[MS-DOM2CE]:
Internet Explorer Extensions to the Document Object Model (DOM) Level 2 Core Specification

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft’s delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs.** To see all of the protocols in scope under a specific license program and the associated patents, visit the Patent Map.
- **Trademarks.** The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names.** The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

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

**Support.** For questions and support, please contact dochelp@microsoft.com.
## Revision Summary

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision History</th>
<th>Revision Class</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>5/26/2010</td>
<td>1.2</td>
<td>None</td>
<td>Introduced no new technical or language changes.</td>
</tr>
<tr>
<td>9/8/2010</td>
<td>1.3</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>10/13/2010</td>
<td>1.4</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>2/10/2011</td>
<td>2.0</td>
<td>None</td>
<td>Introduced no new technical or language changes.</td>
</tr>
<tr>
<td>2/22/2012</td>
<td>3.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>7/25/2012</td>
<td>3.1</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>6/26/2013</td>
<td>4.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>3/31/2014</td>
<td>4.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>1/22/2015</td>
<td>5.0</td>
<td>Major</td>
<td>Updated for new product version.</td>
</tr>
<tr>
<td>7/7/2015</td>
<td>5.1</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>11/2/2015</td>
<td>5.1</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>3/22/2016</td>
<td>5.2</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>11/2/2016</td>
<td>5.2</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>3/14/2017</td>
<td>5.2</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>10/3/2017</td>
<td>5.2</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>2/22/2018</td>
<td>5.2</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>3/23/2018</td>
<td>5.2</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>8/28/2018</td>
<td>5.2</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
</tbody>
</table>
Table of Contents

1 Introduction ......................................................................................................................... 4
  1.1 Glossary .......................................................................................................................... 4
  1.2 References ....................................................................................................................... 4
    1.2.1 Normative References ............................................................................................... 4
    1.2.2 Informative References ............................................................................................ 4
  1.3 Extension Overview (Synopsis) ....................................................................................... 5
    1.3.1 Organization of This Documentation ........................................................................ 6
  1.4 Relationship to Standards and Other Extensions ............................................................ 6
  1.5 Applicability Statement .................................................................................................... 6

2 Extensions .......................................................................................................................... 7
  2.1 Extensions to the Element Interface ................................................................................. 7
    2.1.1 Attributes .................................................................................................................. 8
      2.1.1.1 canHaveChildren ............................................................................................... 8
      2.1.1.2 canHaveHTML.................................................................................................... 8
      2.1.1.3 parentElement .................................................................................................... 8
      2.1.1.4 scopeName .......................................................................................................... 9
      2.1.1.5 sourceIndex ....................................................................................................... 9
      2.1.1.6 tagUrN ................................................................................................................ 9
    2.1.2 Methods ..................................................................................................................... 9
      2.1.2.1 applyElement ..................................................................................................... 9
      2.1.2.2 clearAttributes ................................................................................................. 10
      2.1.2.3 contains.............................................................................................................. 10
      2.1.2.4 insertAdjacentElement .................................................................................... 10
      2.1.2.5 mergeAttributes ............................................................................................... 11
      2.1.2.6 removeNode .................................................................................................... 12
      2.1.2.7 replaceNode .................................................................................................... 12
      2.1.2.8 swapNode ......................................................................................................... 12
    2.1.3 Collections ................................................................................................................. 13
      2.1.3.1 all ....................................................................................................................... 13
      2.1.3.2 children ............................................................................................................. 14
  2.2 Extensions to the Comment Interface .............................................................................. 15
    2.2.1 Attributes ................................................................................................................ 15
      2.2.1.1 text .................................................................................................................... 15
      2.2.1.2 atomic ............................................................................................................... 15
  2.3 Extensions to the Document Interface ............................................................................ 16
    2.3.1 Methods .................................................................................................................... 16
      2.3.1.1 clear ................................................................................................................ 16
      2.3.1.2 removeNode .................................................................................................... 17
      2.3.1.3 replaceNode .................................................................................................... 17
      2.3.1.4 swapNode ......................................................................................................... 17
  2.4 Extensions to the Text Interface ..................................................................................... 18
    2.4.1 Methods ................................................................................................................... 18
      2.4.1.1 removeNode .................................................................................................... 18
      2.4.1.2 replaceNode .................................................................................................... 18
      2.4.1.3 swapNode ......................................................................................................... 19

3 Security Considerations ...................................................................................................... 20

4 Appendix A: Product Behavior ............................................................................................ 21

5 Change Tracking .................................................................................................................. 22

6 Index ................................................................................................................................... 23
1 Introduction

This document describes extensions provided by Microsoft web browsers for the Document Object Model (DOM) Level 2 Core Specification Version 1.0 [DOM Level 2 - Core], published 13 November 2000.

Section 2 of this specification is normative. All other sections and examples in this specification are informative.

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.


1.2.2 Informative References


[MS-CSS21E] Microsoft Corporation, "Internet Explorer Extensions to Cascading Style Sheets (CSS) 2.1 and DOM Level 2 Style Specifications".

1.3 Extension Overview (Synopsis)

The extensions described in this document were selected for their applicability to [DOM Level 2 - Core].

The additional Document Object Model interfaces, attributes, methods and collections are organized based on section 1.2, Fundamental Interfaces, of [DOM Level 2 - Core] as follows:

**Element**

- **Attributes**
  - `canHaveChildren`
  - `canHaveHTML`
  - `parentElement`
  - `scopeName`
  - `sourceIndex`
  - `tagUrn`

- **Methods**
  - `applyElement`
  - `clearAttributes`
  - `contains`
  - `insertAdjacentElement`
  - `mergeAttributes`
  - `removeNode`
  - `replaceNode`
  - `swapNode`

- **Collections**
  - `all`
  - `children`

**Comment**

- **Attributes**
  - `text`
Document

- atomic

Methods
- clear
- removeNode
- replaceNode
- swapNode

Text

Methods
- removeNode
- replaceNode
- swapNode

1.3.1 Organization of This Documentation

This document is organized as follows:

- Interfaces: The extensions are listed according to interface at the highest level.
- Attributes, Methods, Collections: The interface members are described at the next levels.

1.4 Relationship to Standards and Other Extensions

The following documents provide information on additional extensions.

- [MS-CSS21E]: Extensions to the [CSS-Level2-2009] and [DOM Level 2 - Style] specifications.
- [MS-HTML401E]: Extensions to the [HTML] and the [DOM Level 2 - HTML] specifications.
- [MS-DOM2CEX]: Extensions to the [HTML] and the [DOM Level 2 - HTML] specifications for Microsoft XML Core Services.
- [MS-DOM2EE]: Extensions to the [MS-DOM2E] specifications.

1.5 Applicability Statement

This document specifies a set of extensions to the [DOM Level 2 - Core] specification. The extensions in this document provide access to some features that are unique to 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, and Microsoft Edge.
2 Extensions

This section specifies additional attributes and methods to elements from [DOM Level 2 - Core] that are available in Microsoft web browsers.

The extensions to [DOM Level 2 - Core] are as follows:

- Extensions to the Element Interface
- Extensions to the Comment Interface
- Extensions to the Document Interface
- Extensions to the Text Interface

2.1 Extensions to the Element Interface

Extensions have been added to the Element interface of the Document Object Model (DOM) Level 2 Core Specification [DOM Level 2 - Core]. The extensions of the Element interface are:

Attributes

The Elements interface is extended by the following attributes. For details, see Attributes.

- canHaveChildren
- canHaveHTML
- parentElement
- scopeName
- sourcexIndex
- tagUrn

Methods

The Elements interface is extended by the following methods. For details, see Methods.

- applyElement
- clearAttributes
- contains
- insertAdjacentElement
- mergeAttributes
- removeNode
- replaceNode
- swapNode

Collections

The Elements interface is extended by the following collections. For details, see Collections.

- all
The following IDL definition documents the **Element** interface:

```idl
text
interface Element : Node {
    // Extension of DOM Level 2:
    readonly attribute boolean canHaveChildren;
    readonly attribute boolean canHaveHTML;
    readonly attribute boolean parentElement;
    readonly attribute long sourceIndex;
    // Extension of DOM Level 2:
    applyElement void clearAttributes;
    boolean contains;
    insertAdjacentElement Node mergeAttributes;
    removeNode;
    replaceNode (in Node newNode) raises(DOMException);
    swapNode;
};
```

### 2.1.1 Attributes

The **Elements** interface as specified in the [DOM Level 2 Core](https://www.w3.org/TR/DOM-Level-2-Core/), is extended by the addition of the following attributes:

- **canHaveChildren**
- **canHaveHTML**
- **parentElement**
- **sourceIndex**

#### 2.1.1.1 canHaveChildren

**canHaveChildren** of type **boolean**, **read-only**

Gets a Boolean value indicating whether the object can contain child objects. Objects do not have to contain children for the **canHaveChildren** attribute to return **true**. The **canHaveChildren** attribute is useful in determining whether objects can be appended as children. The property has no default value.

#### 2.1.1.2 canHaveHTML

**canHaveHTML** of type **boolean**

Retrieves the Boolean value indicating whether the object can contain rich HTML markup. The property is read-only for all objects except the **defaults** object, which is read-write. The property has no default value.

#### 2.1.1.3 parentElement

**parentElement** of type **Element**, **read-only**

Retrieves the parent object in the object hierarchy. The highest object returns null as its parent. The property has no default value.
2.1.1.4 scopeName

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

**scopeName** of type String, read-only

Gets the namespace defined for the element. The property has a default value of HTML.

2.1.1.5 sourceIndex

**sourceIndex** of type long, read-only

Retrieves the ordinal position of the object, in source order, as the object appears in the document's **all** collection. The property has no default value.

2.1.1.6 tagUrn

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

**tagUrn** of type String

Sets or gets the Uniform Resource Name (URN) specified in the namespace declaration. The property has a default value of null.

2.1.2 Methods

The **Element** interface is extended by the addition of the following methods:

- **applyElement**
- **clearAttributes**
- **contains**
- **insertAdjacentElement**
- **mergeAttributes**
- **removeNode**
- **replaceNode**
- **swapNode**

2.1.2.1 applyElement

**applyElement**

The **applyElement** method makes the element either a child or parent of another element. The **applyElement** method is accessible at run time. If elements are removed at run time before the closing tag is parsed, areas of the document might not render.

**Parameters**

- **oNewElement** of type Element
  
  An object that becomes the child or parent of the current element.

- **sWhere** of type String
Optional. If **outside**, the specified element becomes parent of the current element. The default parameter is **outside**. If **inside**, the specified element becomes a child of the current element, but contains all the child elements of the current element.

**Return Value**

**Element** The applied element, **oNewElement**.

**JScript Error**

**E_INVALIDARG**: Raised if **oNewElement** is null, **oNewElement** is outside the root node, or if **oNewElement** is not owned by the current document.

### 2.1.2.2 clearAttributes

clearAttributes

The **clearAttributes** method removes attributes and values from the object. It clears persistent HTML attributes only; the ID attribute, styles, and script-only properties are not affected.

**Parameters**

None.

**Return Value**

None.

**No Jscript Error**

### 2.1.2.3 contains

contains

The **contains** method checks whether the specified element is contained within the object.

**Parameters**

**oElement**

Element object that specifies the element to check.

**Return Value**

Boolean. Returns one of the following possible values:

- **True** - The element is contained within the object.
- **False** - The element is not contained within the object.

**JScript Error**

The error **'null' is null or not an object** is raised if elements defined in the script are not present.

### 2.1.2.4 insertAdjacentElement

insertAdjacentElement
The **insertAdjacentElement** method inserts an element at the specified location. Text cannot be inserted while a document is loading; the *onload* event must be completed before attempting to call the **insertAdjacentElement** method.

**Parameters**

**sWhere**

A String that specifies where to insert the HTML element, using one of the following values:

- **beforeBegin** - Inserts **oElement** immediately before the object.
- **afterBegin** - Inserts **oElement** after the start of the object, but before all other content in the object.
- **beforeEnd** - Inserts **oElement** immediately before the end of the object, but after all other content in the object.
- **afterEnd** - Inserts **oElement** immediately after the end of the object.

**oElement**

Object that specifies the element to be inserted adjacent to the object that invoked the **insertAdjacentElement** method.

**Return Value**

Returns an element object.

**No JScript Error**

### 2.1.2.5 mergeAttributes

**mergeAttributes**

The **mergeAttributes** method copies all read/write HTML attributes, events, and styles from one element to another specified element.

**Parameters**

**oSource**

Pointer to an **Object** that specifies the attributes copied to the object that invokes the attributes of the **mergeAttributes** method.

**bPreserve**

Optional. Pointer to a Boolean value that specifies one of the following values:

- **True** - Default. The id and name attributes of the element to which attributes are being merged is preserved.
- **False** - The id and name attributes of the element to which attributes are being merged are not preserved.

**Return Value**

No return value.
2.1.2.6 removeNode

The `removeNode` method removes the specified object from the document hierarchy. The `removeNode` method is accessible at run time. If elements are removed at run time, before the closing tag is parsed, areas of the document might not render.

**Parameters**

`bRemoveChildren`

Optional. A Boolean parameter that specifies one of the following values:

- `false` - Default. The `childNodes` collection of the object is not removed.
- `true` - The `childNodes` collection of the object is removed.

**Return Value**

Returns a reference to the removed object.

2.1.2.7 replaceNode

The `replaceNode` method replaces the object with another element. When a node is replaced, all values that are associated with the replaced object are removed. The `replaceNode` method is accessible at run time. If elements are removed at run time before the closing tag is parsed, areas of the document might not render.

**Parameters**

`oNewNode`

An object that specifies the new element to replace the object.

**Return Value**

Returns a reference to the removed object.

2.1.2.8 swapNode

The `swapNode` method exchanges the location of two objects in the document hierarchy. This method is accessible at run time. If elements are removed at run time, before the closing tag is parsed, areas of the document might not render.

**Parameters**

`oNode`

Element that specifies the existing element.

**Return Values**

Returns a reference to the object that invoked the method.
2.1.3 Collections

The following collections are extensions to the Elements interface as specified in the [DOM Level 2 - Core]:

- all
- children

2.1.3.1 all

all

Returns a reference to the collection of elements contained by the object.

**Note.** The all extension is not supported (does not exist) in IE11 Mode.

The all collection includes one element object for each valid HTML tag. If a valid tag has a matching end tag, both tags are represented by the same element object.

The collection returned by the document's all collection always includes a reference to the html, head, and title objects regardless of whether the tags are present in the document. If the BODY tag is not present, but other HTML tags are, a body object is added to the all collection.

If the document contains invalid or unknown tags, the collection includes one element object for each. Unlike valid end tags, unknown end tags are represented by their own element objects. The order of the element objects is the HTML source order. Although the collection indicates the order of tags, it does not indicate hierarchy.

The name property only applies to some elements such as form elements. If the vIndex is set to a string matching the value of a name property in an element that the name property does not apply, then that element is not added to the collection.

**Syntax**

```
[ collAll = ] object.all
[ oObject = ] object.all(vIndex [, iSubIndex])
```

**Possible Values**

- **collAll** Array of elements contained by the object.
- **oObject** Reference to an individual item in the array of elements contained by the object.
- **vIndex** Required. Integer or string that specifies the element or collection to retrieve. If this parameter is an integer, the method returns the element in the collection at the specified position, where the first element has value 0, the second has 1, and so on. If this parameter is a string and there is more than one element with the name or id property equal to the string, the method returns a collection of matching elements.
- **iSubIndex** Optional. Position of an element to retrieve. This parameter is used when vIndex is a string. The method uses the string to construct a collection of all elements that have a name or id property equal to the string, and then retrieves from this collection the element at the position specified by iSubIndex.

**Members Table**

The following table lists the members exposed by the all object.
Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>Sets or retrieves the number of objects in a collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>Retrieves an object from various collections, including the all collection.</td>
</tr>
<tr>
<td>namedItem</td>
<td>Retrieves an object or a collection from a specified collection.</td>
</tr>
<tr>
<td>tags</td>
<td>Retrieves a collection of objects that have the specified HTML tag name.</td>
</tr>
<tr>
<td>urns</td>
<td>Retrieves a collection of all objects to which a specified behavior is attached.</td>
</tr>
</tbody>
</table>

2.1.3.2 children

Retrieves a collection of DHTML Objects that are direct descendants of the object.

Similar to the objects contained in the all collection, the objects contained in the children collection are undefined if the child elements are overlapping tags.

The children collection can contain HTML elements.

Syntax

```
[ collAll = ] object.children
[ oObject = ] object.children(vIndex [, iSubIndex])
```

Possible Values

collAll Array of elements contained by the object.

oObject Reference to an individual item in the array of elements contained by the object.

vIndex Required. Integer or string that specifies the element or collection to retrieve. If this parameter is an integer, the method returns the element in the collection at the specified position, where the first element has value 0, the second has 1, and so on. If this parameter is a string and there is more than one element with the name or id property equal to the string, the method returns a collection of matching elements.

iSubIndex Optional. Position of an element to retrieve. This parameter is used when vIndex is a string. The method uses the string to construct a collection of all elements that have a name or id property equal to the string, and then retrieves from this collection the element at the position specified by iSubIndex.

Members Table

The following table lists the members exposed by the children object.

Attributes
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>constructor</td>
<td>Returns a reference to the constructor of an object.</td>
</tr>
<tr>
<td>length</td>
<td>Sets or retrieves the number of objects in a collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>Retrieves an object from various collections, including the all collection.</td>
</tr>
<tr>
<td>tags</td>
<td>Retrieves a collection of objects that have the specified HTML tag name.</td>
</tr>
<tr>
<td>urns</td>
<td>Retrieves a collection of all objects to which a specified behavior is attached.</td>
</tr>
</tbody>
</table>

2.2 Extensions to the Comment Interface

The following attributes of the `Comment` interface are prototype extensions to the Comment interface of the Document Object Model (DOM) Level 2 Core Specification [DOM Level 2 - Core].

Attributes

The `Comment` interface is extended by the `text` and `atomic` attributes. For details, see Attributes.

The following IDL Definition documents the `Comment` interface:

```idl
definition Comment : CharacterData {
  // Extension of DOM Level 2:
  attribute text;
  readonly attribute atomic;
};
```

2.2.1 Attributes

The `Comment` interface as specified in the [DOM Level 2 - Core] is extended by the addition of the `text` attribute.

2.2.1.1 text

`text` of type DOMString

The `text` attribute retrieves or sets the text of the object as a string.

2.2.1.2 atomic

`atomic`

The `atomic` attribute is functional but has been deprecated. The use of `atomic` indicates whether `comment` is a standalone tag or is used in a tag pair.

Parameters

None
**Return Values**

Returns one of the following possible values:

1 - if the comment is self-closing `<!-- -->` style.

0 - if the comment uses the deprecated `<comment>` tag.

### 2.3 Extensions to the Document Interface

The following methods are extensions to the `Document` interface of the Document Object Model (DOM) Level 2 Core Specification [DOM Level 2 - Core]:

- `clear`
- `removeNode`
- `replaceNode`
- `swapNode`

The following IDL definition documents the `Document` interface:

```idl
interface Element : Node {
  // Extension of DOM Level 2:
  boolean clear;
  boolean removeNode;
  Node replaceNode (in Node newNode)
    raises(DOMException);
  Node swapNode;
};
```

#### 2.3.1 Methods

The following properties are extensions to the `Document` interface as specified in the [DOM Level 2 - Core]:

- `clear`
- `removeNode`
- `replaceNode`
- `swapNode`

#### 2.3.1.1 clear

**clear**

The `clear` method removes all key/value pairs from the DOM Storage area. Session storage is cleared immediately. Local storage key/value pairs are removed from memory, and disk storage quota is updated.

**Return Value**

None

**JScript Error**

None
2.3.1.2 removeNode

removeNode

The **removeNode** method removes the object from the document hierarchy.

**Parameters**

*bRemoveChildren*

Optional. A Boolean parameter that specifies one of the following values:

- **False** - Default. **childNodes** collection of the object is not removed.
- **True** - **childNodes** collection of the object is removed.

**Return Value**

Returns a reference to the removed object.

2.3.1.3 replaceNode

**replaceNode**

The **replaceNode** method replaces the object with another element. When a node is replaced, all values that are associated with the replaced object are removed. The **replaceNode** method is accessible at run time. If elements are removed at run time before the closing tag is parsed, areas of the document might not render.

**Parameters**

*oNewNode*

An element that specifies the new element to replace the object.

**Return Value**

Returns a reference to the removed object.

2.3.1.4 swapNode

**swapNode**

The **swapNode** method exchanges the location of two objects in the document hierarchy. This method is accessible at run time. If elements are removed at run time, before the closing tag is parsed, areas of the document might not render.

**Parameters**

*oNode*

Element that specifies the existing element.

**Return Values**

Returns a reference to the object that invoked the method.
2.4 Extensions to the Text Interface

The following methods are extensions to the Text interface of the Document Object Model (DOM) Level 2 Core Specification [DOM Level 2 - Core]:

- **removeNode**
- **replaceNode**
- **swapNode**

The following IDL definition documents the Document interface:

```idl
interface Element : Node {
    // Extension of DOM Level 2:
    boolean removeNode;
    Node replaceNode (in Node newNode) raises(DOMException);
    Node swapNode;
}
```

2.4.1 Methods

The following properties are extensions to the Text interface as specified in the [DOM Level 2 - Core]:

- **removeNode**
- **replaceNode**
- **swapNode**

2.4.1.1 removeNode

removeNode

The **removeNode** removes the object from the document hierarchy.

**Parameters**

`bRemoveChildren`

Optional. A Boolean parameter that specifies one of the following values:

- False - Default. **childNodes** collection of the object is not removed.
- True - **childNodes** collection of the object is removed.

**Return Value**

Returns a reference to the removed object.

2.4.1.2 replaceNode

replaceNode

The **replaceNode** method replaces the object with another element.
Parameters

oNewNode

An element that specifies the new element to replace the object.

Return Value

Returns a reference to the removed object.

2.4.1.3 swapNode

swapNode

The swapNode method exchanges the location of two objects in the document hierarchy. This method is accessible at run time. If elements are removed at run time, before the closing tag is parsed, areas of the document might not render.

Parameters

oNode

Element that specifies the existing element.

Return Values

Returns a reference to the object that invoked the method.
3 Security Considerations

There are no additional security considerations.
4 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

- 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

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms "SHOULD" or "SHOULD NOT" implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term "MAY" implies that the product does not follow the prescription.
5 Change Tracking

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

A

Applicability 6
Attributes
  canHaveChildren 8
  canHaveHTML 8
  parentElement 8
  sourceIndex 9
  text 15

C

Change tracking 22
Collections
  all 13
  children 14

G

Glossary 4

I

Implementer - security considerations 20
Informative references 4
Interfaces
  Comment 15
  Document 16
  Element 7
  Text 18
  Introduction 4

M

Methods
  applyElement 9
  atomic 15
  clear 16
  clearAttributes 10
  contains 10
  insertAdjacentElement 10
  mergeAttributes 11
  removeNode (section 2.1.2.6 12, section 2.3.1.2 17, section 2.4.1.1 18)
  replaceNode (section 2.1.2.7 12, section 2.3.1.3 17, section 2.4.1.2 18)
  swapNode (section 2.1.2.8 12, section 2.3.1.4 17, section 2.4.1.3 19)

N

Normative references 4

O

Overview (synopsis) 5

P

Product behavior 21

References 4
  informative 4
  normative 4

Security - implementer considerations 20

Tracking changes 22