When called on an **object** element, where the value "*" is passed in as the value of the **name** parameter, the **getElementsByTagName** method returns an empty **NodeList**.
- When called on an **OBJECT** element, where `param` is passed in as the value of the **name** parameter, the **getElementsByTagName** method returns a **NodeList** that contains all the **PARAM** elements in the document.

V0028:

The specification states:

```
Method
splitText
Breaks this Text node into two Text nodes at the specified offset, keeping both in
the tree as siblings. This node then only contains all the content up to the offset
point. And a new Text node, which is inserted as the next sibling of this node,
contains all the content at and after the offset point.

Parameters
offset  The offset at which to split, starting from 0.

Return Value
The new Text node.

Exceptions
DOMException  INDEX_SIZE_ERR: Raised if the specified offset is negative or greater
than the number of characters in data.
NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
```

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

The **offset** parameter is treated as though it is optional. If no offset is provided, then a default offset of 0 is used.

All Document Modes (Internet Explorer 8)

The **childNodes** objects are kept in a cache and are invalidated whenever there is a modification to the markup. Calling the **splitText** method of the **Text** interface does not trigger a markup modification. The **childNodes** collection does not show changes made by **splitText** until the markup is modified, for example, by changing the text of a **DIV** element anywhere on the page.

2.1.2 [DOM Level 1] Section 2.4, Objects related to HTML documents

V0029:

The specification states:

```
Attribute
domain
The domain name of the server that served the document, or a null string if the
server cannot be identified by a domain name.
```

All Document Modes (All Versions)

An empty string is assigned to the value of the **domain** attribute when the local machine is the server.

2.1.3 [DOM Level 1] Section 2.5.5, Object definitions

V0030:

The specification states:

```
interface HTMLSelectElement : HTMLElement

Method
add Add a new element to the collection of OPTION elements for this SELECT.

Parameters
element  The element to add.
before  The element to insert before, or NULL for the head of the list.

This method returns nothing.

This method raises no exceptions.
```

Quirks Mode and IE7 Mode (All Versions)

The **add** method of the **HTMLSelectElement** interface is not supported.

IE8 Mode, IE9 Mode, and IE10 Mode (All Versions)

The **add** method of the **HTMLSelectElement** appends the element to the bottom of the list when the **before** parameter is null.

V0031:

The specification states:

```
IDL Definition
interface HTMLOptionElement : HTMLElement {
```

```
    readonly attribute  HTMLFormElement  form;
             attribute  boolean          defaultSelected;
    readonly attribute  DOMString        text;
             attribute  long             index;
             attribute  boolean          disabled;
             attribute  DOMString        label;
    readonly attribute  boolean          selected;
             attribute  DOMString        value;
  };
```

All Document Modes (All Versions)

The **text** attribute of the **HTMLOptionElement** interface is not read-only.

V0032:

The specification states:

```
Attribute
align
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.
```

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

When specified in HTML, the **align** attribute of the **INPUT** element is ignored and the DOM **align** attribute returns an empty string.

V0033:

The specification states:

```
Attribute
useMap
Use client-side image map. See the usemap attribute definition in HTML 4.0.
```

All Document Modes (All Versions)

The value of the **useMap** attribute is set to the value of the **usemap** attribute in HTML. This string is not constrained to the name of the associated **MAP** element.

V0034:

The specification states:

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

All Document Modes (All Versions)

The **value** attribute of an **LI** element cannot be used to reset the sequence number in an **OL** element.

V0035:

The specification states:

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

All Document Modes (All Versions)

The **width** attribute of a **PRE** element is treated as a string.

V0036:

The specification states:

```
interface HTMLBaseFontElement : HTMLElement
interface HTMLFontElement : HTMLElement

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

All Document Modes (All Versions)

If the value of the **color** attribute is specified as a color string (such as `black`) in HTML, the value is converted and stored as a hexadecimal RGB color value in the DOM **color** attribute.

V0037:

The specification states:

```
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;};
```

All Document Modes (All Versions)

The following variations apply:

- The value of the **hspace** attribute is of type **number**.
- The value of the **vspace** attribute is of type **number**.
- The value of the **object** attribute is of type **object**.

V0038:

The specification states:

```
Attribute
object
Serialized applet file. See the object attribute definition in HTML 4.0. This
attribute is deprecated in HTML 4.0.
```

All Document Modes (All Versions)

The **object** attribute of the **APPLET** element is not supported.

V0039:

The specification states:

```
interface HTMLScriptElement : HTMLElement

Attribute
src
URI designating an external script. See the src attribute definition in HTML 4.0.
```

IE8 Mode, IE9 Mode, and IE10 Mode (All Versions)

When the value of the **src** attribute for the **SCRIPT** element is specified in HTML as a relative URI, the value is converted and stored as an absolute URI in the **src** attribute of the **HTMLScriptElement**.

V0041:

The specification states:

```
Attribute
bgColor
Cell background color. See the bgcolor attribute definition in HTML 4.0. This
attribute is deprecated in HTML 4.0.
```

All Document Modes (All Versions)

If the value of the **bgcolor** attribute is specified as a color string (such as `black`) in HTML, the value is converted and stored as a hexadecimal RGB color value in the DOM **bgColor** attribute.

V0042:

The specification states:

```
Attribute
align
```

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

All Document Modes (All Versions)

The `char` and `justify` values are not supported for the **align** attribute of a **TR** element.

V0043:

The specification states:

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

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

The **ch** attribute on the **TR** element is not supported. The **ch** attribute can be written to and read from the DOM, but it does not change the alignment of text within a cell.

V0044:

The specification states:

```
Attribute
chOff
Offset of alignment character. See the charoff attribute definition in HTML 4.0.
```

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

The **chOff** attribute on the **TR** element is not supported. The **ch** attribute can be written to and read from the DOM, but it does not change the offset of the alignment.

V0045:

The specification states:

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

Quirks Mode and IE7 Mode (All Versions)

If the **rowspan** attribute specified on a cell is greater than the number of rows in a table, the value of this attribute is modified to become equal to the number of rows in the table.

V0047:

The specification states:

```
Attribute
marginHeight
Frame margin height, in pixels. See the marginheight attribute definition in HTML 4.0.
```

All Document Modes (All Versions)

The **marginHeight** attribute is an expando attribute. If the **marginheight** attribute and value are declared in markup, the value is stored and returned by the **marginHeight** attribute. If no value is specified, **marginHeight** is undefined.

V0048:

The specification states:

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

All Document Modes (All Versions)

marginWidth is an expando attribute. If the **marginwidth** attribute and value are declared in markup, the value is stored and returned by the **marginWidth** attribute. If no value is specified, **marginWidth** is undefined.

V0049:

The specification states:

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

All Document Modes (All Versions)

name is an expando attribute. If the **name** attribute and value are declared in markup, the value is stored and returned by the **name** attribute. If no value is specified, **name** returns an empty string in the DOM.

V0050:

The specification states:

```
Attribute
scrolling
Specify whether or not the frame should have scrollbars. See the
scrolling attribute definition in HTML 4.0.
```

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

The **scrolling** attribute is an expando attribute. If the **scrolling** attribute and value are declared in markup, the value is stored and returned by the **scrolling** attribute. If no value is specified, **scrolling** is undefined in the DOM.

V0051:

The specification states:

```
Attribute
longDesc
URI designating a long description of this image or frame. See the longdesc
```

```
attribute definition in HTML 4.0.
```

IE8 Mode, IE9 Mode, and IE10 Mode (All Versions)

When the value of the **longdesc** attribute for the **IFRAME** element is specified in HTML as a relative URI, the value is converted and stored as an absolute URI in the **longDesc** attribute of the **HTMLIFrameElement**. If not already present, a slash is appended to the end of the URI when a file name is not specified.

V0052:

The specification states:

```
Attribute
src
A URI designating the initial frame contents. See the src attribute definition in HTML 4.0.
```

IE8 Mode, IE9 Mode, and IE10 Mode (All Versions)

The **src** attribute is treated as an absolute URI. If no trailing slash is provided, the slash is included when the attribute is fixed up to become an absolute URI before being applied to the DOM property.

2.2 Clarifications

The following subsections identify clarifications to recommendations made by [DOM Level 1].

2.2.1 [DOM Level 1] Section 1.2, Fundamental Interfaces

C0001:

The specification states:

```
Method
createDocumentFragment Creates an empty DocumentFragment object.

Return Value
A new DocumentFragment.

This method has no parameters.

This method raises no exceptions.
```

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

The **createDocumentFragment** method returns an object that is derived from the **document** interface.

C0002:

The specification states:

```
Method
createTextNode Creates a Text node given the specified string.
```

```
Parameters
data  The data for the node.

Return Value
The new Text object.

This method raises no exceptions.
```

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

The **data** parameter is treated as optional. The **createTextNode** method creates a text node even when no parameter is provided.

C0003:

The specification states:

```
Method
createComment Creates a Comment node given the specified string.

Parameters
data  The data for the node.

Return Value
The new Comment object.

This method raises no exceptions.
```

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

The **data** parameter is treated as optional. The **createComment** method creates a COMMENT node even when no parameter is provided.

C0004:

The specification states:

```
Method
getAttributeNode Retrieves an Attr node by name.

Parameters
name  The name of the attribute to retrieve.

Return Value
The Attr node with the specified attribute name or null if there is no such attribute.

This method raises no exceptions.
```

Quirks Mode and IE7 Mode (All Versions)

The **nodeValue** attribute of the object returned from the **getAttributeNode** method returns an empty string if an attribute has not been assigned a value in HTML.

IE8 Mode (All Versions)

The **getAttributeNode** method returns null if an attribute is specified but has not been assigned a value in HTML.

C0005:

The specification states:

```
Method
normalize
Puts all Text nodes in the full depth of the sub-tree underneath this
Element into a "normal" form where only markup (e.g., tags, comments, processing
instructions, CDATA sections, and entity references) separates Text nodes, i.e.,
there are no adjacent Text nodes. This can be used to ensure that the DOM view of a
document is the same as if it were saved and re-loaded, and is useful when
operations (such as XPointer lookups) that depend on a particular document tree
structure are to be used.

This method has no parameters.

This method returns nothing.

This method raises no exceptions.
```

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

The following clarifications apply:

- If the **normalize** method is called on an empty text node, the text node is not combined with adjacent text nodes that have content.
- If the **normalize** method is called on a text node that is empty and is the only child node of an element, the text node is not removed.

2.2.2 [DOM Level 1] Section 2.4, Objects related to HTML documents

C0006:

The specification states:

```
Attribute
cookie
The cookies associated with this document.
```

All Document Modes (All Versions)

An item is not added to the cookie collection if the optional `path=<some_path>` string sequence specifies a file name. The value of `<some_path>` is limited to a subset of URLs for which the cookie is valid.

All Document Modes (Internet Explorer 7)

The maximum size for the value of the **cookie** attribute is defined as 4096 bytes (4 KB).

All Document Modes (All Versions except Internet Explorer 7)

The maximum size for the value of the **cookie** attribute is defined as 10240 bytes (10Kb).

C0007:

The specification states:

```
Method
getElementsByName Returns the (possibly empty) collection of elements whose name
value is given by elementName.

Parameters
elementName  The name attribute value for an element.

Return Value The matching elements.

This method raises no exceptions.
```

IE8 Mode, IE9 Mode, and IE10 Mode (All Versions)

A collection of elements is returned where each element `name`, `id`, or `uniquename` attribute value matches the **elementName** parameter.

2.2.3 [DOM Level 1] Section 2.5.1, Property Attributes

C0008:

The specification states:

```
The return value of an attribute that has a data type that is a value list is
always capitalized, independent of the case of the value in the source document.
```

All Document Modes (All Versions)

Value list attribute values are converted to lowercase.

2.2.4 [DOM Level 1] Section 2.5.5, Object definitions

C0009:

The specification states:

```
Attributes
aLink
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.
```

Quirks Mode and IE7 Mode (All Versions)

The value of the **aLink** attribute is always converted into a hexadecimal RGB value when it is stored in the DOM or retrieved using the **getAttribute** method.

IE8 Mode, IE9 Mode, and IE10 Mode (All Versions)

The value of the **aLink** property of the **HTMLBodyElement** interface is stored in the DOM as a hexadecimal RGB value (i.e. setting it to "blue" yields #0000FF for the color when retrieved via `BODY.aLink`).

C0010:

The specification states:

```
Attribute
background
URI of the background texture tile image. See the background attribute definition
in HTML 4.0. This attribute is deprecated in HTML 4.0.
```

IE8 Mode (All Versions)

When the value of the **background** attribute for the **BODY** element is specified in HTML as a relative URI, the value is converted and stored as an absolute URI in the **background** attribute of the **HTMLBodyElement**.

C0011:

The specification states:

```
Attribute
bgColor
Document background color. See the bgcolor attribute definition in HTML 4.0. This
attribute is deprecated in HTML 4.0.
```

All Document Modes (All Versions)

If the value of the **bgcolor** attribute is specified as a color string (such as `black`) in HTML, the value is converted and stored as a hexadecimal RGB color value in the **bgColor** attribute of the **HTMLBodyElement** interface.

C0012:

The specification states:

```
Attribute
link
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.
```

All Document Modes (All Versions)

If the value of the **link** attribute is specified as a color string (such as `black`) in HTML, the value is converted and stored as a hexadecimal RGB color value in the **link** attribute of the **HTMLBodyElement** interface.

C0013:

The specification states:

```
Attribute
text
Document text color. See the text attribute definition in HTML 4.0. This attribute
is deprecated in HTML 4.0.
```

All Document Modes (All Versions)

If the value of the **text** attribute is specified as a color string (such as `black`) in HTML, the value is converted and stored as a hexadecimal RGB color value in the **text** attribute of the **HTMLBodyElement** interface.

C0014:

The specification states:

```
Attribute
vLink
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.
```

All Document Modes (All Versions)

If the value of the **vlink** attribute is specified as a color string (such as `black`) in HTML, the value is converted and stored as a hexadecimal RGB color value in the **vLink** attribute of the **HTMLBodyElement** interface.

C0015:

The specification states:

```
Attribute
elements
Returns a collection of all control elements in the form.
```

All Document Modes (All Versions)

FIELDSET elements are included when compiling the `elements` collection of a form.

C0016:

The specification states:

```
Attribute
length
The number of form controls in the form.
```

All Document Modes (All Versions)

The **FIELDSET** element is included when counting the number of form controls in a form.

C0017:

The specification states:

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

All Document Modes (All Versions)

The value of `acceptCharset` is set to `unknown` when no value is supplied for the **accept-charset** attribute of a **FORM** element.

C0018:

The specification states:

```
Attribute
action
Server-side form handler. See the action attribute definition in HTML 4.0.
```

IE8 Mode and IE9 (All Versions)

When the value of the **action** attribute for the **FORM** element is specified in HTML as a relative URI, the value is converted and stored as an absolute URI in the **action** attribute of the **HTMLFormElement**. If not already present, a slash is appended to the end of the URI when a file name is not specified.

C0019:

The specification states:

```
Attribute
type
The type of control created.
```

All Document Modes (All Versions)

The **type** attribute returns a value of `select-one`.

C0020:

The specification states:

```
Attribute
value
The current form control value.
```

All Document Modes (All Versions)

The **value** attribute of **HTMLSelectElement** returns the value of the **OPTION** element identified by the **selectedIndex** attribute of **HTMLSelectElement**.

C0021:

The specification states:

```
Attribute
length
The number of options in this SELECT.
```

All Document Modes (All Versions)

The following clarifications apply:

- The `length` attribute is not read-only.
- When the value of the `length` attribute is increased, the number of items in the **select** list collection is increased by appending empty items to the end of the list. The **length** property of the element is adjusted accordingly.
- When the value of the `length` attribute is decreased, the number of items in the **select** list collection is decreased by removing items from the end of the list. The **length** property of the element is adjusted accordingly, and the removed items and their values are not recoverable.

C0022:

The specification states:

```
Attribute
disabled
The control is unavailable in this context. See the disabled attribute definition  in HTML
4.0.
```

All Document Modes (All Versions)

HTMLSelectElement.disabled returns `true` if the content attribute is set to a value other than `false` or no value.

Various attributes of a disabled element are allowed to be set and the disabled element updated accordingly. For example, if the **length** attribute of a disabled **SELECT** element is set, the number of items in the **select** list collection is updated, and the disabled element is rendered.

C0023:

The specification states:

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

All Document Modes (All Versions)

The value of the **tabIndex** attribute is returned as 0 if the **tabIndex** attribute of the element is declared and the attribute value is not specified or invalid.

C0024:

The specification states:

```
Attribute
index
The index of this OPTION in its parent SELECT.
```

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

The **index** attribute is read-only.

C0025:

The specification states:

```
Form control.
Note. Depending upon the environment 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 HTML 4.0.
```

All Document Modes (All Versions)

The following clarifications apply:

- The **value** attribute cannot be set or retrieved if the **type** attribute of the **INPUT** element has a value of `file`.
- If an **INPUT** element has a **type** attribute with a value of `password`, the text field is not masked when the value for the field is retrieved from the **value** attribute of **HTMLInputElement**.

C0026:

The specification states:

```
Attribute
type
The type of button. See the type attribute definition in HTML 4.0.
```

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

The **type** attribute of a button cannot be set.

C0027:

The specification states:

```
Attribute
value
The current form control value. See the value attribute definition in HTML 4.0.
```

Quirks Mode and IE7 Mode (All Versions)

Text contained by a **BUTTON** element is returned by the **value** attribute of **HTMLButtonElement**. If no text is contained by the **BUTTON** element, the value of the **value** attribute for the **BUTTON** element is returned. If no value is assigned or the **value** attribute is not specified for the **BUTTON** element, the **value** attribute of **HTMLButtonElement** returns an empty string.

C0029:

The specification states:

```
Attribute
href
The URI of the linked resource. See the href attribute definition in HTML 4.0.
```

IE8 Mode and IE9 (All Versions)

When the value of the **href** attribute for the **A** element is specified in HTML as a relative URI, the value is converted and stored as an absolute URI in the **href** attribute of the **HTMLAnchorElement**. If not already present, a slash is appended to the end of the URI when a file name is not specified.

C0030:

The specification states:

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

All Document Modes (All Versions)

The following clarifications apply:

- The `height`, `width`, `vspace`, and `hspace` attributes are converted to long integers
- The `lowSrc` attribute is not supported

C0031:

The specification states:

```
Attribute
longDesc
URI designating a long description of this image or frame. See the longdesc
attribute definition in HTML 4.0.
```

IE8 Mode, IE9 Mode, and IE10 Mode (All Versions)

When the value of the **longdesc** attribute for the **IMG** element is specified in HTML as a relative URI, the value is converted and stored as an absolute URI in the **longDesc** attribute of the **HTMLImageElement**. If not already present, a slash is appended to the end of the URI when a file name is not specified.

C0032:

The specification states:

```
Attribute
src
URI designating the source of this image. See the src attribute definition in HTML 4.0.
```

All Document Modes (All Versions)

The **src** value is converted into an absolute URI.

C0033:

The specification states:

```
Attribute
align
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.
```

All Document Modes (All Versions)

The **align** attribute returns an empty string as a default value.

C0034:

The specification states:

```
Attribute
codeBase
Base URI for classid, data, and archive attributes. See the codebase attribute
definition in HTML 4.0.
```

All Document Modes (All Versions)

The **codeBase** attribute returns an empty string as a default value.

C0035:

The specification states:

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

All Document Modes (All Versions)

The **codeType** attribute returns an empty string as a default value.

C0036:

The specification states:

```
Attribute
data
```

```
A URI specifying the location of the object's data. See the data attribute
definition in HTML 4.0.
```

IE8 Mode, IE9 Mode, and IE10 Mode (All Versions)

When the value of the **data** attribute for the **OBJECT** element is specified in HTML as a relative URI, the value is converted and stored as an absolute URI in the **data** attribute of the **HTMLObjectElement**.

C0037:

The specification states:

```
Attribute
hspace
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.
```

All Document Modes (All Versions)

The **hspace** attribute is treated as a number.

C0038:

The specification states:

```
Attribute
vspace
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.
```

All Document Modes (All Versions)

The **vspace** attribute is treated as a number.

C0039:

The specification states:

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

All Document Modes (All Versions)

The default value for **valueType** is an empty string.

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

If the **name** attribute is not provided on the **PARAM** element by the author, the element is not created in the DOM by the HTML parser.

C0040:

The specification states:

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

Quirks Mode and IE7 Mode (All Versions)

The default value for the **valueType** attribute is an empty string.

C0041:

The specification states:

```
Attribute
hspace
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.
```

All Document Modes (All Versions)

The **hspace** attribute is treated as a number.

C0042:

The specification states:

```
Attribute
href
The URI of the linked resource. See the href attribute definition in HTML 4.0.
```

IE8 Mode, IE9 Mode, and IE10 Mode (All Versions)

When reading the value of the **href** attribute specified in markup, an extra slash (/) is added at end of the string to make it an absolute URI if it is not already an absolute URI. If the value of the **href** attribute is set through script, then the extra slash is not added.

C0043:

The specification states:

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

Quirks Mode and IE7 Mode (All Versions)

Shape attribute elements are capitalized when read from the DOM. For example, HTML markup `shape="circle"` is read by the DOM as `CIRCLE`.

IE8 Mode, IE9 Mode, and IE10 Mode (All Versions)

`shape="default"` is interpreted as `"rect"`.

C0044:

The specification states:

```
Method
deleteRow Delete a table row.
Parameters
index  The index of the row to be deleted.
This method returns nothing.
This method raises no exceptions.
```

All Document Modes (All Versions)

The **deleteRow** method with no parameters is also supported. When calling the **deleteRow** method with no parameters, the last row of the table is deleted.

C0045:

The specification states:

```
Interface HTMLTableSectionElement

Attribute align
Horizontal alignment of data in cells. See the align attribute for HTMLTheadElement
for details.
```

All Document Modes (All Versions)

The **Align** attribute is supported as described. However, the **HTMLTheadElement** interface referred to in the specification for the **Align** attribute is not defined in [DOM Level 1].

C0046:

The specification states:

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

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

The **ch** attribute is always returned as an empty string.

C0047:

The specification states:

```
Attribute
```

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

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

The **chOff** attribute is always returned as an empty string.

C0048:

The specification states:

```
Interface HTMLTableSectionElement

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

All Document Modes (All Versions)

The **vAlign** attribute is supported as described. However, the **HTMLTheadElement** interface referred to in the specification for the **vAlign** attribute is not defined in [DOM Level 1].

C0049:

The specification states:

```
Method
insertRow Insert a row into this section.
Parameters
index  The row number where to insert a new row.
Return Value
The newly created row.
This method raises no exceptions.
```

All Document Modes (All Versions)

Specifying a value of `-1` for the **index** parameter causes the **insertRow** method to append a new row to the bottom of the table.

C0050:

The specification states:

```
Method
deleteRow Delete a row from this section.
Parameters
index  The index of the row to be deleted.
This method returns nothing.
This method raises no exceptions.
```

All Document Modes (All Versions)

Specifying a value of `-1` for the **index** parameter causes the **deleteRow** method to remove the last row of the table. This behavior is expected in DOM L2 but not in DOM L1.

C0051:

The specification states:

```
Attribute
align
Horizontal alignment of data in cell. See the align attribute definition in HTML 4.0.
```

All Document Modes (All Versions)

The values `char` or `justify` are not recognized as valid choices for the **align** attribute in a **TD** element. If `char` or `justify` are set in HTML for a **TD** element, **td.align** evaluates to an empty string.

C0052:

The specification states:

```
Attribute
bgColor
Cell background color. See the bgcolor attribute definition in HTML 4.0. This
attribute is deprecated in HTML 4.0.
```

Quirks Mode and IE7 Mode (All Versions)

The value of the **bgColor** attribute in a **TD** element is always converted to a hexadecimal RGB value when it is stored in the DOM. For example, setting the **bgColor** attribute of a **TD** element to `blue` in HTML yields a value of `#0000FF` for the **bgColor** attribute when retrieved using the **getAttribute** method.

IE8 Mode, IE9 Mode, and IE10 Mode (All Versions)

The value of the **bgColor** attribute in a **TD** element remains the same as the author's original setting in HTML. For example, setting the **bgColor** attribute of a **TD** element to `blue` in HTML yields a value of `blue` when using the **getAttribute** method to retrieve the color.

However, the value of the **bgColor** property of the **HTMLFontElement** interface is stored in the DOM as a hexadecimal RGB value (for example, setting it to `blue` in HTML yields `#0000FF` for the color when retrieved using `TD.bgColor`).

C0053:

The specification states:

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

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

The **char** attribute is not associated with the **ch** attribute in the DOM.

C0054:

The specification states:

```
Attribute
chOff
Offset of alignment character. See the charoff attribute definition in HTML 4.0.
```

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

The **chOff** attribute is not supported.

C0055:

The specification states:

```
Attribute
height
Cell height. See the height attribute definition in HTML 4.0. This attribute is
deprecated in HTML 4.0.
```

All Document Modes (All Versions)

The value of the **height** attribute is serialized by dropping the "px" suffix when set in the DOM. For example, writing `<td height="100px">` in markup, accessing **td.height** from the DOM, returns `100`. However, percentage values are unmodified. For example, with `<td height="100%">` in markup, the DOM returns `100%`.

C0056:

The specification states:

```
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;
};
```

IE8 Mode, IE9 Mode, and IE10 Mode (All Versions)

If the value of the **longdesc** attribute or **src** attribute for the **FRAME** element is specified in HTML as a relative URI, the value is converted and stored as an absolute URI in the **longDesc** attribute or **src** attribute, respectively, of the **HTMLFrameElement**. If not already present, a slash is appended to the end of the URI when a file name is not specified.

C0057:

The specification states:

```
interface HTMLFrameElement : HTMLElement

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

All Document Modes (All Versions)

The following clarifications apply:

- The value of the **noresize** attribute is stored as a string.
- This attribute is an expando. If the **noresize** attribute is assigned a value in HTML, that value is returned as the value of the **noResize** DOM attribute rather than the Boolean value `true`.
- If the **noresize** attribute is not specified in HTML, or is specified without a value, the **noResize** DOM attribute returns a value of undefined rather than a default Boolean value of `false`.

2.3 Error Handling

There are no additional considerations for error handling.

2.4 Security

There are no additional security considerations.

3 Change Tracking

This section identifies changes that were made to the [MS-DOM1] protocol document between the February 2012 and July 2012 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated.**

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact protocol@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1 Introduction	Updated document to remove beta tagging.	N	Content updated.

4 Index