[MS-DOM3C]:

Internet Explorer Document Object Model (DOM) Level 3 Core Standards Support Document

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	None	Introduced no new technical or language changes.
2/22/2012	3.0	Major	Significantly changed the technical content.
7/25/2012	3.1	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	5.2	Minor	Clarified the meaning of the technical content.
3/22/2016	5.2	None	No changes to the meaning, language, or formatting of the technical content.
7/19/2016	5.3	Minor	Clarified the meaning of the technical content.
11/2/2016	5.3	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 3 Core Specification Version 1.0* [DOM Level 3 - Core], published 7 April 2004.

The [DOM Level 3 - Core] specification may contain 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 3 - Core] W3C, "Document Object Model (DOM) Level 3 Core Specification Version 1.0", W3C Recommendation 07 April 2004, http://www.w3.org/TR/DOM-Level-3-Core/

[DOM Level 3 - LS] W3C, "Document Object Model (DOM) Level 3 Load and Save Specification Version 1.0", W3C Recommendation 07 April 2004, http://www.w3.org/TR/DOM-Level-3-LS/

[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 3 - 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 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 3 - 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 3 - 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.

For document mode and browser version notation, see section 1.3.

2 Standards Support Statements

This section contains all variations and clarifications for the Microsoft implementation of [DOM Level 3 - Core].

- Section 2.1 describes normative variations from the MUST requirements of the specification.
- Section 2.2 describes clarifications of the MAY and SHOULD requirements.
- Section 2.3 considers error handling aspects of the implementation.
- Section 2.4 considers security aspects of the implementation.

2.1 Normative Variations

The following subsections describe normative variations from the MUST requirements of [DOM Level 3 - Core].

2.1.1 [DOM Level 3 - Core] Section 1.3.1, String Comparisons in the DOM

V0001:

The specification states:

```
The character normalization, i.e. transforming into their fully normalized form as
as defined in [XML 1.1], is assumed to happen at serialization time. The DOM Level
3 Load and Save module [DOM Level 3 - LS] provides a serialization mechanism (see
the DOMSerializer interface, section 2.3.1) and uses the DOMConfiguration
parameters "normalize-characters" and "check-character-normalization" to assure
that text is fully normalized [XML 1.1]. Other serialization mechanisms built on
top of the DOM Level 3 Core also have to assure that text is fully normalized.
```

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

The DOM Level 3 Load and Save module [DOM Level 3 - LS] is not supported; therefore, this type of character normalization is not supported.

2.1.2 [DOM Level 3 - Core] Section 1.3.3, XML Namespaces

V0002:

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.
Consequently, moving a node within a document, using the DOM, in no case results in
a change of its namespace prefix or namespace URI. Similarly, creating a node with
a namespace prefix and namespace URI, or changing the namespace prefix of a node,
does not result in any addition, removal, or modification of any special attributes
for declaring the appropriate XML namespaces. Namespace validation is not enforced;
the DOM application is responsible. In particular, since the mapping between
prefixes and namespace URIs is not enforced, in general, the resulting document
cannot be serialized naively. For example, applications may have to declare every
namespace in use when serializing a document.
```

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

Special attributes for declaring namespaces that begin with "xmlns:" on the root <html> element are not supported. These attributes do not affect the namespace URI of created elements.

2.1.3 [DOM Level 3 - Core] Section 1.4, Fundamental Interfaces: Core Module

V0003:

The specification defines the **DOMException**.

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

The **DOMException** exception is not supported. JavaScript error objects are created when exceptions occur.

V0004:

The specification defines the **ExceptionCode** exception of the **DOMException.**

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

The **ExceptionCode** exception is not supported.

V0005:

The specification defines the **DOMStringList** interface.

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

The **DOMStringList** interface is not supported.

V0006:

The specification defines the **NameList** interface.

All Document Modes (All Versions)

The **NameList** interface is not supported.

V0007:

The specification defines the **DOMImplementationList** interface.

All Document Modes (All Versions)

The **DOMImplementationList** interface is not supported.

V0008:

The specification defines the **DOMImplementationSource** interface.

All Document Modes (All Versions)

The **DOMImplementationSource** interface is not supported.

V0009:

The specification states:

```
Interface DOMImplementation

The DOMImplementation interface provides a number of methods for performing
```

```
    operations that are independent of any particular instance of the document object
    model.
```

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

The following methods of the **DOMImplementation** interface are not supported and return a DOMException of **NOT_SUPPORTED_ERR**:

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

All Document Modes (All Versions)

The **getFeature** method of the **DOMImplementation** interface is not supported and returns a DOMException of **NOT_SUPPORTED_ERR**.

V0057:

The specification defines the **getFeature** method of the **Document** object.

All Document Modes (All Versions)

The **getFeature** method of the **DOMImplementation** interface is not supported and returns a DOMException of **NOT_SUPPORTED_ERR**.

V0010:

The specification states:

```
    Interface DocumentFragment

    DocumentFragment is a "lightweight" or "minimal" Document object. It is very common
    to want to be able to extract a portion of a document's tree or to create a new
    fragment of a document.

    Furthermore, various operations -- such as inserting nodes as children of another
    Node -- may take DocumentFragment objects as arguments; this results in all the
    child nodes of the DocumentFragment being moved to the child list of this node.
    The children of a DocumentFragment node are zero or more nodes representing the
    tops of any sub-trees defining the structure of the document. DocumentFragment
    nodes do not need to be well-formed XML documents (although they do need to follow
    the rules imposed upon well-formed XML parsed entities, which can have multiple top nodes).

    When a DocumentFragment is inserted into a Document (or indeed any other Node that
    may take children) the children of the DocumentFragment and not the
    DocumentFragment itself are inserted into the Node. This makes the DocumentFragment
    very useful when the user wishes to create nodes that are siblings; the
    DocumentFragment acts as the parent of these nodes so that the user can use the
    standard methods from the Node interface, such as Node.insertBefore and
    Node.appendChild.
```

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 it defines.

V0011:

The specification defines the **Document** interface.

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

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

- **inputEncoding**
- **xmlEncoding**
- **xmlStandalone**
- **xmlVersion**

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

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

All Document Modes (All Versions)

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

- **documentURI**
- **domConfig**
- **strictErrorChecking**

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

- **createEntityReference**
- **normalizeDocument**
- **renameNode**

V0012:

The specification states:

```
Interface DOMImplementation
The DOMImplementation interface provides a number of methods for performing
operations that are independent of any particular instance of the document object
model.
```

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

The **doctype** attribute of the **Document** interface is always NULL because the **DocumentType** interface is not supported.

V0013:

The specification defines the **DocumentURI** interface.

All Document Modes (All Versions)

The **DocumentURI** interface is not supported.

V0014:

The specification states:

```
getElementById introduced in DOM Level 2

Returns the Element that has an ID attribute with the given value. If no such
element exists, this returns null. If more than one element has an ID attribute
with that value, what is returned is undefined. The DOM implementation is expected
to use the attribute Attr.isId to determine if an attribute is of type ID.
Note: Attributes with the name "ID" or "id" are not of type ID unless so defined.

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

Return Value
Element The matching element or null if there is none.

No Exceptions
```

Quirks Mode and IE7 Mode (All Versions)

The **getElementById** method of the **Document** interface performs a case-insensitive compare against the IDs of elements and searches "name" attributes in addition to "id" attributes.

V0072:

The specification states:

```
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 is not an XML name
according to the XML version in use specified in the Document.xmlVersion attribute.
```

Quirks Mode, IE7 Mode, and IE8 Mode (Internet Explorer 7, Internet Explorer 8)

The created **Attr** instance has a **nodeValue** value of "undefined" instead of an empty string.

V0015:

The specification states:

```
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. The **createDocumentFragment** method returns a node-derived interface and inherits all the methods of the Document interface in addition to those on the **Node** interface. The **nodeType** and **nodeValue** attributes are correct on the returned object.

V0016:

The specification states:

```
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,
otherwise it depends on the case-sensitivity of the markup language in use. In that
case, the name is mapped to the canonical form of that markup 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 is not an XML name according to
the XML version in use specified in the Document.xmlVersion attribute.
```

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.

V0017:

The specification defines the **Node** interface.

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

The following definition groups of the **Node** interface are not supported:

- DocumentPosition
- NodeType

With **NodeType** definition group values, the actual document types in markup are created as instances of the **Comment** interface, causing the values of the instance to match those of the **Comment** entry.

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

- **baseURI**
- **localName**
- **namespaceURI**
- **prefix**
- **textContent**

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

- **compareDocumentPosition**
- **getFeature**
- **getUserData**
- **isDefaultNamespace**
- **isEqualNode**
- **isSameNode**
- **isSupported**
- **lookupNamespaceURI**
- **lookupPrefix**
- **setUserData**

Quirks Mode, IE7 Mode, IE8 Mode, IE9 Mode, IE10 Mode, and IE11 Mode (All Versions)

The **baseURI** attribute of the **Node** interface is not supported.

All Document Modes (All Versions)

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

- **getFeature**
- **getUserData**
- **setUserData**

V0018:

The specification states:

```
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.
```

Internet Explorer 8 (All Modes)

Splitting multiple text nodes under an element using the **splitText** method can cause the **childNodes** collection to not be immediately updated. Other tree modifications cause the **childNodes** collection to resynchronize.

V0019:

The specification states:

```
Attributes

nodeName of type DOMString, readonlyThe 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, the **nodeName** attribute drops the element prefixes and does not return uppercase values.

V0020:

The specification states:

```
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)

With the **parentNode** attribute of the **Node** interface, when a parentless element has child nodes, a document fragment is created and set as the parent of that element.

V0021:

The specification states:

```
Method

appendChild modified in DOM Level 3 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 or this
node itself, or if this node is of type Document and the DOM application attempts to append a
second DocumentType or Element node.

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 previous parent of the
node being inserted is readonly.

NOT_SUPPORTED_ERR: if the newChild node is a child of the Document node, this exception might
be raised if the DOM implementation doesn't support the removal of the DocumentType child or
Element child.
```

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

With the **appendChild** method of the **Node** interface, the conditions that trigger the HIERARCHY_REQUEST_ERR and WRONG_DOCUMENT_ERR errors both result in a JScript "Invalid argument" error (HRESULT 0x80070057).

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

- **APPLET**
- **AREA**
- **BASE**
- **BGSOUND**
- **BR**
- **COL**
- **COMMENT**
- **EMBED**
- **FRAME**
- **HR**
- **IFRAME**
- **IMG**
- **INPUT**
- **ISINDEX**

- **LINK**
- **META**
- **NEXTID**
- **NOEMBED**
- **NOFRAMES**
- **NOSCRIPT**
- **OBJECT**
- **PARAM**
- **SCRIPT**
- **STYLE**
- **WBR**

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

With the **appendChild** method of the **Node** interface, the conditions that trigger the WRONG_DOCUMENT_ERR error cause the node to automatically be adopted and inserted. No exception is thrown.

V0022:

The specification states:

```
Method

cloneNode  Returns a duplicate of this node, i.e., serves as a generic copy
constructor for nodes.

Cloning an Attr directly, as opposed to be cloned as part of an Element cloning
operation, returns a specified attribute (specified is true). Cloning an Attr
always clones its children, since they represent its value, no matter whether this
is a deep clone or not. In addition, clones of unspecified Attr nodes are
specified. And, cloning Document, DocumentType, Entity, and Notation nodes is
implementation dependent.
```

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

With the **cloneNode** method of the **Node** interface, cloned **Attr** objects do not have the specified attribute set to `true`. Cloned **Attr** objects have their **nodeValue** attributes set to `undefined`.

V0023:

The specification states:

```
Methods

insertBefore modified in DOM Level 3
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.
```

```
Note: Inserting a node before itself is implementation dependent.

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 or this node itself, or if this node is of type Document and the
DOM application attempts to insert a second DocumentType or Element node.

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.

NOT_SUPPORTED_ERR: if this node is of type Document, this exception might be raised
if the DOM implementation doesn't support the insertion of a DocumentType or
Element node.
```

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

With the **insertBefore** method of the **Node** interface, the conditions to trigger the HIERARCHY_REQUEST_ERR and WRONG_DOCUMENT_ERR errors both result in an "Invalid argument" error (HRESULT 0x80070057).

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

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

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

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

With the **insertBefore** method of the **Node** interface, the conditions that trigger the WRONG_DOCUMENT_ERR error cause the node to automatically be adopted and inserted. No exception is thrown.

V0024:

The specification states:

```
normalize modified in DOM Level 3
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. If the parameter "normalize-characters" of the DOMConfiguration object
attached to the Node.ownerDocument is true, this method will also fully normalize
the characters of the Text nodes. 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.
No Parameters
No Return Value
No Exceptions
```

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

The **normalize** method of the **Node** interface does not collapse an empty text node into an adjacent text node. An empty text node is not removed if it is the only child of its parent.

V0025:

The specification states:

```
Method
removeChild modified in DOM Level 3 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.
NOT_SUPPORTED_ERR: if this node is of type Document, this exception might be raised
if the DOM implementation doesn't support the removal of the DocumentType child or
the Element child.
```

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

With the **removeChild** method of the **Node** interface, the conditions that trigger the NOT_FOUND_ERR error result in a JavaScript "Invalid argument" error (HRESULT 0x80070057).

V0026:

The specification states:

```
Method
replaceChild modified in DOM Level 3
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.Note: Replacing a node with
itself is implementation dependent.

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 or this node itself, or if this node is of type Document and the
result of the replacement operation would add a second DocumentType or Element on
the Document node.
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.
NOT_SUPPORTED_ERR: if this node is of type Document, this exception might be raised
if the DOM implementation doesn't support the replacement of the DocumentType child
or Element child.
```

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

With the **replaceChild** method of the **Node** interface, the conditions that trigger the HIERARCHY_REQUEST_ERR, WRONG_DOCUMENT_ERR, and NOT_FOUND_ERR errors result in JavaScript "Invalid argument" errors (HRESULT 0x80070057).

V0027:

The specification defines the **NamedNodeMap** interface.

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

The following methods are not supported by the **NamedNodeMap** interface:

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

V0028:

The specification states:

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

Parameters
index of type unsigned longIndex 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)

With the **item** method of the **NamedNodeMap** interface, NULL is not returned when the index is greater than the number of nodes in the map, but instead results in an HRESULT 0x80070057, which creates a JavaScript Error message "Invalid argument."

V0029:

The specification states:

```
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 DOMStringThe 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)

With the **removeNamedItem** method of the **NamedNodeMap** interface, NULL is returned instead of the NOT_FOUND_ERR exception.

V0030:

The specification states:

```
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. Replacing a node by
itself has no effect.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.
HIERARCHY_REQUEST_ERR: Raised if an attempt is made to add a node doesn't belong in
this NamedNodeMap. Examples would include trying to insert something other than an
Attr node into an Element's map of attributes, or a non-Entity node into the
DocumentType's map of Entities.
```

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

With the **setNamedItem** method of the **NamedNodeMap** interface, an "Invalid Argument" exception is raised instead of INUSE_ATTRIBUTE_ERR. No exception is raised when the argument was created from a different document or when it is not an attribute.

V0031:

The specification states:

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

Parameters
offset of type unsigned longStart offset of substring to extract.
count of type unsigned longThe number of 16-bit units to extract.

Return Value DOMString The specified substring. If the sum of offset and count
exceeds the
length, then all 16-bit units to the end of the data are returned.

Exceptions
DOMException
INDEX_SIZE_ERR: Raised if the specified offset is negative or greater than the
number of 16-bit units in data, or if the specified count is negative.
DOMSTRING_SIZE_ERR: Raised if the specified range of text does not fit into a
DOMString.
```

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

With the **substringData** method of the **CharacterData** interface, an exception is not thrown if the offset is greater than the number of 16-bit units in the data. An "Invalid argument" error is returned instead of a DOMException.

V0032:

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;
  // Introduced in DOM Level 3:
  readonly attribute TypeInfo        schemaTypeInfo;
  // Introduced in DOM Level 3:
  readonly attribute boolean         isId;
};
```

All Document Modes (All Versions)

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

- **schemaTypeInfo**
- **isId**

V0033:

The specification defines the **Element** interface.

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

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

- **getAttributeNodeNS**
- **getAttributeNS**
- **hasAttributeNS**
- **removeAttributeNS**
- **schemaTypeInfo**
- **setAttributeNodesNS**
- **setAttributeNS**
- **setIdAttribute**
- **setIdAttributeNode**
- **setIdAttributeNS**

Attribute subtrees of the **Attr** interface are not supported. Only strings are supported for the **Attr** object.

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

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

V0034:

The specification states:

```
tagName of type DOMString, readonly
The name of the element. If Node.localName is different from null, this attribute
is a qualified name. 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** attribute of the **Element** interface returns uppercase values except for elements with names that resemble namespaces (such as `<test:elementName>`) when a proprietary namespace has been declared because XML support is provided through MSXML. In this case, the **tagName** attribute drops the element prefixes and does not return uppercase values.

V0035:

The specification states:

```
Method
getElementsByTagName  Returns a NodeList of all descendant Elements with a given
tag name, in document order.

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)

With the **getElementsByTagName** method of the **Document** interface, if **object1.getElementsByTagName**("*") is called, an empty collection is returned. If **object1.getElementsByTagName**("param") is called, a collection containing all parameters in the document is returned, as if the call were **document.getElementsByTagName**("param").

V0036:

The specification states:

```
Method
removeAttribute  Removes an attribute by name. If a default value for the removed
attribute is defined in the DTD, a new attribute immediately appears with the
default value as well as the corresponding namespace URI, local name, and prefix
when applicable. The implementation may handle default values from other schemas
similarly but applications should use Document.normalizeDocument() to guarantee
this information is up-to-date.If no attribute with this name is found, this method
has no effect.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 **removeAttribute** method supports an additional parameter called **iCaseSensitive**. The **iCaseSensitive** parameter specifies whether to use a case-sensitive search to find the attribute. Removal of event handler attributes (such as **onClick**) or the style attribute does not cause the actual event handler to be removed, or the inline style to be removed.

Default attributes are not re-created after the attribute is removed.

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

The **removeAttributeNS** method is not supported.

Quirks Mode, IE7 Mode, IE8 Mode, IE9 Mode, IE10 Mode, and IE11 Mode (All Versions)

The **removeAttribute** method of the **Element** interface exists on **CSSStyleDeclaration** objects that are used for inline styles, runtime styles, and style rules.

The **removeAttribute** method returns a Boolean value that indicates whether the operation has succeeded or failed. The **removeAttribute** method is also available on the following objects: **element.currentStyle**, **element.runtimeStyle**, **element.style**, and **stylesheet.style**.

V0037:

The specification states:

```
Method
removeAttributeNode  Removes the specified attribute node. If a default value for
the removed Attr node is defined in the DTD, a new node immediately appears with
the default value as well as the corresponding namespace URI, local name, and
prefix when applicable. The implementation may handle default values from other
schemas similarly but applications should use Document.normalizeDocument() to
guarantee this information is up-to-date.

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

Return Value
Attr  The Attr node that was removed.

Exceptions
DOMExceptionNO_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)

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 the style attribute does not cause the actual event handler to be removed, or the inline style to be removed.

V0038:

The specification states:

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

Exceptions
DOMException  INVALID_CHARACTER_ERR: Raised if the specified name is not an XML
name according to the XML version in use specified in the Document.xmlVersion
attribute.
NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly.

No Return Value
```

Quirks Mode and IE7 Mode (All Versions)

The **setAttribute** method assigns attributes in a case-sensitive manner (expected case-insensitive for HTML).

The second parameter of the **setAttribute** method accepts only strings, not objects. An optional third parameter controls case sensitivity.

Attributes that apply a boolean initial state to the associated DOM properties (for example, **value** and **checked**) are incorrectly associated with their live property rather than their default property. For example, the **setAttribute** method ('checked', 'checked') toggles the DOM **checked** property (the live view of a check box) rather than the **defaultChecked** property (initial value).

The HTML **style** attribute and attributes that are event handlers do not apply their conditions when used with **setAttribute**.

The **setAttribute** method requires DOM property names to apply effects for certain attribute names, such as **className** (instead of 'class'), **htmlFor** (instead of 'for'), and **httpEquiv** (instead of 'http-equiv').

V0039:

The specification defines the **Text** interface.

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

The following attribute and method of the **Text** interface are not supported:

- **wholeText** attribute
- **replaceWholeText** method

All Document Modes (All Versions)

The **isElementContentWhitespace** attribute of the **Text** interface is not supported.

V0040:

The specification states:

```
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 sibling 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 (Internet Explorer 8)

With the **splitText** method of the **Text** interface, any time there is a modification to the markup, a cache of invalidated **childNodes** objects is maintained. Because calling the **splitText** method does not trigger a markup modification, the **childNodes** collection does not show changes made by **splitText** until the markup is modified (for example, by changing the text of a **DIV** element anywhere on the page).

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

The **splitText** method treats the argument as optional; if there is no unsigned long offset provided, then and offset of zero (0) will be 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
```

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

With the **Comment** interface, **comments** are derived from **ELEMENT** rather than from **Node**.

V0042:

The specification defines the **TypeInfo** interface.

All Document Modes (All Versions)

The **TypeInfo** interface is not supported.

V0043:

The specification defines the **UserDataHandler** interface

All Document Modes (All Versions)

The **UserDataHandler** interface is not supported.

V0044:

The specification defines the **DOMError** interface.

IE7 Mode, IE8 Mode, IE9 Mode, and IE10 Mode (All Versions)

The **DOMError** interface is not supported.

V0045:

The specification defines the **DOMErrorHandler** interface.

All Document Modes (All Versions)

The **DOMErrorHandler** interface is not supported.

V0046:

The specification defines the **DOMLocator** interface.

All Document Modes (All Versions)

The **DOMLocator** interface is not supported.

V0047:

The specification defines the **DOMConfiguration** interface.

All Document Modes (All Versions)

The **DOMConfiguration** interface is not supported.

2.1.4 [DOM Level 3 - Core] Section 1.5, Extended Interfaces: XML Module

V0048:

The specification states:

```
In order to fully support this module, an implementation must also support the
"Core" feature defined in Fundamental Interfaces: Core Module and the feature
"XMLVersion" with version "1.0" defined inDocument.xmlVersion.
```

All Document Modes (All Versions)

The **XML Module** extended interface is not supported because Core Module methods such as **getUserData** are not implemented. Additionally, if the **hasFeature** method is run with **XMLVersion** equal to "`1.0`", a value of **false** is returned.

2.2 Clarifications

The following subsections describe clarifications of the MAY and SHOULD requirements of [DOM Level 3 - Core].

2.2.1 [DOM Level 3 - Core] Section 1.4, Fundamental Interfaces: Core Module

C0001:

The specification states:

```
Method
hasFeature Test if the DOM implementation implements a specific feature and
version, as specified in DOM
```

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", "2.0", and "3.0".

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

The **hasFeature** method of the **DOMImplementation** interface returns `false` for the Core module and the version string "3.0" because not all DOM L3 Core methods are supported.

C0002:

The specification states:

```
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.
```

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

The **data** parameter of the **createTextNode** method of the **Document** interface is optional. A text node is created even when no parameter is provided.

C0003:

The specification states:

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

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.

C0004:

The specification states:

```
getAttribute Retrieves an attribute value by name.
Parameters name of type DOMStringThe name of the attribute to retrieve.
Return Value DOMStringThe 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 of the **Element** interface supports an additional parameter that controls case-sensitivity and object extrapolation (such as for the **style** object).

C0005:

The specification states:

```
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)

The **getAttributeNode** method of the **Element** interface returns attribute nodes that have not been specified by the author or have default values other than **NULL**

2.3 Error Handling

There are no additional error handling considerations.

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