[MS-DOM2C]:

Internet Explorer Document Object Model (DOM) Level 2 Core Standards Support Document

---

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **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 can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation 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 might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are 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 as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation does 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 documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
3/17/2010	0.1	New	Released new document.
3/26/2010	1.0	None	Introduced no new technical or language changes.
5/26/2010	1.2	None	Introduced no new technical or language changes.
9/8/2010	1.3	Major	Significantly changed the technical content.
2/10/2011	2.0	Minor	Clarified the meaning of the technical content.
2/22/2012	3.0	Major	Significantly changed the technical content.
7/25/2012	3.1	Minor	Clarified the meaning of the technical content.
2/6/2013	3.2	Minor	Clarified the meaning of the technical content.
6/26/2013	4.0	Major	Significantly changed the technical content.
3/31/2014	4.0	None	No changes to the meaning, language, or formatting of the technical content.
1/22/2015	5.0	Major	Updated for new product version.
7/7/2015	5.1	Minor	Clarified the meaning of the technical content.
11/2/2015	5.1	None	No changes to the meaning, language, or formatting of the technical content.
3/22/2016	5.2	Minor	Clarified the meaning of the technical content.
11/2/2016	5.2	None	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This document describes the level of support provided by Microsoft web browsers for the *Document Object Model (DOM) Level 2 Core Specification Version 1.0* [DOM Level 2 - Core], W3C Recommendation 13 November, 2000.

The [DOM Level 2 - Core] specification contains guidance for authors of webpages and browser users, in addition to 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 defined in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

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.

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

[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

[XPointer] Grosso, P., Maler, E., Marsh, J., and Walsh, N., "XPointer Framework", W3C Recommendation 25 March 2003, http://www.w3.org/TR/2003/REC-xptr-framework-20030325/

1.2.2 Informative References

None.

1.3 Microsoft Implementations

The following Microsoft web browser versions implement some portion of [DOM Level 2 - Core]:

- Windows Internet Explorer 7
- Windows Internet Explorer 8
- Windows Internet Explorer 9
- Windows Internet Explorer 10
- Internet Explorer 11
- Internet Explorer 11 for Windows 10

- Microsoft Edge

Each browser version may implement multiple document rendering modes. The modes vary from one to another in support of the standard. The following table lists the document modes supported by each browser version.

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
Internet Explorer 11	Quirks mode IE7 mode IE8 mode IE9 mode IE10 mode IE11 mode
Internet Explorer 11 for Windows 10	Quirks mode IE7 mode IE8 mode IE9 mode IE10 mode IE11 mode
Microsoft Edge	EdgeHTML Mode

For each variation presented in this document there is a list of the document modes and browser versions that exhibit the behavior described by the variation. All combinations of modes and versions that are not listed conform to the specification. For example, the following list for a variation indicates that the variation exists in three document modes in all browser versions that support these modes:

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

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 2 - Core] 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 2 - Core] and whether they are considered normative or informative.

Sections	Normative/Informative
1	Normative
Appendix A-F	Informative

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####	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####	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####	Identifies extensibility points (such as optional implementation-specific data) in the target specification, which can impair interoperability.

2 Standards Support Statements

This section contains a full list of variations, clarifications, and extension points in the Microsoft implementation of [DOM Level 2 - Core].

- 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 2 - Core].

2.1.1 [DOM Level 2 - Core] Section 1.1.8, XML Namespaces

V0001:

The specification states:

```
As far as the DOM is concerned, special attributes used for declaring XML
namespaces are still exposed and can be manipulated just like any other attribute.
However, nodes are permanently bound to namespace URIs as they get created.
```

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

Special attributes that are used to declare namespace-like constructs that begin with "xmlns:" on the root **html** element are supported in HTML documents. These attributes create side effects during HTML parsing, but they do not affect the namespace URI of created elements.

2.1.2 [DOM Level 2 - Core] Section 1.2, Fundamental Interfaces

V0002:

The specification defines the **DOMException** exception**:**DOMException" .

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

The **DOMException** interface is not supported.

V0003:

The specification states:

```
IDL Definition
interface DOMImplementation {
  boolean            hasFeature(in DOMString feature,
                                in DOMString version);
  // Introduced in DOM Level 2:
  DocumentType       createDocumentType(in DOMString qualifiedName,
                                        in DOMString publicId,
                                        in DOMString systemId)
```

```
                                        raises(DOMException);
  // Introduced in DOM Level 2:
  Document           createDocument(in DOMString namespaceURI,
                                    in DOMString qualifiedName,
                                    in DocumentType doctype)
                                        raises(DOMException);
};
```

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

The following methods of the **DOMImplementation** interface are not supported:

- **createDocumentType**
- **createDocument**

V0004:

The specification states:

```
hasFeature
Test if the DOM implementation implements a specific feature.

Parameters
feature of type DOMString
The name of the feature to test (case-insensitive). The values used by DOM features
are defined throughout the DOM Level 2 specifications and listed in the Conformance
section. The name must be an XML name. To avoid possible conflicts, as a
convention, names referring to features defined outside the DOM specification
should be made unique by reversing the name of the Internet domain name of the
person (or the organization that the person belongs to) who defines the feature,
component by component, and using this as a prefix. For instance, the W3C SVG
Working Group defines the feature "org.w3c.dom.svg".
version of type DOMString

This is the version number of the feature to test. In Level 2, the string can be
either "2.0" or "1.0". If the version is not specified, supporting any version of
the feature causes the method to return true.

Return Value
boolean
 true if the feature is implemented in the specified version, false otherwise.

No Exceptions
```

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

The **hasFeature** method of the **DOMImplementation** interface returns `FALSE` for the Core module and the version strings "1.0" and "2.0".

V0005:

The specification states:

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

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

The **DocumentFragment** interface inherits from the **Document** interface and has all of the methods and properties that a document instance would have.

V0006:

The specification states:

```
IDL Definition
interface Document : Node {
readonly attribute DocumentType     doctype;
readonly attribute DOMImplementation  implementation;
readonly attribute Element          documentElement;
Element            createElement(in DOMString tagName)raises(DOMException);
DocumentFragment   createDocumentFragment();
Text               createTextNode(in DOMString data);
Comment            createComment(in DOMString data);
CDATASection       createCDATASection(in DOMString data)raises(DOMException);
ProcessingInstruction createProcessingInstruction(in DOMString target, in DOMString
data)raises(DOMException);
Attr               createAttribute(in DOMString name)raises(DOMException);
EntityReference    createEntityReference(in DOMString name)raises(DOMException);
NodeList           getElementsByTagName(in DOMString tagname);
// Introduced in DOM Level 2:
Node               importNode(in Node importedNode, in boolean deep)raises(DOMException);
// Introduced in DOM Level 2:
Element            createElementNS(in DOMString namespaceURI, in DOMString
qualifiedName)raises(DOMException);
// Introduced in DOM Level 2:
Attr               createAttributeNS(in DOMString namespaceURI, in DOMString
qualifiedName)raises(DOMException);
// Introduced in DOM Level 2:
NodeList           getElementsByTagNameNS(in DOMString namespaceURI, in DOMString localName);
// Introduced in DOM Level 2:
Element            getElementById(in DOMString elementId);
};
```

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

The following methods of the **Document** interface are not supported:

- **createAttributeNS**
- **createCDATASection**
- **createElementNS**
- **createProcessingInstruction**
- **getElementsByTagNameNS**
- **importNode**

All Document Modes (All Versions)

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

V0007:

The specification states:

```
Interface Document

Methods
createAttribute
```

```
Creates an Attr of the given name. Note that the Attr instance can then be set on
an Element using the setAttributeNode method.

To create an attribute with a qualified name and namespace URI, use the
createAttributeNS method.

Parameters
name of type DOMString
The name of the attribute.

Return Value
Attr A new Attr object with the nodeName attribute set to name, and localName,
prefix, and namespaceURI set to null. The value of the attribute is the empty
string.

Exceptions
DOMException INVALID CHARACTER ERR: Raised if the specified name contains an
illegal character.
```

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

The **Attr** instance created by the **createAttribute** method has an undefined **nodeValue** instead of an empty string.

V0008:

The specification states:

```
Interface Document

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

Parameters
data of type DOMString
The data for the CDATASection contents.

Return Value
CDATASection 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.

V0009:

The specification states:

```
Interface Document

Methods
createDocumentFragment
Creates an empty DocumentFragment object.

Return Value
DocumentFragment A new DocumentFragment.

No Parameters
```

```
No Exceptions
```

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

A full document-derived object is returned when the **createDocumentFragment** method is called.

V0010:

The specification states:

```
Interface Document

Methods
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.In addition, if there are known attributes with default values,
Attr nodes representing them are automatically created and attached to the
element.To create an element with a qualified name and namespace URI, use the
createElementNS method.

Parameters
tagName of type DOMString
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
Element A new Element object with the nodeName attribute set to tagName, and
localName, prefix, and namespaceURI set to null.

Exceptions
DOMException INVALID CHARACTER ERR: Raised if the specified name contains an
illegal character.
```

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

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**.**

V0011:

The specification defines the **createEntityReference** and **createProcessingInstruction** methods:

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

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

All Document Modes (All Versions)

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

V0012:

The specification states:

```
Interface Document

Method
getElementById introduced in DOM Level 2
Returns the  Element whose ID is given by elementId. If no such element exists,
returns null. Behavior is not defined if more than one element has this ID.

Note: The DOM implementation must have information that says which attributes are
of type ID. Attributes with the name "ID" are not of type ID unless so defined.
Implementations that do not know whether attributes are of type ID or not are
expected to return null.

Parameters
elementId of type DOMString
The unique id value for an element.

Return Value
Element The matching element.

No Exceptions
```

Quirks Mode and IE7 Mode (All Versions)

The **getElementById** method of the **Document** interface performs a case-insensitive compare against the ID values of elements.

V0013:

The specification defines the **importNode** method.

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

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

V0014:

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;
    // Modified in DOM Level 2:
    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);
    // Modified in DOM Level 2:
    void               normalize();
    // Introduced in DOM Level 2:
    boolean            isSupported(in DOMString feature,
                                   in DOMString version);
    // Introduced in DOM Level 2:
    readonly attribute DOMString        namespaceURI;
    // Introduced in DOM Level 2:
             attribute DOMString        prefix;
                                        // raises(DOMException) on setting

    // Introduced in DOM Level 2:
    readonly attribute DOMString        localName;
    // Introduced in DOM Level 2:
    boolean            hasAttributes();
};
```

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

The following attributes of the **Node** interface are not supported:

- **localName**
- **namespaceURI**
- **prefix**

The following constants of the **Node** interface are not supported:

- ATTRIBUTE_NODE
- CDATA_SECTION_NODE
- COMMENT_NODE
- DOCUMENT_FRAGMENT_NODE
- DOCUMENT_NODE
- DOCUMENT_TYPE_NODE
- ELEMENT_NODE
- ENTITY_REFERENCE_NODE

- ENTITY_NODE
- NOTATION_NODE
- PROCESSING_INSTRUCTION_NODE
- TEXT_NODE

The following methods of the **Node** interface are not supported:

- **hasAttributes**
- **isSupported**

V0015:

The specification states:

```
The values of nodeName, nodeValue, and attributes vary according to the node type as follows:

Interface             nodeName           nodeValue                    attributes
Attr                  name of attribute  value of attribute           null
CDATASection          #cdata-section     content of the CDATA Section null
Comment               #comment           content of the comment       null
Document              #document          null                         null
DocumentFragment      #document-fragment null                         null
DocumentType          document type name null                         null
Element               tag name           null                         NamedNodeMap
Entity                entity name        null                         null
EntityReference       name of entity     null                         null
                      referenced
Notation              notation name      null                         null
ProcessingInstruction target             entire content excluding     null
                                         the target
Text                  #text              content of the text node     null
```

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

The **DocumentType** interface is not supported. Actual document types in markup are created as instances of **Comment**, causing the values of the instance to match those of the **Comment** entry as opposed to the **DocumentType** entry.

V0016:

The specification states:

```
Interface Node

Attributes
childNodes of type NodeList, readonly
A NodeList that contains all children of this node. If there are no children, this
is a NodeList containing no nodes
```

IE8 Mode (All Versions)

Splitting multiple text nodes under an element with **splitText()** can prevent the **childNodes** collection from immediately updating. The addition of other tree modifications causes the **childNodes** collection to synchronize again.

V0017:

The specification states:

```
Interface Node

The values of nodeName, nodeValue, and attributes vary according to the node type

Attributes

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

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

The **nodeName** attribute of the **Node** interface 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, **nodeName** drops the element prefixes and does not return uppercase values.

V0018:

The specification states:

```
Interface Node

Attributes
parentNode of type Node, readonly
The parent of this node. All nodes, except Attr, Document, DocumentFragment,
Entity, and Notation 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.

V0019:

The specification states:

```
Interface Node

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 of type Node
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
Node 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 that trigger **HIERARCHY_REQUEST_ERR** and **WRONG_DOCUMENT_ERR** result in a JavaScript error. The error message is `Invalid argument` with the HRESULT value `0x80070057`.

The following elements raise an exception when an attempt is made 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**

IE9 Mode, IE10 Mode, and IE11 Mode (All Versions)

The conditions that trigger the **WRONG_DOCUMENT_ERR** exception cause the following behavior:

- The node is adopted and inserted.
- No exception is thrown.

V0020:

The specification states:

```
Interface Node

Method
cloneNode
Returns a duplicate of this node, i.e., serves as a generic copy constructor for
nodes. The duplicate node has no parent; (parentNode is null.).Cloning an Element
copies all attributes and their values, including those generated by the XML
processor to represent defaulted attributes, but this method does not copy any text
it contains unless it is a deep clone, since the text is contained in a child Text
node.

Cloning an Attribute directly, as opposed to be cloned as part of an Element
cloning operation, returns a specified attribute (specified is true). Cloning any
other type of node simply returns a copy of this node.

Note that cloning an immutable subtree results in a mutable copy, but the children
of an EntityReference clone are readonly. In addition, clones of unspecified Attr
nodes are specified. And, cloning Document, DocumentType, Entity, and Notation
nodes is implementation dependent.

Parameters
deep of type boolean
If true, recursively clone the subtree under the specified node; if false, clone
only the node itself (and its attributes, if it is an Element).

Return Value
Node The duplicate node.

No Exceptions
```

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

The cloned attribute objects do not have to be specifically set to `true`.

V0021:

The specification states:

```
Interface Node

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 of type Node
The node to insert.

refChild of type Node
The reference node, i.e., the node before which the new node must be inserted.

Return Value
```

```
Node 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 or if the parent of the node
being inserted is readonly.

NOT_FOUND_ERR: Raised if refChild is not a child of this node.
```

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

With the **insertBefore** method, conditions that trigger **HIERARCHY_REQUEST_ERR** and **WRONG_DOCUMENT_ERR** result in a JavaScript Error. The error message is `Invalid argument` with the HRESULT value `0x80070057`.

The following elements raise an exception when an attempt is made 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**

IE9 Mode, IE10 Mode, and IE11 Mode (All Versions)

The conditions that trigger the **WRONG_DOCUMENT_ERR** exception cause the following behavior:

- The node is adopted and inserted.
- No exception is thrown.

V0022:

The specification states:

```
Interface Node

Method
normalize modified in DOM Level 2
Puts all Text nodes in the full depth of the sub-tree underneath this Node,
including attribute nodes, into a "normal" form where only structure (e.g.,
elements, comments, processing instructions, CDATA sections, and entity references)
separates Text nodes, i.e., there are neither adjacent Text nodes nor empty 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
[XPointer] lookups) that depend on a particular document tree structure are to be
used.

Note: In cases where the document contains CDATASections, the normalize operation
alone may not be sufficient, since XPointers do not differentiate between Text
nodes and CDATASection nodes.
```

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

The following variations apply:

- An empty text node is not collapsed into an adjacent text node when calling the **normalize** method.
- An empty text node is not removed if that node is the only child of its parent.

V0023:

The specification states:

```
Interface Node

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

Parameters
oldChild of type Node
The node being removed.
```

```
Return Value
Node 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)

With the **removeChild** method, conditions that trigger **NOT_FOUND_ERR** result in a JavaScript error. The error message is `Invalid argument` with the HRESULT value `0x80070057`.

V0024:

The specification states:

```
Interface Node

Method
replaceChild
Replaces the child node oldChild with newChild in the list of children, and returns
the oldChild node.

If newChild is a DocumentFragment object, oldChild is replaced by all of the
DocumentFragment children, which are inserted in the same order. If the newChild is
already in the tree, it is first removed.

Parameters
newChild of type Node
The new node to put in the child list.

oldChild of type Node
The node being replaced in the list.

Return Value
Node 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 if 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 or the parent of the new 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)

With the **replaceChild** method, conditions that trigger **HIERARCHY_REQUEST_ERR**, **WRONG_DOCUMENT_ERR**, and **NOT_FOUND_ERR** result in a JavaScript error. The error message is `Invalid argument` with the HRESULT value `0x80070057`.

V0025:

The specification states:

```
interface NamedNodeMap {
  Node               getNamedItem(in DOMString name);
  Node               setNamedItem(in Node arg)
                                     raises(DOMException);
  Node               removeNamedItem(in DOMString name)
                                     raises(DOMException);
  Node               item(in unsigned long index);
  readonly attribute unsigned long    length;
  // Introduced in DOM Level 2:
  Node               getNamedItemNS(in DOMString namespaceURI,
                                    in DOMString localName);
  // Introduced in DOM Level 2:
  Node               setNamedItemNS(in Node arg)
                                     raises(DOMException);
  // Introduced in DOM Level 2:
  Node               removeNamedItemNS(in DOMString namespaceURI,
                                      in DOMString localName)
                                     raises(DOMException);
};
```

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

The following methods are not supported:

- **getNamedItemNS**
- **removeNamedItemNS**
- **setNamedItemNS**

V0026:

The specification states:

```
Interface NamedNodeMap

Method
getNamedItem
Retrieves a node specified by name.

Parameters
name of type DOMString
The nodeName of a node to retrieve.

Return Value
Node A Node (of any type) with the specified nodeName, or null if it does not
identify any node in this map.

No Exceptions
```

Quirks Mode and IE7 Mode (All Versions)

The **getNamedItem** method creates objects for attributes that do not exist in the collection.

V0027:

The specification states:

```
Interface NamedNodeMap

Method
item
Returns the indexth item in the map. If index is greater than or equal to the
```

```
number of nodes in this map, this returns null.

Parameters
index of type unsigned long
Index into this map.

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

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 raises a JSError exception with an error message of `Invalid argument` and an HRESULT value of `0x80070057`.

V0028:

The specification states:

```
Interface NamedNodeMap

Method
removeNamedItem
Removes a node specified by name. When this map contains the attributes attached to
an element, if the removed attribute is known to have a default value, an attribute
immediately appears containing the default value as well as the corresponding
namespace URI, local name, and prefix when applicable.

Parameters
name of type DOMString
The nodeName of the node to remove.

Return Value
Node The node removed from this map if a node with such a name exists.

Exceptions
DOMException NOT_FOUND_ERR: Raised if there is no node named name in this map.

NO_MODIFICATION_ALLOWED_ERR: Raised if this map is readonly.
```

Quirks Mode and IE7 Mode (All Versions)

Exceptions are not raised when the node cannot be found and the return value for the **removeNamedItem** method is null.

V0030:

The specification defines the **substringData** method of the **CharacterData** interface.

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

The following variations apply:

- An exception is not raised if the offset is greater than the number of 16-bit units in the data.
- Named **DOMExceptions** are not returned. The exception creates an error object for Invalid Parameter with a number property = (0xFFFF0000 or 0x57) rather than an INDEX_SIZE_ERR exception with code=0x1.

V0031:

The specification states:

```
IDL Definition
interface Attr : Node {
  readonly attribute DOMString        name;
  readonly attribute boolean          specified;
           attribute DOMString        value;
                                        // raises(DOMException) on setting

  // Introduced in DOM Level 2:
  readonly attribute Element          ownerElement;
};
```

Quirks Mode and IE7 Mode (All Versions)

The **ownerElement** attribute of the **Attr** interface is not supported.

V0032:

The specification states:

```
Interface Attr

Attribute
specified of type boolean, readonly
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.
• If the attribute has no assigned value in the document and has a value of
#IMPLIED in the DTD, then the attribute does not appear in the structure model of
the document.
• If the ownerElement attribute is null (i.e. because it was just created or was
set to null by the various removal and cloning operations) specified is true.
```

Quirks Mode and IE7 Mode (All Versions)

The value of the **specified** attribute is not automatically changed to `true` when the **ownerElement** attribute is null.

V0033:

The specification states:

```
Interface Element
The Element interface represents an element in an HTML or XML document. Elements
may have attributes associated with them; since the Element interface inherits from
Node, the generic Node interface attribute attributes 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.

Note: In DOM Level 2, the method normalize is inherited from the Node interface
where it was moved.
```

All Document Modes (All Versions)

Attribute subtrees are not supported; only strings are supported.

V0034:

The specification states:

```
Interface Element

Method
getAttribute
Retrieves an attribute value by name.

Parameters
name of type DOMString
The name of the attribute to retrieve.

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

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.

V0035:

The specification states:

```
Interface Element
Method
getAttributeNode
Retrieves an attribute node by name.To retrieve an attribute node by qualified name
and namespace URI, use the getAttributeNodeNS method.

Parameters
name of type DOMString
The name (nodeName) of the attribute to retrieve.

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

No Exceptions
```

Quirks Mode and IE7 Mode (All Versions)

When using the **getAttributeNode** attribute of the **Element** interface, attribute nodes that are not specified (or have default values) are returned rather than being given a null value.

V0036:

The specification states:

```
Interface Element

Method
getElementsByTagName
Returns a NodeList of all descendant Elements with a given tag name, in the order
in which they are encountered in a preorder traversal of this Element tree.

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

Return Value
NodeList A list of matching Element nodes.

No Exceptions
```

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

The **getElementsByTagName** method of the **Element** interface implements two conditions not covered in the specification:

- If `object1.getElementsByTagName("*")` is called, an empty collection is returned.
- If `object1.getElementsByTagName("param")` is called, a collection containing all of the parameters in the document is returned, as if the call made actually was `document.getElementsByTagName("param")`.

V0037:

The specification states:

```
Interface Element

Method
removeAttribute
Removes an attribute by name. If the removed attribute is known to have a default
value, an attribute immediately appears containing the default value as well as the
corresponding namespace URI, local name, and prefix when applicable.

To remove an attribute by local name and namespace URI, use the removeAttributeNS
method.

Parameters
name of type DOMString
The name of the attribute to remove.

Exceptions
DOMException NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
No Return Value
```

Quirks Mode and IE7 Mode (All Versions)

The following variations apply:

- The **removeAttribute** method lookup is case-sensitive; this method includes an additional parameter.
- Default attributes are not re-created after the attribute is removed.
- 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.

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

The **removeAttribute** method implements one additional return value that reports whether the operations succeeded or failed. This is an extension to the standard, and the data type returned is a void.

The **removeAttribute** method does not remove attributes that are pre-defined in the XHTML DTD. Default values are re-created after the **removeAttribute** method is called on these attributes.

V0038:

The specification states:

```
Interface Element

Method
removeAttributeNode
Removes the specified attribute node. If the removed Attr has a default value it is
immediately replaced. The replacing attribute has the same namespace URI and local
name, as well as the original prefix, when applicable.

Parameters
oldAttr of type Attr
The Attr node to remove from the attribute list.

Return Value
Attr        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)

The following variations apply:

- With the **removeAttributeNode** method of the **Element** interface, default attributes are not re-created after the attribute is removed.
- Removal of event handler attributes (such as **onClick**) or style attributesdoes not cause the actual event handler to be removed or the inline style to be removed.

V0039:

The specification states:

```
Interface Element

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.

To set an attribute with a qualified name and namespace URI, use the setAttributeNS
method.

Parameters
name of type DOMString
The name of the attribute to create or alter.

value of type DOMString
Value to set in string form.

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

Quirks Mode and IE7 Mode (All Versions)

The following variations apply:

- The **setAttribute** method assigns attributes in a case-sensitive manner.
- The **setAttribute** method has an optional third parameter that 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, **setAttribute('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; for example, **className** (instead of 'class'), **htmlFor** (instead of 'for'), or **httpEquiv** (instead of 'http-equiv').

V0040:

The specification states:

```
Interface Text

Method
splitText
Breaks this node into two nodes at the specified offset, keeping both in the tree
as siblings. After being split, this node will contain all the content up to the
offset point. A new node of the same type, which contains all the content at and
after the offset point, is returned. If the original node had a parent node, the
new node is inserted as the next siblings of the original node. When the offset is
equal to the length of this node, the new node has no data.

Parameters
offset of type unsigned long
The 16-bit unit offset at which to split, starting from 0.

Return Value
Text The new node, of the same type as this node.

Exceptions
```

```
DOMException INDEX_SIZE_ERR: Raised if the specified offset is negative or greater
than the number of 16-bit units in data.
NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.
```

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

The following variations apply:

- The **childNodes** objects are kept in a cache and are invalidated any time there is a modification to the markup. Calling the **splitText** method of the **Text** interface does not trigger a markup modification and 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.
- The offset parameter is treated as though it is optional. If no offset is provided, then a default offset of 0 is used.

V0041:

The specification states:

```
Interface Comment
This interface inherits from CharacterData and represents the content of a comment,
i.e., all the characters between the starting '<!--' and ending '-->'. Note that
this is the definition of a comment in XML, and, in practice, HTML, although some
HTML tools may implement the full SGML comment structure.

IDL Definition
interface Comment : CharacterData {
};
```

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

The **Comment** interface inherits from **Element** rather than from **Node**.

V0029:

The specification states:

```
Interface NamedNodeMap

Method
setNamedItem
Adds a node using its nodeName attribute. If a node with that name is already
present in this map, it is replaced by the new one.
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 of type Node
A node to store in this map. The node will later be accessible using the value of
its nodeName attribute.

Return Value
Node If the new Node replaces an existing node the replaced 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 this map.
```

```
NO_MODIFICATION_ALLOWED_ERR: Raised if this map 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, IE9 Mode, IE10 Mode, and IE11 Mode (All Versions)

An exception is not raised when the argument was created from a different document.

2.2 Clarifications

The following subsections identify clarifications to recommendations made by [DOM Level 2 - Core].

2.2.1 [DOM Level 2 - Core] Section 1.2, Fundamental Interfaces

C0006:

The specification describes the **getElementsByTagName** and **getElementsByTagNameNS** methods on the Document interface.

IE9 Mode, IE10 Mode, and IE11 Mode (All Versions)

getElementsByTagName and **getElementsByTagNameNS** return a **HTMLCollection** rather than a NodeList.

C0001:

The specification states:

```
Interface Document

Attributes
Doctype of type DocumentType, readonly
The Document Type Declaration (see DocumentType) associated with this document. For
HTML documents as well as XML documents without a document type declaration this
returns null. The DOM Level 2 does not support editing the Document Type
Declaration. docType cannot be altered in any way, including through the use of
methods inherited from the Node interface, such as insertNode or removeNode.
```

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

The following clarifcations apply:

- Because the **DocumentType** interface is not supported, the **doctype** attribute returns null.
- The **DocumentType** instances in HTML documents are created as **Comment** instances and can be accessed using other DOM methods, typically **document.firstChild**.

C0002:

The specification states:

```
Interface Document

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

```
Parameters
data of type DOMString
The data for the node.

Return Value
Comment The new Comment object.

No Exceptions
```

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

With the **createComment** method of the **Document** interface, the **data** parameter is treated as optional and creates a **Comment Node** even when no parameter is provided.

C0003:

The specification states:

```
Interface Document

Method
createTextNode
Creates a Text node given the specified string.

Parameters
data of type DOMString
The data for the node.

Return Value
Text The new Text object.

No Exceptions
```

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

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

C0004:

The specification states:

```
Attributes
length of type unsigned long, readonly
The number of 16-bit units that are available through data and the substringData
method below. This may have the value zero, i.e., CharacterData nodes may be empty.
```

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

Text nodes are not created when a node contains only white space. The **length** attribute is not available if the text node is not created.

C0005:

The specification states:

```
Interface Element

Attribute
tagName of type DOMString, readonly
```

```
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.

2.3 Error Handling

There are no additional considerations for error handling.

2.4 Security

There are no additional security considerations.

3 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

4 Index