**[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](http://go.microsoft.com/fwlink/?LinkId=214445) or the [Microsoft Community Promise](https://go.microsoft.com/fwlink/?LinkId=214448). 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](mailto:iplg@microsoft.com).
* **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the [Patent Map](https://msdn.microsoft.com/en-us/openspecifications/dn750984).
* **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](http://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.

**Support.** For questions and support, please contact [dochelp@microsoft.com](mailto:dochelp@microsoft.com).

**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. |
| 3/14/2017 | 5.2 | None | No changes to the meaning, language, or formatting of the technical content. |
| 10/3/2017 | 5.2 | None | No changes to the meaning, language, or formatting of the technical content. |
| 2/22/2018 | 5.2 | None | No changes to the meaning, language, or formatting of the technical content. |
| 3/23/2018 | 5.2 | None | No changes to the meaning, language, or formatting of the technical content. |
| 8/28/2018 | 5.2 | None | No changes to the meaning, language, or formatting of the technical content. |

Table of Contents

[1 Introduction 4](#_Toc522770516)

[1.1 Glossary 4](#_Toc522770517)

[1.2 References 4](#_Toc522770518)

[1.2.1 Normative References 4](#_Toc522770519)

[1.2.2 Informative References 4](#_Toc522770520)

[1.3 Microsoft Implementations 4](#_Toc522770521)

[1.4 Standards Support Requirements 6](#_Toc522770522)

[1.5 Notation 6](#_Toc522770523)

[2 Standards Support Statements 7](#_Toc522770524)

[2.1 Normative Variations 7](#_Toc522770525)

[2.1.1 [DOM Level 2 - Core] Section 1.1.8, XML Namespaces 7](#_Toc522770526)

[2.1.2 [DOM Level 2 - Core] Section 1.2, Fundamental Interfaces 7](#_Toc522770527)

[2.2 Clarifications 29](#_Toc522770528)

[2.2.1 [DOM Level 2 - Core] Section 1.2, Fundamental Interfaces 29](#_Toc522770529)

[2.3 Error Handling 31](#_Toc522770530)

[2.4 Security 31](#_Toc522770531)

[3 Change Tracking 32](#_Toc522770532)

[4 Index 33](#_Toc522770533)

# 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]](https://go.microsoft.com/fwlink/?LinkId=182703), published 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.

## Glossary

**MAY, SHOULD, MUST, SHOULD NOT, MUST NOT:** These terms (in all caps) are used as defined in [[RFC2119]](https://go.microsoft.com/fwlink/?LinkId=90317). All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

## 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](https://go.microsoft.com/fwlink/?linkid=850906).

### 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](mailto:dochelp@microsoft.com). We will assist you in finding the relevant information.

[DOM Level 2 - Core] World Wide Web Consortium, "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/](https://go.microsoft.com/fwlink/?LinkId=182703)

[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](https://go.microsoft.com/fwlink/?LinkId=90317)

[XPointer] World Wide Web Consortium, "XPointer Framework", W3C Recommendation 25 March 2003, [http://www.w3.org/TR/2003/REC-xptr-framework-20030325/](https://go.microsoft.com/fwlink/?LinkID=183836&clcid=0x409)

### Informative References

None.

## Microsoft Implementations

The following Microsoft web browser versions implement some portion of [[DOM Level 2 - Core]](https://go.microsoft.com/fwlink/?LinkId=182703):

* 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.

## Standards Support Requirements

To conform to [[DOM Level 2 - Core]](https://go.microsoft.com/fwlink/?LinkId=182703) 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]](https://go.microsoft.com/fwlink/?LinkId=90317).)

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 |

## 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]](https://go.microsoft.com/fwlink/?LinkId=90317).) This does not include extensibility points. |
| E#### | Identifies extensibility points (such as optional implementation-specific data) in the target specification, which can impair interoperability. |

# Standards Support Statements

This section contains all variations and clarifications for the Microsoft implementation of [[DOM Level 2 - Core]](https://go.microsoft.com/fwlink/?LinkId=182703).

* Section [2.1](#Section_b26fe38dccb3468ab26b8858508b554a) describes normative variations from the MUST requirements of the specification.
* Section [2.2](#Section_94a21fd990c341e191546bacc37aa0ac) describes clarifications of the MAY and SHOULD requirements.
* Section [2.3](#Section_e2d4bb858bcc466fba131aa00873c980) considers error handling aspects of the implementation.
* Section [2.4](#Section_1f614ae31a5d4b5da91fbc868340a77c) considers security aspects of the implementation.

## Normative Variations

The following subsections describe normative variations from the MUST requirements of [[DOM Level 2 - Core]](https://go.microsoft.com/fwlink/?LinkId=182703).

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

V0001:

The specification states:

1. As far as the DOM is concerned, special attributes used for declaring XML
2. namespaces are still exposed and can be manipulated just like any other attribute.
3. 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.

### [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:

1. IDL Definition
2. interface DOMImplementation {
3. boolean hasFeature(in DOMString feature,
4. in DOMString version);
5. // Introduced in DOM Level 2:
6. DocumentType createDocumentType(in DOMString qualifiedName,
7. in DOMString publicId,
8. in DOMString systemId)
9. raises(DOMException);
10. // Introduced in DOM Level 2:
11. Document createDocument(in DOMString namespaceURI,
12. in DOMString qualifiedName,
13. in DocumentType doctype)
14. raises(DOMException);
15. };

*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:

1. hasFeature
2. Test if the DOM implementation implements a specific feature.
3. Parameters
4. feature of type DOMString
5. The name of the feature to test (case-insensitive). The values used by DOM features
6. are defined throughout the DOM Level 2 specifications and listed in the Conformance
7. section. The name must be an XML name. To avoid possible conflicts, as a
8. convention, names referring to features defined outside the DOM specification
9. should be made unique by reversing the name of the Internet domain name of the
10. person (or the organization that the person belongs to) who defines the feature,
11. component by component, and using this as a prefix. For instance, the W3C SVG
12. Working Group defines the feature "org.w3c.dom.svg".
13. version of type DOMString
14. This is the version number of the feature to test. In Level 2, the string can be
15. either "2.0" or "1.0". If the version is not specified, supporting any version of
16. the feature causes the method to return true.
17. Return Value
18. boolean
19. true if the feature is implemented in the specified version, false otherwise.
21. 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:

1. IDL Definition
2. interface DocumentFragment : Node {
3. };

*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:

1. IDL Definition
2. interface Document : Node {
3. readonly attribute DocumentType doctype;
4. readonly attribute DOMImplementation implementation;
5. readonly attribute Element documentElement;
6. Element createElement(in DOMString tagName)raises(DOMException);
7. DocumentFragment createDocumentFragment();
8. Text createTextNode(in DOMString data);
9. Comment createComment(in DOMString data);
10. CDATASection createCDATASection(in DOMString data)raises(DOMException);
11. ProcessingInstruction createProcessingInstruction(in DOMString target, in DOMString data)raises(DOMException);
12. Attr createAttribute(in DOMString name)raises(DOMException);
13. EntityReference createEntityReference(in DOMString name)raises(DOMException);
14. NodeList getElementsByTagName(in DOMString tagname);
15. // Introduced in DOM Level 2:
16. Node importNode(in Node importedNode, in boolean deep)raises(DOMException);
17. // Introduced in DOM Level 2:
18. Element createElementNS(in DOMString namespaceURI, in DOMString qualifiedName)raises(DOMException);
19. // Introduced in DOM Level 2:
20. Attr createAttributeNS(in DOMString namespaceURI, in DOMString qualifiedName)raises(DOMException);
21. // Introduced in DOM Level 2:
22. NodeList getElementsByTagNameNS(in DOMString namespaceURI, in DOMString localName);
23. // Introduced in DOM Level 2:
24. Element getElementById(in DOMString elementId);
25. };

*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:

1. Interface Document
2. Methods
3. createAttribute
4. Creates an Attr of the given name. Note that the Attr instance can then be set on
5. an Element using the setAttributeNode method.
6. To create an attribute with a qualified name and namespace URI, use the
7. createAttributeNS method.
8. Parameters
9. name of type DOMString
10. The name of the attribute.
11. Return Value
12. Attr A new Attr object with the nodeName attribute set to name, and localName,
13. prefix, and namespaceURI set to null. The value of the attribute is the empty
14. string.
15. Exceptions
16. DOMException INVALID\_CHARACTER\_ERR: Raised if the specified name contains an
17. 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:

1. Interface Document
2. Method
3. createCDATASection
4. Creates a CDATASection node whose value is the specified string.
5. Parameters
6. data of type DOMString
7. The data for the CDATASection contents.
8. Return Value
9. CDATASection The new CDATASection object.
10. Exceptions
11. 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:

1. Interface Document
2. Methods
3. createDocumentFragment
4. Creates an empty DocumentFragment object.
5. Return Value
6. DocumentFragment A new DocumentFragment.
7. No Parameters
8. 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:

1. Interface Document
2. Methods
3. createElement
4. Creates an element of the type specified. Note that the instance returned
5. implements the Element interface, so attributes can be specified directly on the
6. returned object.In addition, if there are known attributes with default values,
7. Attr nodes representing them are automatically created and attached to the
8. element.To create an element with a qualified name and namespace URI, use the
9. createElementNS method.
10. Parameters
11. tagName of type DOMString
12. The name of the element type to instantiate. For XML, this is case-sensitive. For
13. HTML, the tagName parameter may be provided in any case, but it must be mapped to
14. the canonical uppercase form by the DOM implementation.
15. Return Value
16. Element A new Element object with the nodeName attribute set to tagName, and
17. localName, prefix, and namespaceURI set to null.
18. Exceptions
19. DOMException INVALID\_CHARACTER\_ERR: Raised if the specified name contains an
20. 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:

1. Interface Document
2. Method
3. getElementById introduced in DOM Level 2
4. Returns the Element whose ID is given by elementId. If no such element exists,
5. returns null. Behavior is not defined if more than one element has this ID.
6. Note: The DOM implementation must have information that says which attributes are
7. of type ID. Attributes with the name "ID" are not of type ID unless so defined.
8. Implementations that do not know whether attributes are of type ID or not are
9. expected to return null.
10. Parameters
11. elementId of type DOMString
12. The unique id value for an element.
13. Return Value
14. Element The matching element.
15. 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:

1. IDL Definition
2. interface Node {
3. // NodeType
4. const unsigned short ELEMENT\_NODE = 1;
5. const unsigned short ATTRIBUTE\_NODE = 2;
6. const unsigned short TEXT\_NODE = 3;
7. const unsigned short CDATA\_SECTION\_NODE = 4;
8. const unsigned short ENTITY\_REFERENCE\_NODE = 5;
9. const unsigned short ENTITY\_NODE = 6;
10. const unsigned short PROCESSING\_INSTRUCTION\_NODE = 7;
11. const unsigned short COMMENT\_NODE = 8;
12. const unsigned short DOCUMENT\_NODE = 9;
13. const unsigned short DOCUMENT\_TYPE\_NODE = 10;
14. const unsigned short DOCUMENT\_FRAGMENT\_NODE = 11;
15. const unsigned short NOTATION\_NODE = 12;
16. readonly attribute DOMString nodeName;
17. attribute DOMString nodeValue;
18. // raises(DOMException) on setting
19. // raises(DOMException) on retrieval
20. readonly attribute unsigned short nodeType;
21. readonly attribute Node parentNode;
22. readonly attribute NodeList childNodes;
23. readonly attribute Node firstChild;
24. readonly attribute Node lastChild;
25. readonly attribute Node previousSibling;
26. readonly attribute Node nextSibling;
27. readonly attribute NamedNodeMap attributes;
28. // Modified in DOM Level 2:
29. readonly attribute Document ownerDocument;
30. Node insertBefore(in Node newChild,
31. in Node refChild)
32. raises(DOMException);
33. Node replaceChild(in Node newChild,
34. in Node oldChild)
35. raises(DOMException);
36. Node removeChild(in Node oldChild)
37. raises(DOMException);
38. Node appendChild(in Node newChild)
39. raises(DOMException);
40. boolean hasChildNodes();
41. Node cloneNode(in boolean deep);
42. // Modified in DOM Level 2:
43. void normalize();
44. // Introduced in DOM Level 2:
45. boolean isSupported(in DOMString feature,
46. in DOMString version);
47. // Introduced in DOM Level 2:
48. readonly attribute DOMString namespaceURI;
49. // Introduced in DOM Level 2:
50. attribute DOMString prefix;
51. // raises(DOMException) on setting
52. // Introduced in DOM Level 2:
53. readonly attribute DOMString localName;
54. // Introduced in DOM Level 2:
55. boolean hasAttributes();
56. };

*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:

1. The values of nodeName, nodeValue, and attributes vary according to the node type as follows:
2. Interface nodeName nodeValue attributes
3. Attr name of attribute value of attribute null
4. CDATASection #cdata-section content of the CDATA Section null
5. Comment #comment content of the comment null
6. Document #document null null
7. DocumentFragment #document-fragment null null
8. DocumentType document type name null null
9. Element tag name null NamedNodeMap
10. Entity entity name null null
11. EntityReference name of entity null null
12. referenced
13. Notation notation name null null
14. ProcessingInstruction target entire content excluding null
15. the target
16. 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:

1. Interface Node
2. Attributes
3. childNodes of type NodeList, readonly
4. A NodeList that contains all children of this node. If there are no children, this
5. 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:

1. Interface Node
2. The values of nodeName, nodeValue, and attributes vary according to the node type
3. Attributes
4. nodeName of type DOMString, readonly
5. 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:

1. Interface Node
2. Attributes
3. parentNode of type Node, readonly
4. The parent of this node. All nodes, except Attr, Document, DocumentFragment,
5. Entity, and Notation may have a parent. However, if a node has just been created
6. and not yet added to the tree, or if it has been removed from the tree, this is
7. 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:

1. Interface Node
2. Method
3. appendChild
4. Adds the node newChild to the end of the list of children of this node. If the
5. newChild is already in the tree, it is first removed.
6. Parameters
7. newChild of type Node
8. The node to add.If it is a DocumentFragment object, the entire contents of the
9. document fragment are moved into the child list of this node
10. Return Value
11. Node The node added.
12. Exceptions
13. DOMException HIERARCHY\_REQUEST\_ERR: Raised if this node is of a type that does not
14. allow children of the type of the newChild node, or if the node to append is one of
15. this node's ancestors.
16. WRONG\_DOCUMENT\_ERR: Raised if newChild was created from a different document than
17. the one that created this node.
18. 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:

1. Interface Node
2. Method
3. cloneNode
4. Returns a duplicate of this node, i.e., serves as a generic copy constructor for
5. nodes. The duplicate node has no parent; (parentNode is null.).Cloning an Element
6. copies all attributes and their values, including those generated by the XML
7. processor to represent defaulted attributes, but this method does not copy any text
8. it contains unless it is a deep clone, since the text is contained in a child Text
9. node.
10. Cloning an Attribute directly, as opposed to be cloned as part of an Element
11. cloning operation, returns a specified attribute (specified is true). Cloning any
12. other type of node simply returns a copy of this node.
13. Note that cloning an immutable subtree results in a mutable copy, but the children
14. of an EntityReference clone are readonly. In addition, clones of unspecified Attr
15. nodes are specified. And, cloning Document, DocumentType, Entity, and Notation
16. nodes is implementation dependent.
17. Parameters
18. deep of type boolean
19. If true, recursively clone the subtree under the specified node; if false, clone
20. only the node itself (and its attributes, if it is an Element).
21. Return Value
22. Node The duplicate node.
23. 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:

1. Interface Node
2. Method
3. insertBefore
4. Inserts the node newChild before the existing child node refChild. If refChild is
5. null, insert newChild at the end of the list of children.If newChild is a
6. DocumentFragment object, all of its children are inserted, in the same order,
7. before refChild. If the newChild is already in the tree, it is first removed.
8. Parameters
9. newChild of type Node
10. The node to insert.
11. refChild of type Node
12. The reference node, i.e., the node before which the new node must be inserted.
13. Return Value
14. Node The node being inserted.
15. Exceptions
16. 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.
17. WRONG\_DOCUMENT\_ERR: Raised if newChild was created from a different document than the one that created this node.
18. NO\_MODIFICATION\_ALLOWED\_ERR: Raised if this node is readonly or if the parent of the node being inserted is readonly.
19. 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:

1. Interface Node
2. Method
3. normalize modified in DOM Level 2
4. Puts all Text nodes in the full depth of the sub-tree underneath this Node,
5. including attribute nodes, into a "normal" form where only structure (e.g.,
6. elements, comments, processing instructions, CDATA sections, and entity references)
7. separates Text nodes, i.e., there are neither adjacent Text nodes nor empty Text
8. nodes. This can be used to ensure that the DOM view of a document is the same as if
9. it were saved and re-loaded, and is useful when operations (such as XPointer
10. [[XPointer]](https://go.microsoft.com/fwlink/?LinkID=183836&clcid=0x409) lookups) that depend on a particular document tree structure are to be
11. used.
12. Note: In cases where the document contains CDATASections, the normalize operation
13. alone may not be sufficient, since XPointers do not differentiate between Text
14. 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:

1. Interface Node
2. Method
3. removeChild Removes the child node indicated by oldChild from the list of children,
4. and returns it.
5. Parameters
6. oldChild of type Node
7. The node being removed.
8. Return Value
9. Node The node removed.
10. Exceptions
11. DOMException NO\_MODIFICATION\_ALLOWED\_ERR: Raised if this node is readonly.
12. 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:

1. Interface Node
2. Method
3. replaceChild
4. Replaces the child node oldChild with newChild in the list of children, and returns
5. the oldChild node.
6. If newChild is a DocumentFragment object, oldChild is replaced by all of the
7. DocumentFragment children, which are inserted in the same order. If the newChild is
8. already in the tree, it is first removed.
9. Parameters
10. newChild of type Node
11. The new node to put in the child list.
12. oldChild of type Node
13. The node being replaced in the list.
14. Return Value
15. Node The node replaced.
16. Exceptions
17. DOMException HIERARCHY\_REQUEST\_ERR: Raised if this node is of a type that does not
18. allow children of the type of the newChild node, or if the node to put in is one of
19. this node's ancestors.
20. WRONG\_DOCUMENT\_ERR: Raised if newChild was created from a different document than
21. the one that created this node.
22. NO\_MODIFICATION\_ALLOWED\_ERR: Raised if this node or the parent of the new node is
23. readonly.
24. 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:

1. interface NamedNodeMap {
2. Node getNamedItem(in DOMString name);
3. Node setNamedItem(in Node arg)
4. raises(DOMException);
5. Node removeNamedItem(in DOMString name)
6. raises(DOMException);
7. Node item(in unsigned long index);
8. readonly attribute unsigned long length;
9. // Introduced in DOM Level 2:
10. Node getNamedItemNS(in DOMString namespaceURI,
11. in DOMString localName);
12. // Introduced in DOM Level 2:
13. Node setNamedItemNS(in Node arg)
14. raises(DOMException);
15. // Introduced in DOM Level 2:
16. Node removeNamedItemNS(in DOMString namespaceURI,
17. in DOMString localName)
18. raises(DOMException);
19. };

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

The following methods are not supported:

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

V0026:

The specification states:

1. Interface NamedNodeMap
2. Method
3. getNamedItem
4. Retrieves a node specified by name.
5. Parameters
6. name of type DOMString
7. The nodeName of a node to retrieve.
8. Return Value
9. Node A Node (of any type) with the specified nodeName, or null if it does not
10. identify any node in this map.
11. 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:

1. Interface NamedNodeMap
2. Method
3. item
4. Returns the indexth item in the map. If index is greater than or equal to the
5. number of nodes in this map, this returns null.
6. Parameters
7. index of type unsigned long
8. Index into this map.
9. Return Value
10. Node The node at the indexth position in the map, or null if that is not a valid
11. index.
12. 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:

1. Interface NamedNodeMap
2. Method
3. removeNamedItem
4. Removes a node specified by name. When this map contains the attributes attached to
5. an element, if the removed attribute is known to have a default value, an attribute
6. immediately appears containing the default value as well as the corresponding
7. namespace URI, local name, and prefix when applicable.
8. Parameters
9. name of type DOMString
10. The nodeName of the node to remove.
11. Return Value
12. Node The node removed from this map if a node with such a name exists.
13. Exceptions
14. DOMException NOT\_FOUND\_ERR: Raised if there is no node named name in this map.
15. 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:

1. IDL Definition
2. interface Attr : Node {
3. readonly attribute DOMString name;
4. readonly attribute boolean specified;
5. attribute DOMString value;
6. // raises(DOMException) on setting
7. // Introduced in DOM Level 2:
8. readonly attribute Element ownerElement;
9. };

*Quirks Mode and IE7 Mode (All Versions)*

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

V0032:

The specification states:

1. Interface Attr
2. Attribute
3. specified of type boolean, readonly
4. If this attribute was explicitly given a value in the original document, this is
5. true; otherwise, it is false. Note that the implementation is in charge of this
6. attribute, not the user. If the user changes the value of the attribute (even if it
7. ends up having the same value as the default value) then the specified flag is
8. automatically flipped to true. To re-specify the attribute as the default value
9. from the DTD, the user must delete the attribute. The implementation will then make
10. a new attribute available with specified set to false and the default value (if one
11. exists).
12. In summary:
13. • If the attribute has an assigned value in the document then specified is true,
14. and the value is the assigned value.
15. • If the attribute has no assigned value in the document and has a default value in
16. the DTD, then specified is false, and the value is the default value in the DTD.
17. • If the attribute has no assigned value in the document and has a value of
18. #IMPLIED in the DTD, then the attribute does not appear in the structure model of
19. the document.
20. • If the ownerElement attribute is null (i.e. because it was just created or was
21. 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:

1. Interface Element
2. The Element interface represents an element in an HTML or XML document. Elements
3. may have attributes associated with them; since the Element interface inherits from
4. Node, the generic Node interface attribute attributes may be used to retrieve the
5. set of all attributes for an element. There are methods on the Element interface to
6. retrieve either an Attr object by name or an attribute value by name. In XML, where
7. an attribute value may contain entity references, an Attr object should be
8. retrieved to examine the possibly fairly complex sub-tree representing the
9. attribute value. On the other hand, in HTML, where all attributes have simple
10. string values, methods to directly access an attribute value can safely be used as
11. a convenience.
12. Note: In DOM Level 2, the method normalize is inherited from the Node interface
13. where it was moved.

*All Document Modes (All Versions)*

Attribute subtrees are not supported; only strings are supported.

V0034:

The specification states:

1. Interface Element
2. Method
3. getAttribute
4. Retrieves an attribute value by name.
5. Parameters
6. name of type DOMString
7. The name of the attribute to retrieve.
8. Return Value
9. DOMString The Attr value as a string, or the empty string if that attribute
10. does not have a specified or default value.
11. 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:

1. Interface Element
2. Method
3. getAttributeNode
4. Retrieves an attribute node by name.To retrieve an attribute node by qualified name
5. and namespace URI, use the getAttributeNodeNS method.
6. Parameters
7. name of type DOMString
8. The name (nodeName) of the attribute to retrieve.
9. Return Value
10. Attr The Attr node with the specified name (nodeName) or null if there is no such
11. attribute.
12. 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:

1. Interface Element
2. Method
3. getElementsByTagName
4. Returns a NodeList of all descendant Elements with a given tag name, in the order
5. in which they are encountered in a preorder traversal of this Element tree.
6. Parameters
7. name of type DOMString
8. The name of the tag to match on. The special value "\*" matches all tags.
9. Return Value
10. NodeList A list of matching Element nodes.
11. 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:

1. Interface Element
2. Method
3. removeAttribute
4. Removes an attribute by name. If the removed attribute is known to have a default
5. value, an attribute immediately appears containing the default value as well as the
6. corresponding namespace URI, local name, and prefix when applicable.
7. To remove an attribute by local name and namespace URI, use the removeAttributeNS
8. method.
9. Parameters
10. name of type DOMString
11. The name of the attribute to remove.
12. Exceptions
13. DOMException NO\_MODIFICATION\_ALLOWED\_ERR: Raised if this node is readonly.
14. 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:

1. Interface Element
2. Method
3. removeAttributeNode
4. Removes the specified attribute node. If the removed Attr has a default value it is
5. immediately replaced. The replacing attribute has the same namespace URI and local
6. name, as well as the original prefix, when applicable.
7. Parameters
8. oldAttr of type Attr
9. The Attr node to remove from the attribute list.
10. Return Value
11. Attr The Attr node that was removed.
12. Exceptions
13. DOMException
14. NO\_MODIFICATION\_ALLOWED\_ERR: Raised if this node is readonly.
15. 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:

1. Interface Element
2. setAttribute
3. Adds a new attribute. If an attribute with that name is already present in the
4. element, its value is changed to be that of the value parameter. This value is a
5. simple string; it is not parsed as it is being set. So any markup (such as syntax
6. to be recognized as an entity reference) is treated as literal text, and needs to
7. be appropriately escaped by the implementation when it is written out. In order to
8. assign an attribute value that contains entity references, the user must create an
9. Attr node plus any Text and EntityReference nodes, build the appropriate subtree,
10. and use setAttributeNode to assign it as the value of an attribute.
11. To set an attribute with a qualified name and namespace URI, use the setAttributeNS
12. method.
13. Parameters
14. name of type DOMString
15. The name of the attribute to create or alter.
16. value of type DOMString
17. Value to set in string form.
18. Exceptions
19. DOMException
20. INVALID\_CHARACTER\_ERR: Raised if the specified name contains an illegal character.
21. 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:

1. Interface Text
2. Method
3. splitText
4. Breaks this node into two nodes at the specified offset, keeping both in the tree
5. as siblings. After being split, this node will contain all the content up to the
6. offset point. A new node of the same type, which contains all the content at and
7. after the offset point, is returned. If the original node had a parent node, the
8. new node is inserted as the next siblings of the original node. When the offset is
9. equal to the length of this node, the new node has no data.
10. Parameters
11. offset of type unsigned long
12. The 16-bit unit offset at which to split, starting from 0.
13. Return Value
14. Text The new node, of the same type as this node.
15. Exceptions
16. DOMException INDEX\_SIZE\_ERR: Raised if the specified offset is negative or greater
17. than the number of 16-bit units in data.
18. 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:

1. Interface Comment
2. This interface inherits from CharacterData and represents the content of a comment,
3. i.e., all the characters between the starting '<!--' and ending '-->'. Note that
4. this is the definition of a comment in XML, and, in practice, HTML, although some
5. HTML tools may implement the full SGML comment structure.
6. IDL Definition
7. interface Comment : CharacterData {
8. };

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

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

V0029:

The specification states:

1. Interface NamedNodeMap
2. Method
3. setNamedItem
4. Adds a node using its nodeName attribute. If a node with that name is already
5. present in this map, it is replaced by the new one.
6. As the nodeName attribute is used to derive the name which the node must be stored
7. under, multiple nodes of certain types (those that have a "special" string value)
8. cannot be stored as the names would clash. This is seen as preferable to allowing
9. nodes to be aliased.
10. Parameters
11. arg of type Node
12. A node to store in this map. The node will later be accessible using the value of
13. its nodeName attribute.
14. Return Value
15. Node If the new Node replaces an existing node the replaced Node is returned,
16. otherwise null is returned.
17. Exceptions
18. DOMException WRONG\_DOCUMENT\_ERR: Raised if arg was created from a different
19. document than the one that created this map.
20. NO\_MODIFICATION\_ALLOWED\_ERR: Raised if this map is readonly.
21. INUSE\_ATTRIBUTE\_ERR: Raised if arg is an Attr that is already an attribute of
22. another Element object. The DOM user must explicitly clone Attr nodes to re-use
23. 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.

## Clarifications

The following subsections describe clarifications of the MAY and SHOULD requirements of [[DOM Level 2 - Core]](https://go.microsoft.com/fwlink/?LinkId=182703).

### [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:

1. Interface Document
2. Attributes
3. Doctype of type DocumentType, readonly
4. The Document Type Declaration (see DocumentType) associated with this document. For
5. HTML documents as well as XML documents without a document type declaration this
6. returns null. The DOM Level 2 does not support editing the Document Type
7. Declaration. docType cannot be altered in any way, including through the use of
8. 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:

1. Interface Document
2. Method
3. createComment
4. Creates a Comment node given the specified string.
5. Parameters
6. data of type DOMString
7. The data for the node.
8. Return Value
9. Comment The new Comment object.
10. 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:

1. Interface Document
2. Method
3. createTextNode
4. Creates a Text node given the specified string.
5. Parameters
6. data of type DOMString
7. The data for the node.
8. Return Value
9. Text The new Text object.
10. 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:

1. Attributes
2. length of type unsigned long, readonly
3. The number of 16-bit units that are available through data and the substringData
4. 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:

1. Interface Element
2. Attribute
3. tagName of type DOMString, readonly
4. The name of the element. For example, in:
5. <elementExample id="demo">
6. </elementExample> ,
7. tagName has the value "elementExample". Note that this is case-preserving in XML,
8. as are all of the operations of the DOM. The HTML DOM returns the tagName of an
9. HTML element in the canonical uppercase form, regardless of the case in the source
10. 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.

## Error Handling

There are no additional error handling considerations.

## Security

There are no additional security considerations.

# Change Tracking

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

# Index

A

Attribute

[setAttribute](#section_f2386301cab44d9fb6e4282d7b83520a) 7

Attributes

[ATTRIBUTE\_NODE](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[CDATA\_SECTION\_NODE](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[childNodes](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[COMMENT\_NODE](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[doctype](#section_61e955b5c3924ef786441d72677e4ced) 29

[DOCUMENT\_FRAGMENT\_NODE](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[DOCUMENT\_NODE](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[DOCUMENT\_TYPE\_NODE](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[ELEMENT\_NODE](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[ENTITY\_NODE](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[ENTITY\_REFERENCE\_NODE](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[localName](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[namespaceURI](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[nodeValue](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[NOTATION\_NODE](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[ownerElement](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[parentNode](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[prefix](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[PROCESSING\_INSTRUCTION\_NODE](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[TEXT\_NODE](#section_f2386301cab44d9fb6e4282d7b83520a) 7

C

[Change tracking](#section_c877820ed13344acb9725e2b6b150c71) 32

F

Fundamental Interfaces ([section 2.1.2](#section_f2386301cab44d9fb6e4282d7b83520a) 7, [section 2.2.1](#section_61e955b5c3924ef786441d72677e4ced) 29)

G

[Glossary](#section_3afe96402b6846f8b4c2733e1a6c702c) 4

I

[Informative references](#section_2e548446f2bd41cc8741c490a1fda286) 4

[Interfaces](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[Attr](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[CharacterData](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[Comment](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[Document](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[DocumentFragment](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[DOMImplementation](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[Element](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[Node](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[Text](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[Introduction](#section_3543bb545e544d3a9372a58ef238c58a) 4

M

Methods

[appendChild](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[cloneNode](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[createAttribute](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[createAttributeNS](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[createCDATASection](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[createComment](#section_61e955b5c3924ef786441d72677e4ced) 29

[createDocument](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[createDocumentFragment](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[createDocumentType](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[createElementNS](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[createEntityReference](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[createProcessingInstruction](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[createTextNode](#section_61e955b5c3924ef786441d72677e4ced) 29

[getAttributeNode](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[getElementById](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[getElementsByTagNameNS](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[getNamedItem](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[getNamedItemNS](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[hasAttributes](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[hasFeature](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[importNode](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[insertBefore](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[isSupported](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[item](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[normalize](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[removeAttribute](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[removeAttributeNode](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[removeChild](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[removeNamedItem](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[replaceChild](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[setNamedItem](#section_61e955b5c3924ef786441d72677e4ced) 29

[setNamedItemNS](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[splitText](#section_f2386301cab44d9fb6e4282d7b83520a) 7

[substringData](#section_f2386301cab44d9fb6e4282d7b83520a) 7

N

[Normative references](#section_03a02476490b416fa23c9929faaca508) 4

R

References

[informative](#section_2e548446f2bd41cc8741c490a1fda286) 4

[normative](#section_03a02476490b416fa23c9929faaca508) 4

T

[Tracking changes](#section_c877820ed13344acb9725e2b6b150c71) 32

X

[XML Namespaces](#section_d6ad7f2425f44ab0a36b32ddc08f413c) 7