@翻译：helurong

@版本：1.0

@开始翻译时间：2016.4.27

**DOM**

**Living Standard — Last Updated 26 April 2016**

Participate:

[GitHub whatwg/dom](https://github.com/whatwg/dom) ([new issue](https://github.com/whatwg/dom/issues/new), [open issues](https://github.com/whatwg/dom/issues), [legacy open bugs](https://www.w3.org/Bugs/Public/buglist.cgi?component=DOM&product=WebAppsWG&resolution=---))

[IRC: #whatwg on Freenode](https://wiki.whatwg.org/wiki/IRC)

Commits:

[GitHub whatwg/dom/commits](https://github.com/whatwg/dom/commits)

[@thedomstandard](https://twitter.com/thedomstandard)

**Abstract摘要**

DOM defines a platform-neutral model for events and node trees.

Dom定义了一个关于事件和节点树的相对独立的平台模型

**Table of Contents 目录**

1. [Goals](https://dom.spec.whatwg.org/#goals)
2. [1 Conformance](https://dom.spec.whatwg.org/#conformance)
   1. [1.1 Dependencies](https://dom.spec.whatwg.org/#dependencies)
   2. [1.2 Extensibility](https://dom.spec.whatwg.org/#extensibility)
3. [2 Terminology](https://dom.spec.whatwg.org/#terminology)
   1. [2.1 Trees](https://dom.spec.whatwg.org/#trees)
   2. [2.2 Strings](https://dom.spec.whatwg.org/#strings)
   3. [2.3 Ordered sets](https://dom.spec.whatwg.org/#ordered%20sets)
   4. [2.4 Selectors](https://dom.spec.whatwg.org/#selectors)
   5. [2.5 Namespaces](https://dom.spec.whatwg.org/#namespaces)
4. [3 Events](https://dom.spec.whatwg.org/#events)
   1. [3.1 Introduction to "DOM Events"](https://dom.spec.whatwg.org/#introduction-to-dom-events)
   2. [3.2 Interface Event](https://dom.spec.whatwg.org/#interface-event)
   3. [3.3 Interface CustomEvent](https://dom.spec.whatwg.org/#interface-customevent)
   4. [3.4 Constructing events](https://dom.spec.whatwg.org/#constructing-events)
   5. [3.5 Defining event interfaces](https://dom.spec.whatwg.org/#defining-event-interfaces)
   6. [3.6 Interface EventTarget](https://dom.spec.whatwg.org/#interface-eventtarget)
   7. [3.7 Observing event listeners](https://dom.spec.whatwg.org/#observing-event-listeners)
   8. [3.8 Dispatching events](https://dom.spec.whatwg.org/#dispatching-events)
   9. [3.9 Firing events](https://dom.spec.whatwg.org/#firing-events)
   10. [3.10 Action versus occurrence](https://dom.spec.whatwg.org/#action-versus-occurance)
5. [4 Nodes](https://dom.spec.whatwg.org/#nodes)
   1. [4.1 Introduction to "The DOM"](https://dom.spec.whatwg.org/#introduction-to-the-dom)
   2. [4.2 Node tree](https://dom.spec.whatwg.org/#node-trees)
      1. [4.2.1 Document tree](https://dom.spec.whatwg.org/#document-trees)
      2. [4.2.2 Shadow tree](https://dom.spec.whatwg.org/#shadow-trees)
         1. [4.2.2.1 Slots](https://dom.spec.whatwg.org/#shadow-tree-slots)
         2. [4.2.2.2 Slotables](https://dom.spec.whatwg.org/#light-tree-slotables)
         3. [4.2.2.3 Finding slots and slotables](https://dom.spec.whatwg.org/#finding-slots-and-slotables)
         4. [4.2.2.4 Assigning slotables and slots](https://dom.spec.whatwg.org/#assigning-slotables-and-slots)
         5. [4.2.2.5 Signaling slot change](https://dom.spec.whatwg.org/#signaling-slot-change)
      3. [4.2.3 Mutation algorithms](https://dom.spec.whatwg.org/#mutation-algorithms)
      4. [4.2.4 Mixin NonElementParentNode](https://dom.spec.whatwg.org/#interface-nonelementparentnode)
      5. [4.2.5 Mixin DocumentOrShadowRoot](https://dom.spec.whatwg.org/#mixin-documentorshadowroot)
      6. [4.2.6 Mixin ParentNode](https://dom.spec.whatwg.org/#interface-parentnode)
      7. [4.2.7 Mixin NonDocumentTypeChildNode](https://dom.spec.whatwg.org/#interface-nondocumenttypechildnode)
      8. [4.2.8 Mixin ChildNode](https://dom.spec.whatwg.org/#interface-childnode)
      9. [4.2.9 Mixin: Slotable](https://dom.spec.whatwg.org/#mixin-slotable)
      10. [4.2.10 Old-style collections: NodeList and HTMLCollection](https://dom.spec.whatwg.org/#old-style-collections)
          1. [4.2.10.1 Interface NodeList](https://dom.spec.whatwg.org/#interface-nodelist)
          2. [4.2.10.2 Interface HTMLCollection](https://dom.spec.whatwg.org/#interface-htmlcollection)
   3. [4.3 Mutation observers](https://dom.spec.whatwg.org/#mutation-observers)
      1. [4.3.1 Interface MutationObserver](https://dom.spec.whatwg.org/#interface-mutationobserver)
      2. [4.3.2 Queuing a mutation record](https://dom.spec.whatwg.org/#queueing-a-mutation-record)
      3. [4.3.3 Interface MutationRecord](https://dom.spec.whatwg.org/#interface-mutationrecord)
      4. [4.3.4 Garbage collection](https://dom.spec.whatwg.org/#garbage-collection)
   4. [4.4 Interface Node](https://dom.spec.whatwg.org/#interface-node)
   5. [4.5 Interface Document](https://dom.spec.whatwg.org/#interface-document)
      1. [4.5.1 Interface DOMImplementation](https://dom.spec.whatwg.org/#interface-domimplementation)
   6. [4.6 Interface DocumentType](https://dom.spec.whatwg.org/#interface-documenttype)
   7. [4.7 Interface DocumentFragment](https://dom.spec.whatwg.org/#interface-documentfragment)
   8. [4.8 Interface ShadowRoot](https://dom.spec.whatwg.org/#interface-shadowroot)
   9. [4.9 Interface Element](https://dom.spec.whatwg.org/#interface-element)
      1. [4.9.1 Interface NamedNodeMap](https://dom.spec.whatwg.org/#interface-namednodemap)
      2. [4.9.2 Interface Attr](https://dom.spec.whatwg.org/#interface-attr)
   10. [4.10 Interface CharacterData](https://dom.spec.whatwg.org/#interface-characterdata)
   11. [4.11 Interface Text](https://dom.spec.whatwg.org/#interface-text)
   12. [4.12 Interface ProcessingInstruction](https://dom.spec.whatwg.org/#interface-processinginstruction)
   13. [4.13 Interface Comment](https://dom.spec.whatwg.org/#interface-comment)
6. [5 Ranges](https://dom.spec.whatwg.org/#ranges)
   1. [5.1 Introduction to "DOM Ranges"](https://dom.spec.whatwg.org/#introduction-to-dom-ranges)
   2. [5.2 Interface Range](https://dom.spec.whatwg.org/#interface-range)
7. [6 Traversal](https://dom.spec.whatwg.org/#traversal)
   1. [6.1 Interface NodeIterator](https://dom.spec.whatwg.org/#interface-nodeiterator)
   2. [6.2 Interface TreeWalker](https://dom.spec.whatwg.org/#interface-treewalker)
   3. [6.3 Interface NodeFilter](https://dom.spec.whatwg.org/#interface-nodefilter)
8. [7 Sets](https://dom.spec.whatwg.org/#sets)
   1. [7.1 Interface DOMTokenList](https://dom.spec.whatwg.org/#interface-domtokenlist)
9. [8 Historical](https://dom.spec.whatwg.org/#historical)
   1. [8.1 DOM Events](https://dom.spec.whatwg.org/#dom-events-changes)
   2. [8.2 DOM Core](https://dom.spec.whatwg.org/#dom-core-changes)
   3. [8.3 DOM Ranges](https://dom.spec.whatwg.org/#dom-ranges-changes)
   4. [8.4 DOM Traversal](https://dom.spec.whatwg.org/#dom-traversal-changes)
10. [Acknowledgments](https://dom.spec.whatwg.org/#acks)
11. [Index](https://dom.spec.whatwg.org/#index)
    1. [Terms defined by this specification](https://dom.spec.whatwg.org/#index-defined-here)
    2. [Terms defined by reference](https://dom.spec.whatwg.org/#index-defined-elsewhere)
12. [References](https://dom.spec.whatwg.org/#references)
    1. [Normative References](https://dom.spec.whatwg.org/#normative)
    2. [Informative References](https://dom.spec.whatwg.org/#informative)
13. [IDL Index](https://dom.spec.whatwg.org/#idl-index)

**Goals目标**

This specification standardizes the DOM. It does so as follows:

本说明书标准化了DOM，如下：

1. By consolidating *DOM Level 3 Core* [[DOM-Level-3-Core]](https://dom.spec.whatwg.org/#biblio-dom-level-3-core), *Element Traversal* [[ELEMENTTRAVERSAL]](https://dom.spec.whatwg.org/#biblio-elementtraversal), *Selectors API Level 2* [[SELECTORS-API2]](https://dom.spec.whatwg.org/#biblio-selectors-api2), the "DOM Event Architecture" and "Basic Event Interfaces" chapters of *DOM Level 3 Events* [[uievents-20031107]](https://dom.spec.whatwg.org/#biblio-uievents-20031107) (specific types of events do not belong in the DOM Standard), and *DOM Level 2 Traversal and Range* [[DOM-Level-2-Traversal-Range]](https://dom.spec.whatwg.org/#biblio-dom-level-2-traversal-range), and:
   * Aligning them with the JavaScript ecosystem where possible.
   * Aligning them with existing implementations.
   * Simplifying them as much as possible.
2. By moving features from the HTML Standard [[HTML]](https://dom.spec.whatwg.org/#biblio-html) that make more sense to be specified as part of the DOM Standard.
3. By defining a replacement for the "Mutation Events" and "Mutation Name Event Types" chapters of *DOM Level 3 Events* [[uievents-20031107]](https://dom.spec.whatwg.org/#biblio-uievents-20031107) as the old model was problematic.

The old model is expected to be removed from implementations in due course.

1. By defining new features that simplify common DOM operations.

**1. Conformance**

All diagrams, examples, and notes in this specification are non-normative, as are all sections explicitly marked non-normative. Everything else in this specification is normative.

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119. For readability, these words do not appear in all uppercase letters in this specification. [[RFC2119]](https://dom.spec.whatwg.org/#biblio-rfc2119)

Requirements phrased in the imperative as part of algorithms (such as "strip any leading space characters" or "return false and terminate these steps") are to be interpreted with the meaning of the keyword ("must", "should", "may", etc.) used in introducing the algorithm.

Conformance requirements phrased as algorithms or specific steps may be implemented in any manner, so long as the end result is equivalent. (In particular, the algorithms defined in this specification are intended to be easy to follow, and not intended to be performant.)

User agents may impose implementation-specific limits on otherwise unconstrained inputs, e.g. to prevent denial of service attacks, to guard against running out of memory, or to work around platform-specific limitations.

When a method or an attribute is said to call another method or attribute, the user agent must invoke its internal API for that attribute or method so that e.g. the author can’t change the behavior by overriding attributes or methods with custom properties or functions in JavaScript.

Unless otherwise stated, string comparisons are done in a [case-sensitive](https://dom.spec.whatwg.org/#case-sensitive) manner.

**1.1. Dependencies**

The IDL fragments in this specification must be interpreted as required for conforming IDL fragments, as described in the Web IDL specification. [[WEBIDL]](https://dom.spec.whatwg.org/#biblio-webidl)

Some of the terms used in this specification are defined in *Encoding*, *Selectors*, *Web IDL*, *XML*, and *Namespaces in XML*. [[ENCODING]](https://dom.spec.whatwg.org/#biblio-encoding) [[SELECTORS4]](https://dom.spec.whatwg.org/#biblio-selectors4) [[WEBIDL]](https://dom.spec.whatwg.org/#biblio-webidl) [[XML]](https://dom.spec.whatwg.org/#biblio-xml) [[XML-NAMES]](https://dom.spec.whatwg.org/#biblio-xml-names)

**1.2. Extensibility**

Vendor-specific proprietary extensions to this specification are strongly discouraged. Authors must not use such extensions, as doing so reduces interoperability and fragments the user base, allowing only users of specific user agents to access the content in question.

When extensions are needed, the DOM Standard can be updated accordingly, or a new standard can be written that hooks into the provided extensibility hooks for *applicable specifications*.

**2. Terminology术语**

The term *context object* means the object on which the algorithm, attribute getter, attribute setter, or method being discussed was called. When the [context object](https://dom.spec.whatwg.org/#context-object) is unambiguous, the term can be omitted.

**2.1. Trees树**

A *tree* is a finite hierarchical tree structure. In *tree order* is preorder, depth-first traversal of a [tree](https://dom.spec.whatwg.org/#concept-tree).

树是一种有限的有层次的树形结构。

An object that *participates* in a [tree](https://dom.spec.whatwg.org/#concept-tree) has a *parent*, which is either another object or null, and an ordered list of zero or more *child* objects. An object *A* whose [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is object *B* is a [child](https://dom.spec.whatwg.org/#concept-tree-child) of *B*.

The *root* of an object is itself, if its [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is null, or else it is the [root](https://dom.spec.whatwg.org/#concept-tree-root) of its [parent](https://dom.spec.whatwg.org/#concept-tree-parent). The [root](https://dom.spec.whatwg.org/#concept-tree-root) of a [tree](https://dom.spec.whatwg.org/#concept-tree) is any object [participating](https://dom.spec.whatwg.org/#concept-tree-participate) in that [tree](https://dom.spec.whatwg.org/#concept-tree) whose [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is null.

An object *A* is called a *descendant* of an object *B*, if either *A* is a [child](https://dom.spec.whatwg.org/#concept-tree-child) of *B* or *A* is a [child](https://dom.spec.whatwg.org/#concept-tree-child) of an object *C* that is a [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) of *B*.

An *inclusive descendant* is an object or one of its [descendants](https://dom.spec.whatwg.org/#concept-tree-descendant).

An object *A* is called an *ancestor* of an object *B* if and only if *B* is a [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) of *A*.

An *inclusive ancestor* is an object or one of its [ancestors](https://dom.spec.whatwg.org/#concept-tree-ancestor).

An object *A* is called a *sibling* of an object *B*, if and only if *B* and *A* share the same non-null [parent](https://dom.spec.whatwg.org/#concept-tree-parent).

An *inclusive sibling* is an object or one of its [siblings](https://dom.spec.whatwg.org/#concept-tree-sibling).

An object *A* is *preceding* an object *B* if *A* and *B* are in the same [tree](https://dom.spec.whatwg.org/#concept-tree) and *A* comes before *B* in [tree order](https://dom.spec.whatwg.org/#concept-tree-order).

An object *A* is *following* an object *B* if *A* and *B* are in the same [tree](https://dom.spec.whatwg.org/#concept-tree) and *A* comes after *B* in [tree order](https://dom.spec.whatwg.org/#concept-tree-order).

The *first child* of an object is its first [child](https://dom.spec.whatwg.org/#concept-tree-child) or null if it has no [children](https://dom.spec.whatwg.org/#concept-tree-child).

The *last child* of an object is its last [child](https://dom.spec.whatwg.org/#concept-tree-child) or null if it has no [children](https://dom.spec.whatwg.org/#concept-tree-child).

The *previous sibling* of an object is its first [preceding](https://dom.spec.whatwg.org/#concept-tree-preceding) [sibling](https://dom.spec.whatwg.org/#concept-tree-sibling) or null if it has no [preceding](https://dom.spec.whatwg.org/#concept-tree-preceding) [sibling](https://dom.spec.whatwg.org/#concept-tree-sibling).

The *next sibling* of an object is its first [following](https://dom.spec.whatwg.org/#concept-tree-following) [sibling](https://dom.spec.whatwg.org/#concept-tree-sibling) or null if it has no [following](https://dom.spec.whatwg.org/#concept-tree-following) [sibling](https://dom.spec.whatwg.org/#concept-tree-sibling).

The *index* of an object is its number of [preceding](https://dom.spec.whatwg.org/#concept-tree-preceding) [siblings](https://dom.spec.whatwg.org/#concept-tree-sibling).

**2.2. Strings**

Comparing two strings in a *case-sensitive* manner means comparing them exactly, code point for code point.

Comparing two strings in a *ASCII case-insensitive* manner means comparing them exactly, code point for code point, except that the characters in the range U+0041 to U+005A (i.e. LATIN CAPITAL LETTER A to LATIN CAPITAL LETTER Z), inclusive, and the corresponding characters in the range U+0061 to U+007A (i.e. LATIN SMALL LETTER A to LATIN SMALL LETTER Z), inclusive, are considered to also match.

*Converting a string to ASCII uppercase* means replacing all characters in the range U+0061 to U+007A, inclusive, with the corresponding characters in the range U+0041 to U+005A, inclusive.

*Converting a string to ASCII lowercase* means replacing all characters in the range U+0041 to U+005A, inclusive, with the corresponding characters in the range U+0061 to U+007A, inclusive.

A string *pattern* is a *prefix match* for a string *s* when *pattern* is not longer than *s* and truncating *s* to *pattern*’s length leaves the two strings as matches of each other.

**2.3. Ordered sets**

The *ordered set parser* takes a string *input* and then runs these steps:

1. Let *position* be a pointer into *input*, initially pointing at the start of the string.
2. Let *tokens* be an ordered set of tokens, initially empty.
3. [Skip ASCII whitespace](https://dom.spec.whatwg.org/#skip-ascii-whitespace).
4. While *position* is not past the end of *input*:
   1. [Collect a code point sequence](https://dom.spec.whatwg.org/#collect-a-code-point-sequence) of code points that are not [ASCII whitespace](https://encoding.spec.whatwg.org/#ascii-whitespace).
   2. If the collected string is not in *tokens*, append the collected string to *tokens*.
   3. [Skip ASCII whitespace](https://dom.spec.whatwg.org/#skip-ascii-whitespace).
5. Return *tokens*.

To *collect a code point sequence* of *code points*, run these steps:

1. Let *input* and *position* be the same variables as those of the same name in the algorithm that invoked these steps.
2. Let *result* be the empty string.
3. While *position* does not point past the end of *input* and the code point at *position* is one of *code points*, append that code point to the end of *result* and advance *position* to the next code point in *input*.
4. Return *result*.

To *skip ASCII whitespace* means to [collect a code point sequence](https://dom.spec.whatwg.org/#collect-a-code-point-sequence) of [ASCII whitespace](https://encoding.spec.whatwg.org/#ascii-whitespace) and discard the return value.

The *ordered set serializer* takes a *set* and returns the concatenation of the strings in *set*, separated from each other by U+0020, if *set* is non-empty, and the empty string otherwise.

**2.4. Selectors**

To *match a relative selectors string* *relativeSelectors* against a *set*, run these steps:

1. Let *s* be the result of [parse a relative selector](https://drafts.csswg.org/selectors-4/#parse-a-relative-selector) from *relativeSelectors* against *set*. [[SELECTORS4]](https://dom.spec.whatwg.org/#biblio-selectors4)
2. If *s* is failure, [throw](https://heycam.github.io/webidl/#dfn-throw) a [SyntaxError](https://heycam.github.io/webidl/#syntaxerror).
3. Return the result of [evaluate a selector](https://drafts.csswg.org/selectors-4/#evaluate-a-selector) *s* using [:scope elements](https://drafts.csswg.org/selectors-4/#scope-element) *set*. [[SELECTORS4]](https://dom.spec.whatwg.org/#biblio-selectors4)

To *scope-match a selectors string* *selectors* against a *node*, run these steps:

1. Let *s* be the result of [parse a selector](https://drafts.csswg.org/selectors-4/#parse-a-selector) *selectors*. [[SELECTORS4]](https://dom.spec.whatwg.org/#biblio-selectors4)
2. If *s* is failure, [throw](https://heycam.github.io/webidl/#dfn-throw) a [SyntaxError](https://heycam.github.io/webidl/#syntaxerror).
3. Return the result of [evaluate a selector](https://drafts.csswg.org/selectors-4/#evaluate-a-selector) *s* against *node*’s [root](https://dom.spec.whatwg.org/#concept-tree-root) using [scoping root](https://drafts.csswg.org/selectors-4/#scoping-root) *node* and scoping method [scope-filtered](https://drafts.csswg.org/selectors-4/#scope-filtered). [[SELECTORS4]](https://dom.spec.whatwg.org/#biblio-selectors4).

Support for namespaces within selectors is not planned and will not be added.

**2.5. Namespaces**

The *HTML namespace* is http://www.w3.org/1999/xhtml.

The *SVG namespace* is http://www.w3.org/2000/svg.

The *XML namespace* is http://www.w3.org/XML/1998/namespace.

The *XMLNS namespace* is http://www.w3.org/2000/xmlns/.

To *validate* a *qualifiedName*, run these steps:

1. If *qualifiedName* does not match the [Name](https://www.w3.org/TR/xml/#NT-Name) production, then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror).
2. If *qualifiedName* does not match the [QName](https://www.w3.org/TR/xml-names/#NT-QName) production, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NamespaceError](https://heycam.github.io/webidl/#namespaceerror).

To *validate and extract* a *namespace* and *qualifiedName*, run these steps:

1. If *namespace* is the empty string, set it to null.
2. [Validate](https://dom.spec.whatwg.org/#validate) *qualifiedName*. Rethrow any exceptions.
3. Let *prefix* be null.
4. Let *localName* be *qualifiedName*.
5. If *qualifiedName* contains a ":" (U+003E), then split the string on it and set *prefix* to the part before and *localName* to the part after.
6. If *prefix* is non-null and *namespace* is null, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NamespaceError](https://heycam.github.io/webidl/#namespaceerror).
7. If *prefix* is "xml" and *namespace* is not the [XML namespace](https://dom.spec.whatwg.org/#xml-namespace), then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NamespaceError](https://heycam.github.io/webidl/#namespaceerror).
8. If either *qualifiedName* or *prefix* is "xmlns" and *namespace* is not the [XMLNS namespace](https://dom.spec.whatwg.org/#xmlns-namespace), then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NamespaceError](https://heycam.github.io/webidl/#namespaceerror).
9. If *namespace* is the [XMLNS namespace](https://dom.spec.whatwg.org/#xmlns-namespace) and neither *qualifiedName* nor *prefix* is "xmlns", then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NamespaceError](https://heycam.github.io/webidl/#namespaceerror).
10. Return *namespace*, *prefix*, and *localName*.

**3. Events**

**3.1. Introduction to "DOM Events"**

Throughout the web platform [events](https://dom.spec.whatwg.org/#concept-event) are [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch) to objects to signal an occurrence, such as network activity or user interaction. These objects implement the [EventTarget](https://dom.spec.whatwg.org/#eventtarget) interface and can therefore add [event listeners](https://dom.spec.whatwg.org/#concept-event-listener) to observe [events](https://dom.spec.whatwg.org/#concept-event) by calling [addEventListener()](https://dom.spec.whatwg.org/#dom-eventtarget-addeventlistener):

obj.addEventListener("load", imgFetched)

function imgFetched(ev) {

// great success

…

}

[Event listeners](https://dom.spec.whatwg.org/#concept-event-listener) can be removed by utilizing the [removeEventListener()](https://dom.spec.whatwg.org/#dom-eventtarget-removeeventlistener) method, passing the same arguments.

[Events](https://dom.spec.whatwg.org/#concept-event) are objects too and implement the [Event](https://dom.spec.whatwg.org/#event) interface (or a derived interface). In the example above *ev* is the [event](https://dom.spec.whatwg.org/#concept-event). It is passed as argument to [event listener](https://dom.spec.whatwg.org/#concept-event-listener)’s **callback** (typically a JavaScript Function as shown above). [Event listeners](https://dom.spec.whatwg.org/#concept-event-listener) key off the [event](https://dom.spec.whatwg.org/#concept-event)’s [type](https://dom.spec.whatwg.org/#dom-event-type) attribute value ("load" in the above example). The [event](https://dom.spec.whatwg.org/#concept-event)’s [target](https://dom.spec.whatwg.org/#dom-event-target) attribute value returns the object to which the [event](https://dom.spec.whatwg.org/#concept-event) was [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch) (*obj* above).

Now while typically [events](https://dom.spec.whatwg.org/#concept-event) are [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch) by the user agent as the result of user interaction or the completion of some task, applications can [dispatch](https://dom.spec.whatwg.org/#concept-event-dispatch) [events](https://dom.spec.whatwg.org/#concept-event) themselves, commonly known as synthetic events:

// add an appropriate event listener

obj.addEventListener("cat", function(e) { process(e.detail) })

// create and dispatch the event

var event = new CustomEvent("cat", {"detail":{"hazcheeseburger":true}})

obj.dispatchEvent(event)

Apart from signaling, [events](https://dom.spec.whatwg.org/#concept-event) are sometimes also used to let an application control what happens next in an operation. For instance as part of form submission an [event](https://dom.spec.whatwg.org/#concept-event) whose [type](https://dom.spec.whatwg.org/#dom-event-type) attribute value is "submit" is [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch). If this [event](https://dom.spec.whatwg.org/#concept-event)’s [preventDefault()](https://dom.spec.whatwg.org/#dom-event-preventdefault) method is invoked, form submission will be terminated. Applications who wish to make use of this functionality through [events](https://dom.spec.whatwg.org/#concept-event) [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch) by the application (synthetic events) can make use of the return value of the [dispatchEvent()](https://dom.spec.whatwg.org/#dom-eventtarget-dispatchevent) method:

if(obj.dispatchEvent(event)) {

// event was not canceled, time for some magic

…

}

When an [event](https://dom.spec.whatwg.org/#concept-event) is [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch) to an object that [participates](https://dom.spec.whatwg.org/#concept-tree-participate) in a [tree](https://dom.spec.whatwg.org/#concept-tree) (e.g. an [element](https://dom.spec.whatwg.org/#concept-element)), it can reach [event listeners](https://dom.spec.whatwg.org/#concept-event-listener) on that object’s [ancestors](https://dom.spec.whatwg.org/#concept-tree-ancestor) too. First all object’s [ancestor](https://dom.spec.whatwg.org/#concept-tree-ancestor) [event listeners](https://dom.spec.whatwg.org/#concept-event-listener) whose **capture** variable is set to true are invoked, in [tree order](https://dom.spec.whatwg.org/#concept-tree-order). Second, object’s own [event listeners](https://dom.spec.whatwg.org/#concept-event-listener) are invoked. And finally, and only if [event](https://dom.spec.whatwg.org/#concept-event)’s [bubbles](https://dom.spec.whatwg.org/#dom-event-bubbles) attribute value is true, object’s [ancestor](https://dom.spec.whatwg.org/#concept-tree-ancestor) [event listeners](https://dom.spec.whatwg.org/#concept-event-listener) are invoked again, but now in reverse [tree order](https://dom.spec.whatwg.org/#concept-tree-order).

Lets look at an example of how [events](https://dom.spec.whatwg.org/#concept-event) work in a [tree](https://dom.spec.whatwg.org/#concept-tree):

<!doctype html>

<html>

<head>

<title>Boring example</title>

</head>

<body>

<p>Hello <span id=x>world</span>!</p>

<script>

function test(e) {

debug(e.target, e.currentTarget, e.eventPhase)

}

document.addEventListener("hey", test, {capture: true})

document.body.addEventListener("hey", test)

var ev = new Event("hey", {bubbles:true})

document.getElementById("x").dispatchEvent(ev)

</script>

</body>

</html>

The debug function will be invoked twice. Each time the [event](https://dom.spec.whatwg.org/#concept-event)’s [target](https://dom.spec.whatwg.org/#dom-event-target) attribute value will be the span [element](https://dom.spec.whatwg.org/#concept-element). The first time [currentTarget](https://dom.spec.whatwg.org/#dom-event-currenttarget) attribute’s value will be the [document](https://dom.spec.whatwg.org/#concept-document), the second time the body [element](https://dom.spec.whatwg.org/#concept-element). [eventPhase](https://dom.spec.whatwg.org/#dom-event-eventphase) attribute’s value switches from [CAPTURING\_PHASE](https://dom.spec.whatwg.org/#dom-event-capturing_phase) to [BUBBLING\_PHASE](https://dom.spec.whatwg.org/#dom-event-bubbling_phase). If an [event listener](https://dom.spec.whatwg.org/#concept-event-listener) was registered for the span [element](https://dom.spec.whatwg.org/#concept-element), [eventPhase](https://dom.spec.whatwg.org/#dom-event-eventphase) attribute’s value would have been [AT\_TARGET](https://dom.spec.whatwg.org/#dom-event-at_target).

**3.2. Interface** [**Event**](https://dom.spec.whatwg.org/#event)

[*Constructor*(DOMString *type*, optional [EventInit](https://dom.spec.whatwg.org/#dictdef-eventinit) *eventInitDict*),

Exposed=(Window,Worker)]

interface *Event* {

readonly attribute DOMString [type](https://dom.spec.whatwg.org/#dom-event-type);

readonly attribute [EventTarget](https://dom.spec.whatwg.org/#eventtarget)? [target](https://dom.spec.whatwg.org/#dom-event-target);

readonly attribute [EventTarget](https://dom.spec.whatwg.org/#eventtarget)? [currentTarget](https://dom.spec.whatwg.org/#dom-event-currenttarget);

const unsigned short [NONE](https://dom.spec.whatwg.org/#dom-event-none) = 0;

const unsigned short [CAPTURING\_PHASE](https://dom.spec.whatwg.org/#dom-event-capturing_phase) = 1;

const unsigned short [AT\_TARGET](https://dom.spec.whatwg.org/#dom-event-at_target) = 2;

const unsigned short [BUBBLING\_PHASE](https://dom.spec.whatwg.org/#dom-event-bubbling_phase) = 3;

readonly attribute unsigned short [eventPhase](https://dom.spec.whatwg.org/#dom-event-eventphase);

void [stopPropagation](https://dom.spec.whatwg.org/#dom-event-stoppropagation)();

void [stopImmediatePropagation](https://dom.spec.whatwg.org/#dom-event-stopimmediatepropagation)();

readonly attribute boolean [bubbles](https://dom.spec.whatwg.org/#dom-event-bubbles);

readonly attribute boolean [cancelable](https://dom.spec.whatwg.org/#dom-event-cancelable);

void [preventDefault](https://dom.spec.whatwg.org/#dom-event-preventdefault)();

readonly attribute boolean [defaultPrevented](https://dom.spec.whatwg.org/#dom-event-defaultprevented);

[Unforgeable] readonly attribute boolean [isTrusted](https://dom.spec.whatwg.org/#dom-event-istrusted);

readonly attribute [DOMTimeStamp](https://heycam.github.io/webidl/#common-domtimestamp) [timeStamp](https://dom.spec.whatwg.org/#dom-event-timestamp);

void [initEvent](https://dom.spec.whatwg.org/#dom-event-initevent)(DOMString *type*, boolean *bubbles*, boolean *cancelable*);

};

dictionary *EventInit* {

boolean *bubbles* = false;

boolean *cancelable* = false;

};

An *event* allows for signaling that something has occurred, e.g. that an image has completed downloading. It is represented by the [Event](https://dom.spec.whatwg.org/#event) interface or an interface that inherits from the [Event](https://dom.spec.whatwg.org/#event) interface.

*event* = new [Event](https://dom.spec.whatwg.org/#dom-event-event)(*type* [, *eventInitDict*])

Returns a new *event* whose [type](https://dom.spec.whatwg.org/#dom-event-type) attribute value is set to *type*. The optional *eventInitDict* argument allows for setting the [bubbles](https://dom.spec.whatwg.org/#dom-event-bubbles) and [cancelable](https://dom.spec.whatwg.org/#dom-event-cancelable) attributes via object members of the same name.

*event* . [type](https://dom.spec.whatwg.org/#dom-event-type)

Returns the type of *event*, e.g. "click", "hashchange", or "submit".

*event* . [target](https://dom.spec.whatwg.org/#dom-event-target)

Returns the object to which *event* is [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch).

*event* . [currentTarget](https://dom.spec.whatwg.org/#dom-event-currenttarget)

Returns the object whose [event listener](https://dom.spec.whatwg.org/#concept-event-listener)’s **callback** is currently being invoked.

*event* . [eventPhase](https://dom.spec.whatwg.org/#dom-event-eventphase)

Returns the [event](https://dom.spec.whatwg.org/#concept-event)’s phase, which is one of [NONE](https://dom.spec.whatwg.org/#dom-event-none), [CAPTURING\_PHASE](https://dom.spec.whatwg.org/#dom-event-capturing_phase), [AT\_TARGET](https://dom.spec.whatwg.org/#dom-event-at_target), and [BUBBLING\_PHASE](https://dom.spec.whatwg.org/#dom-event-bubbling_phase).

*event* . [stopPropagation](https://dom.spec.whatwg.org/#dom-event-stoppropagation)()

When [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch) in a [tree](https://dom.spec.whatwg.org/#concept-tree), invoking this method prevents *event* from reaching any objects other than the current object.

*event* . [stopImmediatePropagation](https://dom.spec.whatwg.org/#dom-event-stopimmediatepropagation)()

Invoking this method prevents *event* from reaching any registered [event listeners](https://dom.spec.whatwg.org/#concept-event-listener) after the current one finishes running and, when [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch) in a [tree](https://dom.spec.whatwg.org/#concept-tree), also prevents *event* from reaching any other objects.

*event* . [bubbles](https://dom.spec.whatwg.org/#dom-event-bubbles)

Returns true or false depending on how *event* was initialized. True if *event* goes through its [target](https://dom.spec.whatwg.org/#dom-event-target) attribute value’s [ancestors](https://dom.spec.whatwg.org/#concept-tree-ancestor) in reverse [tree order](https://dom.spec.whatwg.org/#concept-tree-order), and false otherwise.

*event* . [cancelable](https://dom.spec.whatwg.org/#dom-event-cancelable)

Returns true or false depending on how *event* was initialized. Its return value does not always carry meaning, but true can indicate that part of the operation during which *event* was [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch), can be canceled by invoking the [preventDefault()](https://dom.spec.whatwg.org/#dom-event-preventdefault) method.

*event* . [preventDefault](https://dom.spec.whatwg.org/#dom-event-preventdefault)()

If invoked when the [cancelable](https://dom.spec.whatwg.org/#dom-event-cancelable) attribute value is true, and while executing a listener for the *event* with [passive](https://dom.spec.whatwg.org/#dom-addeventlisteneroptions-passive) set to false, signals to the operation that caused *event* to be [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch) that it needs to be canceled.

*event* . [defaultPrevented](https://dom.spec.whatwg.org/#dom-event-defaultprevented)

Returns true if [preventDefault()](https://dom.spec.whatwg.org/#dom-event-preventdefault) was invoked successfully to indicate cancellation, and false otherwise.

*event* . [isTrusted](https://dom.spec.whatwg.org/#dom-event-istrusted)

Returns true if *event* was [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch) by the user agent, and false otherwise.

*event* . [timeStamp](https://dom.spec.whatwg.org/#dom-event-timestamp)

Returns the creation time of *event* as the number of milliseconds that passed since 00:00:00 UTC on 1 January 1970.

The *type* attribute must return the value it was initialized to. When an [event](https://dom.spec.whatwg.org/#concept-event) is created the attribute must be initialized to the empty string.

The *target* and *currentTarget* attributes must return the values they were initialized to. When an [event](https://dom.spec.whatwg.org/#concept-event) is created the attributes must be initialized to null.

The *eventPhase* attribute must return the value it was initialized to, which must be one of the following:

*NONE* (numeric value 0)

[Events](https://dom.spec.whatwg.org/#concept-event) not currently [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch) are in this phase.

*CAPTURING\_PHASE* (numeric value 1)

When an [event](https://dom.spec.whatwg.org/#concept-event) is [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch) to an object that [participates](https://dom.spec.whatwg.org/#concept-tree-participate) in a [tree](https://dom.spec.whatwg.org/#concept-tree) it will be in this phase before it reaches its [target](https://dom.spec.whatwg.org/#dom-event-target) attribute value.

*AT\_TARGET* (numeric value 2)

When an [event](https://dom.spec.whatwg.org/#concept-event) is [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch) it will be in this phase on its [target](https://dom.spec.whatwg.org/#dom-event-target) attribute value.

*BUBBLING\_PHASE* (numeric value 3)

When an [event](https://dom.spec.whatwg.org/#concept-event) is [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch) to an object that [participates](https://dom.spec.whatwg.org/#concept-tree-participate) in a [tree](https://dom.spec.whatwg.org/#concept-tree) it will be in this phase after it reaches its [target](https://dom.spec.whatwg.org/#dom-event-target) attribute value.

Initially the attribute must be initialized to [NONE](https://dom.spec.whatwg.org/#dom-event-none).

Each [event](https://dom.spec.whatwg.org/#concept-event) has the following associated flags that are all initially unset:

* *stop propagation flag*
* *stop immediate propagation flag*
* *canceled flag*
* *initialized flag*
* *dispatch flag*
* *in passive listener flag*

The *stopPropagation()* method must set the [stop propagation flag](https://dom.spec.whatwg.org/#stop-propagation-flag).

The *stopImmediatePropagation()* method must set both the [stop propagation flag](https://dom.spec.whatwg.org/#stop-propagation-flag) and [stop immediate propagation flag](https://dom.spec.whatwg.org/#stop-immediate-propagation-flag).

The *bubbles* and *cancelable* attributes must return the values they were initialized to.

The *preventDefault()* method, when invoked, must set the [canceled flag](https://dom.spec.whatwg.org/#canceled-flag) if the [cancelable](https://dom.spec.whatwg.org/#dom-event-cancelable) attribute value is true and the [in passive listener flag](https://dom.spec.whatwg.org/#in-passive-listener-flag) is unset.

This means there are scenarios where invoking [preventDefault()](https://dom.spec.whatwg.org/#dom-event-preventdefault) has no effect. User agents are encouraged to log the precise cause in a developer console, to aid debugging.

The *defaultPrevented* attribute must return true if the [canceled flag](https://dom.spec.whatwg.org/#canceled-flag) is set and false otherwise.

The *isTrusted* attribute must return the value it was initialized to. When an [event](https://dom.spec.whatwg.org/#concept-event) is created the attribute must be initialized to false.

The *timeStamp* attribute must return the value it was initialized to. When an [event](https://dom.spec.whatwg.org/#concept-event) is created the attribute must be initialized to the number of milliseconds that have passed since 00:00:00 UTC on 1 January 1970, ignoring leap seconds.

This is highly likely to change and already does not reflect implementations well. Please see [dom #23](https://github.com/whatwg/dom/issues/23) for more details.

To *initialize* an *event*, with *type*, *bubbles*, and *cancelable*, run these steps:

1. Set the [initialized flag](https://dom.spec.whatwg.org/#initialized-flag).
2. Unset the [stop propagation flag](https://dom.spec.whatwg.org/#stop-propagation-flag), [stop immediate propagation flag](https://dom.spec.whatwg.org/#stop-immediate-propagation-flag), and [canceled flag](https://dom.spec.whatwg.org/#canceled-flag).
3. Set the [isTrusted](https://dom.spec.whatwg.org/#dom-event-istrusted) attribute to false.
4. Set the [target](https://dom.spec.whatwg.org/#dom-event-target) attribute to null.
5. Set the [type](https://dom.spec.whatwg.org/#dom-event-type) attribute to *type*.
6. Set the [bubbles](https://dom.spec.whatwg.org/#dom-event-bubbles) attribute to *bubbles*.
7. Set the [cancelable](https://dom.spec.whatwg.org/#dom-event-cancelable) attribute to *cancelable*.

The *initEvent(type, bubbles, cancelable)* method, when invoked, must run these steps:

1. If [context object](https://dom.spec.whatwg.org/#context-object)’s [dispatch flag](https://dom.spec.whatwg.org/#dispatch-flag) is set, terminate these steps.
2. [Initialize](https://dom.spec.whatwg.org/#concept-event-initialize) the [context object](https://dom.spec.whatwg.org/#context-object) with *type*, *bubbles*, and *cancelable*.

As [events](https://dom.spec.whatwg.org/#concept-event) have constructors [initEvent()](https://dom.spec.whatwg.org/#dom-event-initevent) is superfluous. However, it has to be supported for legacy content.

**3.3. Interface** [**CustomEvent**](https://dom.spec.whatwg.org/#customevent)

[*Constructor*(DOMString *type*, optional [CustomEventInit](https://dom.spec.whatwg.org/#dictdef-customeventinit) *eventInitDict*),

Exposed=(Window,Worker)]

interface *CustomEvent* : [Event](https://dom.spec.whatwg.org/#event) {

readonly attribute any [detail](https://dom.spec.whatwg.org/#dom-customevent-detail);

void [initCustomEvent](https://dom.spec.whatwg.org/#dom-customevent-initcustomevent)(DOMString *type*, boolean *bubbles*, boolean *cancelable*, any *detail*);

};

dictionary *CustomEventInit* : [EventInit](https://dom.spec.whatwg.org/#dictdef-eventinit) {

any *detail* = null;

};

[Events](https://dom.spec.whatwg.org/#concept-event) using the [CustomEvent](https://dom.spec.whatwg.org/#customevent) interface can be used to carry custom data.

*event* = new [CustomEvent](https://dom.spec.whatwg.org/#dom-customevent-customevent)(*type* [, *eventInitDict*])

Works analogously to the constructor for [Event](https://dom.spec.whatwg.org/#event) except that the optional *eventInitDict* argument now allows for setting the [detail](https://dom.spec.whatwg.org/#dom-customevent-detail) attribute too.

*event* . [detail](https://dom.spec.whatwg.org/#dom-customevent-detail)

Returns any custom data *event* was created with. Typically used for synthetic events.

The *detail* attribute must return the value it was initialized to.

The *initCustomEvent(type, bubbles, cancelable, detail)* method must, when invoked, run these steps:

1. If [context object](https://dom.spec.whatwg.org/#context-object)’s [dispatch flag](https://dom.spec.whatwg.org/#dispatch-flag) is set, terminate these steps.
2. [Initialize](https://dom.spec.whatwg.org/#concept-event-initialize) the [context object](https://dom.spec.whatwg.org/#context-object) with *type*, *bubbles*, and *cancelable*.
3. Set [context object](https://dom.spec.whatwg.org/#context-object)’s [detail](https://dom.spec.whatwg.org/#dom-customevent-detail) attribute to *detail*.

**3.4. Constructing events**

When a *constructor* of the [Event](https://dom.spec.whatwg.org/#event) interface, or of an interface that inherits from the [Event](https://dom.spec.whatwg.org/#event) interface, is invoked, these steps must be run:

1. Create an [event](https://dom.spec.whatwg.org/#concept-event) that uses the interface the constructor was invoked upon.
2. Set its [initialized flag](https://dom.spec.whatwg.org/#initialized-flag).
3. Initialize the [type](https://dom.spec.whatwg.org/#dom-event-type) attribute to the *type* argument.
4. If there is an *eventInitDict* argument, then for each [dictionary member](https://heycam.github.io/webidl/#dfn-dictionary-member) present, find the attribute on [event](https://dom.spec.whatwg.org/#concept-event) whose [identifier](https://heycam.github.io/webidl/#dfn-identifier) matches the key of the [dictionary member](https://heycam.github.io/webidl/#dfn-dictionary-member) and then set the attribute to the value of that [dictionary member](https://heycam.github.io/webidl/#dfn-dictionary-member).
5. Return the [event](https://dom.spec.whatwg.org/#concept-event).

**3.5. Defining event interfaces**

In general, when defining a new interface that inherits from [Event](https://dom.spec.whatwg.org/#event) please always ask feedback from the [WHATWG](https://whatwg.org/) or the [W3C WebApps WG](https://www.w3.org/2008/webapps/) community.

The [CustomEvent](https://dom.spec.whatwg.org/#customevent) interface can be used as starting point. However, do not introduce any init*\**Event() methods as they are redundant with constructors. Interfaces that inherit from the [Event](https://dom.spec.whatwg.org/#event) interface that have such a method only have it for historical reasons.

**3.6. Interface** [**EventTarget**](https://dom.spec.whatwg.org/#eventtarget)

[Exposed=(Window,Worker)]

interface *EventTarget* {

void [addEventListener](https://dom.spec.whatwg.org/#dom-eventtarget-addeventlistener)(DOMString *type*, [EventListener](https://dom.spec.whatwg.org/#callbackdef-eventlistener)? callback, optional ([AddEventListenerOptions](https://dom.spec.whatwg.org/#dictdef-addeventlisteneroptions) or boolean) *options*);

void [removeEventListener](https://dom.spec.whatwg.org/#dom-eventtarget-removeeventlistener)(DOMString *type*, [EventListener](https://dom.spec.whatwg.org/#callbackdef-eventlistener)? callback, optional ([EventListenerOptions](https://dom.spec.whatwg.org/#dictdef-eventlisteneroptions) or boolean) *options*);

boolean [dispatchEvent](https://dom.spec.whatwg.org/#dom-eventtarget-dispatchevent)([Event](https://dom.spec.whatwg.org/#event) *event*);

};

callback interface *EventListener* {

void *handleEvent*([Event](https://dom.spec.whatwg.org/#event) *event*);

};

dictionary *EventListenerOptions* {

boolean *capture* = false;

};

dictionary *AddEventListenerOptions* : [EventListenerOptions](https://dom.spec.whatwg.org/#dictdef-eventlisteneroptions) {

boolean *passive* = false;

boolean *once* = false;

};

The [EventTarget](https://dom.spec.whatwg.org/#eventtarget) object represents the target to which an [event](https://dom.spec.whatwg.org/#concept-event) is [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch) when something has occurred.

Each [EventTarget](https://dom.spec.whatwg.org/#eventtarget) object has an associated list of [event listeners](https://dom.spec.whatwg.org/#concept-event-listener).

An *event listener* can be used to observe a specific [event](https://dom.spec.whatwg.org/#concept-event).

An [event listener](https://dom.spec.whatwg.org/#concept-event-listener) consists of these fields:

* **type** (a string)
* **callback** (an [EventListener](https://dom.spec.whatwg.org/#callbackdef-eventlistener))
* **capture** (a boolean, initially false)
* **passive** (a boolean, initially false)
* **once** (a boolean, initially false)
* **removed** (a boolean for bookkeeping purposes, initially false)

Although **callback** is an [EventListener](https://dom.spec.whatwg.org/#callbackdef-eventlistener), as can be seen from the fields above, an [event listener](https://dom.spec.whatwg.org/#concept-event-listener) is a broader concept.

Each [EventTarget](https://dom.spec.whatwg.org/#eventtarget) object also has an associated *get the parent* algorithm, which takes an [event](https://dom.spec.whatwg.org/#concept-event) *event*, and returns an [EventTarget](https://dom.spec.whatwg.org/#eventtarget) object. Unless specified otherwise it returns null.

[Nodes](https://dom.spec.whatwg.org/#concept-node) and [documents](https://dom.spec.whatwg.org/#concept-document) override the [get the parent](https://dom.spec.whatwg.org/#get-the-parent) algorithm.

*target* . [addEventListener](https://dom.spec.whatwg.org/#dom-eventtarget-addeventlistener)(*type*, *callback* [, *options*])

Appends an [event listener](https://dom.spec.whatwg.org/#concept-event-listener) for [events](https://dom.spec.whatwg.org/#concept-event) whose [type](https://dom.spec.whatwg.org/#dom-event-type) attribute value is *type*. The *callback* argument sets the **callback** that will be invoked when the [event](https://dom.spec.whatwg.org/#concept-event) is [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch).

The *options* argument sets listener-specific options. For compatibility this can be just a boolean, in which case the method behaves exactly as if the value was specified as *options*’ capture member.

When set to true, *options*’ capture member prevents **callback** from being invoked when the [event](https://dom.spec.whatwg.org/#concept-event)’s [eventPhase](https://dom.spec.whatwg.org/#dom-event-eventphase) attribute value is [BUBBLING\_PHASE](https://dom.spec.whatwg.org/#dom-event-bubbling_phase). When false (or not present), **callback** will not be invoked when [event](https://dom.spec.whatwg.org/#concept-event)’s [eventPhase](https://dom.spec.whatwg.org/#dom-event-eventphase) attribute value is [CAPTURING\_PHASE](https://dom.spec.whatwg.org/#dom-event-capturing_phase). Either way, **callback** will be invoked if [event](https://dom.spec.whatwg.org/#concept-event)’s [eventPhase](https://dom.spec.whatwg.org/#dom-event-eventphase) attribute value is [AT\_TARGET](https://dom.spec.whatwg.org/#dom-event-at_target).

When set to true, *options*’ passive member indicates that the **callback** will not cancel the event by invoking [preventDefault()](https://dom.spec.whatwg.org/#dom-event-preventdefault). This is used to enable performance optimizations described in [§3.7 Observing event listeners](https://dom.spec.whatwg.org/#observing-event-listeners).

When set to true, *options*’s once member indicates that the **callback** will only be invoked once after which the event listener will be removed.

The [event listener](https://dom.spec.whatwg.org/#concept-event-listener) is appended to *target*’s list of [event listeners](https://dom.spec.whatwg.org/#concept-event-listener) and is not appended if it is a duplicate, i.e., having the same **type**, **callback**, and **capture** values.

*target* . [removeEventListener](https://dom.spec.whatwg.org/#dom-eventtarget-removeeventlistener)(*type*, *callback* [, *options*])

Remove the [event listener](https://dom.spec.whatwg.org/#concept-event-listener) in *target*’s list of [event listeners](https://dom.spec.whatwg.org/#concept-event-listener) with the same *type*, *callback*, and *options*.

*target* . [dispatchEvent](https://dom.spec.whatwg.org/#dom-eventtarget-dispatchevent)(*event*)

[Dispatches](https://dom.spec.whatwg.org/#concept-event-dispatch) a synthetic event *event* to *target* and returns true if either *event*’s [cancelable](https://dom.spec.whatwg.org/#dom-event-cancelable) attribute value is false or its [preventDefault()](https://dom.spec.whatwg.org/#dom-event-preventdefault) method was not invoked, and false otherwise.

To *flatten* *options*, run these steps:

1. Let *capture* be false.
2. If *options* is a boolean, set *capture* to *options*.
3. If *options* is a dictionary, then set *capture* to *options*’s [capture](https://dom.spec.whatwg.org/#dom-eventlisteneroptions-capture).
4. Return *capture*.

To *flatten more* *options*, run these steps:

1. Let *capture* be the result of [flattening](https://dom.spec.whatwg.org/#concept-flatten-options) *options*.
2. Let *once* and *passive* be false.
3. If *options* is a dictionary, then set *passive* to *options*’s [passive](https://dom.spec.whatwg.org/#dom-addeventlisteneroptions-passive) and *once* to *options*’s [once](https://dom.spec.whatwg.org/#dom-addeventlisteneroptions-once).
4. Return *capture*, *passive*, and *once*.

The *addEventListener(type, callback, options)* method, when invoked, must run these steps:

1. If the [context object](https://dom.spec.whatwg.org/#context-object)’s global object is a [ServiceWorkerGlobalScope](https://slightlyoff.github.io/ServiceWorker/spec/service_worker/#service-worker-global-scope-interface) object and its associated [service worker](https://slightlyoff.github.io/ServiceWorker/spec/service_worker/#dfn-service-worker)’s [script resource](https://slightlyoff.github.io/ServiceWorker/spec/service_worker/#dfn-script-resource)’s [has ever been evaluated flag](https://slightlyoff.github.io/ServiceWorker/spec/service_worker/#dfn-has-ever-been-evaluated-flag) is set, [throw](https://heycam.github.io/webidl/#dfn-throw) a TypeError. [[SERVICE-WORKERS]](https://dom.spec.whatwg.org/#biblio-service-workers)

To optimize storing the event types allowed for the service worker and to avoid non-deterministic changes to the event listeners, invocation of the method is allowed only during the very first evaluation of the service worker script.

1. If *callback* is null, terminate these steps.
2. Let *capture*, *passive*, and *once* be the result of [flattening more](https://dom.spec.whatwg.org/#event-flatten-more) *options*.
3. If [context object](https://dom.spec.whatwg.org/#context-object)’s associated list of [event listener](https://dom.spec.whatwg.org/#concept-event-listener) does not contain an [event listener](https://dom.spec.whatwg.org/#concept-event-listener) whose **type** is *type*, **callback** is *callback*, and **capture** is *capture*, then append a new [event listener](https://dom.spec.whatwg.org/#concept-event-listener) to it, whose **type** is *type*, **callback** is *callback*, **capture** is *capture*, **passive** is *passive*, and **once** is *once*.

The *removeEventListener(type, callback, options)* method, when invoked, must, run these steps

1. If the [context object](https://dom.spec.whatwg.org/#context-object)’s global object is a [ServiceWorkerGlobalScope](https://slightlyoff.github.io/ServiceWorker/spec/service_worker/#service-worker-global-scope-interface) object and its associated [service worker](https://slightlyoff.github.io/ServiceWorker/spec/service_worker/#dfn-service-worker)’s [script resource](https://slightlyoff.github.io/ServiceWorker/spec/service_worker/#dfn-script-resource)’s [has ever been evaluated flag](https://slightlyoff.github.io/ServiceWorker/spec/service_worker/#dfn-has-ever-been-evaluated-flag) is set, [throw](https://heycam.github.io/webidl/#dfn-throw) a TypeError. [[SERVICE-WORKERS]](https://dom.spec.whatwg.org/#biblio-service-workers)
2. Let *capture* be the result of [flattening](https://dom.spec.whatwg.org/#concept-flatten-options) *options*.
3. If there is an [event listener](https://dom.spec.whatwg.org/#concept-event-listener) in the associated list of [event listeners](https://dom.spec.whatwg.org/#concept-event-listener) whose **type** is *type*, **callback** is *callback*, and **capture** is *capture*, then set that [event listener](https://dom.spec.whatwg.org/#concept-event-listener)’s **removed** to true and remove it from the associated list of [event listeners](https://dom.spec.whatwg.org/#concept-event-listener).

The *dispatchEvent(event)* method, when invoked, must run these steps:

1. If *event*’s [dispatch flag](https://dom.spec.whatwg.org/#dispatch-flag) is set, or if its [initialized flag](https://dom.spec.whatwg.org/#initialized-flag) is not set, then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidStateError](https://heycam.github.io/webidl/#invalidstateerror).
2. Initialize *event*’s [isTrusted](https://dom.spec.whatwg.org/#dom-event-istrusted) attribute to false.
3. Return the result of [dispatching](https://dom.spec.whatwg.org/#concept-event-dispatch) *event* to the [context object](https://dom.spec.whatwg.org/#context-object).

**3.7. Observing event listeners**

In general, developers do not expect the presence of an [event listener](https://dom.spec.whatwg.org/#concept-event-listener) to be observable. The impact of an [event listener](https://dom.spec.whatwg.org/#concept-event-listener) is determined by its **callback**. That is, a developer adding a no-op [event listener](https://dom.spec.whatwg.org/#concept-event-listener) would not expect it to have any side effects.

Unfortunately, some event APIs have been designed such that implementing them efficiently requires observing [event listeners](https://dom.spec.whatwg.org/#concept-event-listener). This can make the presence of listeners observable in that even empty listeners can have a dramatic performance impact on the behavior of the application. For example, touch and wheel events which can be used to block asynchronous scrolling. In some cases this problem can be mitigated by specifying the event to be [cancelable](https://dom.spec.whatwg.org/#dom-event-cancelable) only when there is at least one non-[passive](https://dom.spec.whatwg.org/#dom-addeventlisteneroptions-passive) listener. For example, non-[passive](https://dom.spec.whatwg.org/#dom-addeventlisteneroptions-passive) [TouchEvent](https://www.w3.org/TR/touch-events/#touchevent-interface) listeners must block scrolling, but if all listeners are [passive](https://dom.spec.whatwg.org/#dom-addeventlisteneroptions-passive) then scrolling can be allowed to start [in parallel](https://html.spec.whatwg.org/multipage/infrastructure.html#in-parallel) by making the [TouchEvent](https://www.w3.org/TR/touch-events/#touchevent-interface) uncancelable (so that calls to [preventDefault()](https://dom.spec.whatwg.org/#dom-event-preventdefault) are ignored). So code dispatching an event is able to observe the absence of non-[passive](https://dom.spec.whatwg.org/#dom-addeventlisteneroptions-passive) listeners, and use that to clear the [cancelable](https://dom.spec.whatwg.org/#dom-event-cancelable) property of the event being dispatched.

Ideally, any new event APIs are defined such that they do not need this property (use [public-scrip-coord@w3.org](https://lists.w3.org/Archives/Public/public-script-coord/) for discussion).

**3.8. Dispatching events**

To *dispatch* an *event* to a *target*, with an optional *target override*, run these steps:

1. Set *event*’s [dispatch flag](https://dom.spec.whatwg.org/#dispatch-flag).
2. Initialize *event*’s [target](https://dom.spec.whatwg.org/#dom-event-target) attribute to *target override*, if it is given, and *target* otherwise.
3. Let *eventPath* be a list containing *target*.
4. While invoking [get the parent](https://dom.spec.whatwg.org/#get-the-parent), given *event*, on *eventPath*’s last item, does not return null, append the return value to *eventPath*.
5. Initialize *event*’s [eventPhase](https://dom.spec.whatwg.org/#dom-event-eventphase) attribute to [CAPTURING\_PHASE](https://dom.spec.whatwg.org/#dom-event-capturing_phase).
6. For each *object* in *eventPath*, in reverse order, if *object* is not *target*, [invoke](https://dom.spec.whatwg.org/#concept-event-listener-invoke) *object* with *event*.
7. Initialize *event*’s [eventPhase](https://dom.spec.whatwg.org/#dom-event-eventphase) attribute to [AT\_TARGET](https://dom.spec.whatwg.org/#dom-event-at_target).
8. [Invoke](https://dom.spec.whatwg.org/#concept-event-listener-invoke) *target* with *event*.
9. If *event*’s [bubbles](https://dom.spec.whatwg.org/#dom-event-bubbles) attribute value is true, run these substeps:
   1. Initialize *event*’s [eventPhase](https://dom.spec.whatwg.org/#dom-event-eventphase) attribute to [BUBBLING\_PHASE](https://dom.spec.whatwg.org/#dom-event-bubbling_phase).
   2. For each *object* in *eventPath*, if *object* is not *target*, [invoke](https://dom.spec.whatwg.org/#concept-event-listener-invoke) *object* with *event*.
10. Unset *event*’s [dispatch flag](https://dom.spec.whatwg.org/#dispatch-flag).
11. Initialize *event*’s [eventPhase](https://dom.spec.whatwg.org/#dom-event-eventphase) attribute to [NONE](https://dom.spec.whatwg.org/#dom-event-none).
12. Initialize *event*’s [currentTarget](https://dom.spec.whatwg.org/#dom-event-currenttarget) attribute to null.
13. Return false if *event*’s [canceled flag](https://dom.spec.whatwg.org/#canceled-flag) is set, and true otherwise.

To *invoke* an *object* with *event*, run these steps:

1. If *event*’s [stop propagation flag](https://dom.spec.whatwg.org/#stop-propagation-flag) is set, then terminate these steps.
2. Let *listeners* be the empty list.
3. For each [event listener](https://dom.spec.whatwg.org/#concept-event-listener) associated with *object*, append a pointer to the [event listener](https://dom.spec.whatwg.org/#concept-event-listener) to *listeners*.

This avoids [event listeners](https://dom.spec.whatwg.org/#concept-event-listener) added after this point from being run. Note that removal still has an effect due to the **removed** field.

1. Initialize *event*’s [currentTarget](https://dom.spec.whatwg.org/#dom-event-currenttarget) attribute to *object*.
2. Let *found* be the result of running [inner invoke](https://dom.spec.whatwg.org/#concept-event-listener-inner-invoke) *object* with *event*.
3. If *found* is false, run these substeps:
   1. Let *originalEventType* be *event*’s [type](https://dom.spec.whatwg.org/#dom-event-type) attribute value.
   2. If *event*’s [type](https://dom.spec.whatwg.org/#dom-event-type) attribute value is a match for any of the strings in the first column in the following table, set *event*’s [type](https://dom.spec.whatwg.org/#dom-event-type) attribute value to the string in the second column on the same row as the matching string, and terminate these substeps otherwise.

| **Event type** | **Legacy event type** |
| --- | --- |
| "animationend" | "webkitAnimationEnd" |
| "animationiteration" | "webkitAnimationIteration" |
| "animationstart" | "webkitAnimationStart" |
| "transitionend" | "webkitTransitionEnd" |

* 1. [Inner invoke](https://dom.spec.whatwg.org/#concept-event-listener-inner-invoke) *object* with *event*.
  2. Set *event*’s [type](https://dom.spec.whatwg.org/#dom-event-type) attribute value to *originalEventType*.

To *inner invoke* an *object* with *event*, run these steps:

1. Let *found* be false.
2. For each *listener* in *listeners*, whose **removed** is false, run these substeps:
   1. If *event*’s [type](https://dom.spec.whatwg.org/#dom-event-type) attribute value is not *listener*’s **type**, terminate these substeps (and run them for the next [event listener](https://dom.spec.whatwg.org/#concept-event-listener)).
   2. Set *found* to true.
   3. If *event*’s [eventPhase](https://dom.spec.whatwg.org/#dom-event-eventphase) attribute value is [CAPTURING\_PHASE](https://dom.spec.whatwg.org/#dom-event-capturing_phase) and *listener*’s **capture** is false, terminate these substeps (and run them for the next [event listener](https://dom.spec.whatwg.org/#concept-event-listener)).
   4. If *event*’s [eventPhase](https://dom.spec.whatwg.org/#dom-event-eventphase) attribute value is [BUBBLING\_PHASE](https://dom.spec.whatwg.org/#dom-event-bubbling_phase) and *listener*’s **capture** is true, terminate these substeps (and run them for the next [event listener](https://dom.spec.whatwg.org/#concept-event-listener)).
   5. If *listener*’s **passive** is true, set *event*’s [in passive listener flag](https://dom.spec.whatwg.org/#in-passive-listener-flag).
   6. Call *listener*’s **callback**’s [handleEvent()](https://dom.spec.whatwg.org/#dom-eventlistener-handleevent), with *event* as argument and *event*’s [currentTarget](https://dom.spec.whatwg.org/#dom-event-currenttarget) attribute value as [callback this value](https://heycam.github.io/webidl/#dfn-callback-this-value). If this throws an exception, [report the exception](https://html.spec.whatwg.org/multipage/webappapis.html#report-the-exception).
   7. Unset *event*’s [in passive listener flag](https://dom.spec.whatwg.org/#in-passive-listener-flag).
   8. If *listener*’s **once** is true, then set *listener*’s **removed** to true and remove it from *object*’s associated list of [event listeners](https://dom.spec.whatwg.org/#concept-event-listener).
   9. If *event*’s [stop immediate propagation flag](https://dom.spec.whatwg.org/#stop-immediate-propagation-flag) is set, return *found*.
3. Return *found*.

**3.9. Firing events**

To *fire an event named e* means that a new [event](https://dom.spec.whatwg.org/#concept-event) using the [Event](https://dom.spec.whatwg.org/#event) interface, with its [type](https://dom.spec.whatwg.org/#dom-event-type) attribute initialized to *e*, and its [isTrusted](https://dom.spec.whatwg.org/#dom-event-istrusted) attribute initialized to true, is to be [dispatched](https://dom.spec.whatwg.org/#concept-event-dispatch) to the given object.

Fire in the context of DOM is short for creating, initializing, and [dispatching](https://dom.spec.whatwg.org/#concept-event-dispatch) an [event](https://dom.spec.whatwg.org/#concept-event). [Fire an event](https://dom.spec.whatwg.org/#concept-event-fire) makes that process easier to write down. If the [event](https://dom.spec.whatwg.org/#concept-event) needs its [bubbles](https://dom.spec.whatwg.org/#dom-event-bubbles) or [cancelable](https://dom.spec.whatwg.org/#dom-event-cancelable) attribute initialized, one could write "[fire an event](https://dom.spec.whatwg.org/#concept-event-fire) named submit with its [cancelable](https://dom.spec.whatwg.org/#dom-event-cancelable) attribute initialized to true".

**3.10. Action versus occurrence**

An [event](https://dom.spec.whatwg.org/#concept-event) signifies an occurrence, not an action. Phrased differently, it represents a notification from an algorithm and can be used to influence the future course of that algorithm (e.g., through invoking [preventDefault()](https://dom.spec.whatwg.org/#dom-event-preventdefault)). [Events](https://dom.spec.whatwg.org/#concept-event) must not be used as actions or initiators that cause some algorithm to start running. That is not what they are for.

This is called out here specifically because previous iterations of the DOM had a concept of "default actions" associated with [events](https://dom.spec.whatwg.org/#concept-event) that gave folks all the wrong ideas. [Events](https://dom.spec.whatwg.org/#concept-event) do not represent or cause actions, they can only be used to influence an ongoing one.

**4. Nodes**

**4.1. Introduction to "The DOM"**

In its original sense, "The DOM" is an API for accessing and manipulating documents (in particular, HTML and XML documents). In this specification, the term "document" is used for any markup-based resource, ranging from short static documents to long essays or reports with rich multimedia, as well as to fully-fledged interactive applications.

Each such document is represented as a [node tree](https://dom.spec.whatwg.org/#concept-node-tree). Some of the [nodes](https://dom.spec.whatwg.org/#concept-node) in a [tree](https://dom.spec.whatwg.org/#concept-tree) can have [children](https://dom.spec.whatwg.org/#concept-tree-child), while others are always leaves.

To illustrate, consider this HTML document:

<!DOCTYPE html>

<html class=e>

<head><title>Aliens?</title></head>

<body>Why yes.</body>

</html>

It is represented as follows:

* [Document](https://dom.spec.whatwg.org/#concept-document)
  + [Doctype](https://dom.spec.whatwg.org/#concept-doctype): html
  + [Element](https://dom.spec.whatwg.org/#element): html class="e"
    - [Element](https://dom.spec.whatwg.org/#element): head
      * [Element](https://dom.spec.whatwg.org/#element): title
        + [Text](https://dom.spec.whatwg.org/#text): Aliens?
    - [Text](https://dom.spec.whatwg.org/#text): ⏎␣
    - [Element](https://dom.spec.whatwg.org/#element): body
      * [Text](https://dom.spec.whatwg.org/#text): Why yes.⏎

Note that, due to the magic that is [HTML parsing](https://html.spec.whatwg.org/multipage/syntax.html#html-parser), not all [ASCII whitespace](https://encoding.spec.whatwg.org/#ascii-whitespace) were turned into [Text](https://dom.spec.whatwg.org/#text) [nodes](https://dom.spec.whatwg.org/#concept-node), but the general concept is clear. Markup goes in, a [tree](https://dom.spec.whatwg.org/#concept-tree) of [nodes](https://dom.spec.whatwg.org/#concept-node) comes out.

The most excellent [Live DOM Viewer](http://software.hixie.ch/utilities/js/live-dom-viewer/) can be used to explore this matter in more detail.

**4.2. Node tree**

[Document](https://dom.spec.whatwg.org/#document), [DocumentType](https://dom.spec.whatwg.org/#documenttype), [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment), [ShadowRoot](https://dom.spec.whatwg.org/#shadowroot), [Element](https://dom.spec.whatwg.org/#element), [Text](https://dom.spec.whatwg.org/#text), [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction), and [Comment](https://dom.spec.whatwg.org/#comment) objects (simply called *nodes*) [participate](https://dom.spec.whatwg.org/#concept-tree-participate) in a [tree](https://dom.spec.whatwg.org/#concept-tree), simply named the *node tree*.

A [node tree](https://dom.spec.whatwg.org/#concept-node-tree) is constrained as follows, expressed as a relationship between the type of [node](https://dom.spec.whatwg.org/#concept-node) and its allowed [children](https://dom.spec.whatwg.org/#concept-tree-child):

[Document](https://dom.spec.whatwg.org/#document)

In [tree order](https://dom.spec.whatwg.org/#concept-tree-order):

1. Zero or more nodes each of which is [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction) or [Comment](https://dom.spec.whatwg.org/#comment).
2. Optionally one [DocumentType](https://dom.spec.whatwg.org/#documenttype) node.
3. Zero or more nodes each of which is [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction) or [Comment](https://dom.spec.whatwg.org/#comment).
4. Optionally one [Element](https://dom.spec.whatwg.org/#element) node.
5. Zero or more nodes each of which is [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction) or [Comment](https://dom.spec.whatwg.org/#comment).

[DocumentFragment](https://dom.spec.whatwg.org/#documentfragment)

[ShadowRoot](https://dom.spec.whatwg.org/#shadowroot)

[Element](https://dom.spec.whatwg.org/#element)

Zero or more nodes each of which is [Element](https://dom.spec.whatwg.org/#element), [Text](https://dom.spec.whatwg.org/#text), [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction), or [Comment](https://dom.spec.whatwg.org/#comment).

[DocumentType](https://dom.spec.whatwg.org/#documenttype)

[Text](https://dom.spec.whatwg.org/#text)

[ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction)

[Comment](https://dom.spec.whatwg.org/#comment)

None.

The *length* of a [node](https://dom.spec.whatwg.org/#concept-node) *node* depends on *node*:

[DocumentType](https://dom.spec.whatwg.org/#documenttype)

Zero.

[Text](https://dom.spec.whatwg.org/#text)

[ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction)

[Comment](https://dom.spec.whatwg.org/#comment)

Its [length](https://dom.spec.whatwg.org/#dom-characterdata-length) attribute value.

Any other node

Its number of [children](https://dom.spec.whatwg.org/#concept-tree-child).

A [node](https://dom.spec.whatwg.org/#concept-node) is considered *empty* if its [length](https://dom.spec.whatwg.org/#concept-node-length) is zero.

**4.2.1. Document tree**

A *document tree* is a [node tree](https://dom.spec.whatwg.org/#concept-node-tree) whose [root](https://dom.spec.whatwg.org/#concept-tree-root) is a [document](https://dom.spec.whatwg.org/#concept-document).

The *document element* of a [document](https://dom.spec.whatwg.org/#concept-document) is the [element](https://dom.spec.whatwg.org/#concept-element) whose [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is that [document](https://dom.spec.whatwg.org/#concept-document), if it exists, and null otherwise.

Per the [node tree](https://dom.spec.whatwg.org/#concept-node-tree) constraints, there can be only one such [element](https://dom.spec.whatwg.org/#concept-element).

An [element](https://dom.spec.whatwg.org/#concept-element) is *in a document* if its [root](https://dom.spec.whatwg.org/#concept-tree-root) is a [document](https://dom.spec.whatwg.org/#concept-document).

**4.2.2. Shadow tree**

A *shadow tree* is a [node tree](https://dom.spec.whatwg.org/#concept-node-tree) whose [root](https://dom.spec.whatwg.org/#concept-tree-root) is a [shadow root](https://dom.spec.whatwg.org/#concept-shadow-root).

A [shadow root](https://dom.spec.whatwg.org/#concept-shadow-root) is always connected to another [node tree](https://dom.spec.whatwg.org/#concept-node-tree) through its [host](https://dom.spec.whatwg.org/#concept-documentfragment-host). A [shadow tree](https://dom.spec.whatwg.org/#concept-shadow-tree) is therefore never alone. The [node tree](https://dom.spec.whatwg.org/#concept-node-tree) of a [shadow root](https://dom.spec.whatwg.org/#concept-shadow-root)’s [host](https://dom.spec.whatwg.org/#concept-documentfragment-host) is sometimes referred to as the *light tree*.

A [shadow tree](https://dom.spec.whatwg.org/#concept-shadow-tree)’s corresponding [light tree](https://dom.spec.whatwg.org/#concept-light-tree) can be a [shadow tree](https://dom.spec.whatwg.org/#concept-shadow-tree) itself.

An [element](https://dom.spec.whatwg.org/#concept-element) is *in a shadow-including document* if its [shadow-including root](https://dom.spec.whatwg.org/#concept-shadow-including-root) is a [document](https://dom.spec.whatwg.org/#concept-document).

**4.2.2.1. Slots**

A [shadow tree](https://dom.spec.whatwg.org/#concept-shadow-tree) contains zero or more [elements](https://dom.spec.whatwg.org/#concept-element) that are *slots*.

A [slot](https://dom.spec.whatwg.org/#concept-slot) can only be created through HTML’s [slot](https://html.spec.whatwg.org/multipage/scripting.html#the-slot-element) element.

A [slot](https://dom.spec.whatwg.org/#concept-slot) has an associated *name* (a string). Unless stated otherwise it is the empty string.

Use these [attribute change steps](https://dom.spec.whatwg.org/#concept-element-attributes-change-ext) to update a [slot](https://dom.spec.whatwg.org/#concept-slot)’s [name](https://dom.spec.whatwg.org/#slot-name):

1. If *element* is a [slot](https://dom.spec.whatwg.org/#concept-slot), *localName* is name, and *namespace* is null, then:
   1. If *value* is *oldValue*, then return.
   2. If *value* is null and *oldValue* is the empty string, then return.
   3. If *value* is the empty string and *oldValue* is null, then return.
   4. If *value* is null or the empty string, then set *element*’s [name](https://dom.spec.whatwg.org/#slot-name) to the empty string.
   5. Otherwise, set *element*’s [name](https://dom.spec.whatwg.org/#slot-name) to *value*.
   6. Run [assign slotables for a tree](https://dom.spec.whatwg.org/#assign-slotables-for-a-tree) with *element*’s [tree](https://dom.spec.whatwg.org/#concept-tree).

The first [slot](https://dom.spec.whatwg.org/#concept-slot) in a [shadow tree](https://dom.spec.whatwg.org/#concept-shadow-tree), in [tree order](https://dom.spec.whatwg.org/#concept-tree-order), whose [name](https://dom.spec.whatwg.org/#slot-name) is the empty string, is sometimes known as the "default slot".

A [slot](https://dom.spec.whatwg.org/#concept-slot) has an associated *assigned nodes* (a list of [slotables](https://dom.spec.whatwg.org/#concept-slotable)). Unless stated otherwise it is empty.

**4.2.2.2. Slotables**

[Element](https://dom.spec.whatwg.org/#element) and [Text](https://dom.spec.whatwg.org/#text) [nodes](https://dom.spec.whatwg.org/#concept-node) are *slotables*.

A [slot](https://dom.spec.whatwg.org/#concept-slot) can be a [slotable](https://dom.spec.whatwg.org/#concept-slotable).

A [slotable](https://dom.spec.whatwg.org/#concept-slotable) has an associated *name* (a string). Unless stated otherwise it is the empty string.

Use these [attribute change steps](https://dom.spec.whatwg.org/#concept-element-attributes-change-ext) to update a [slotable](https://dom.spec.whatwg.org/#concept-slotable)’s [name](https://dom.spec.whatwg.org/#slotable-name):

1. If *localName* is slot and *namespace* is null, then:
   1. If *value* is *oldValue*, then return.
   2. If *value* is null and *oldValue* is the empty string, then return.
   3. If *value* is the empty string and *oldValue* is null, then return.
   4. If *value* is null or the empty string, then set *element*’s [name](https://dom.spec.whatwg.org/#slotable-name) to the empty string.
   5. Otherwise, set *element*’s [name](https://dom.spec.whatwg.org/#slotable-name) to *value*.
   6. If *element* is [assigned](https://dom.spec.whatwg.org/#slotable-assigned), then run [assign slotables](https://dom.spec.whatwg.org/#assign-slotables) for *element*’s [assigned slot](https://dom.spec.whatwg.org/#slotable-assigned-slot).
   7. Run [assign a slot](https://dom.spec.whatwg.org/#assign-a-slot) for *element*.

A [slotable](https://dom.spec.whatwg.org/#concept-slotable) has an associated *assigned slot* (null or a [slot](https://dom.spec.whatwg.org/#concept-slot)). Unless stated otherwise it is null. A [slotable](https://dom.spec.whatwg.org/#concept-slotable) is *assigned* if its [assigned slot](https://dom.spec.whatwg.org/#slotable-assigned-slot) is non-null.

**4.2.2.3. Finding slots and slotables**

To *find a slot* for a given [slotable](https://dom.spec.whatwg.org/#concept-slotable) *slotable* and an optional *open flag* (unset unless stated otherwise), run these steps:

1. If *slotable*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is null, then return null.
2. Let *shadow* be *slotable*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent)’s [shadow root](https://dom.spec.whatwg.org/#concept-element-shadow-root).
3. If *shadow* is null, then return null.
4. If the *open flag* is set and *shadow*’s [mode](https://dom.spec.whatwg.org/#shadowroot-mode) is *not* "open", then return null.
5. Return the first [slot](https://dom.spec.whatwg.org/#concept-slot) in *shadow*’s [tree](https://dom.spec.whatwg.org/#concept-tree) whose [name](https://dom.spec.whatwg.org/#slot-name) is *slotable*’s [name](https://dom.spec.whatwg.org/#slotable-name), if any, and null otherwise.

To *find slotables* for a given [slot](https://dom.spec.whatwg.org/#concept-slot) *slot*, run these steps:

1. Let *result* be an empty list.
2. If *slot*’s [root](https://dom.spec.whatwg.org/#concept-tree-root) is not a [shadow root](https://dom.spec.whatwg.org/#concept-shadow-root), then return *result*.
3. Let *host* be *slot*’s [root](https://dom.spec.whatwg.org/#concept-tree-root)’s [host](https://dom.spec.whatwg.org/#concept-documentfragment-host).
4. For each [slotable](https://dom.spec.whatwg.org/#concept-slotable) [child](https://dom.spec.whatwg.org/#concept-tree-child) of *host*, *slotable*, in [tree order](https://dom.spec.whatwg.org/#concept-tree-order), run these substeps:
   1. Let *foundSlot* be the result of [finding a slot](https://dom.spec.whatwg.org/#find-a-slot) given *slotable*.
   2. If *foundSlot* is *slot*, then append *slotable* to *result*.
5. Return *result*.

To *find flattened slotables* for a given [slot](https://dom.spec.whatwg.org/#concept-slot) *slot*, run these steps:

1. Let *result* be an empty list.
2. Let *slotables* be the result of [finding slotables](https://dom.spec.whatwg.org/#find-slotables) given *slot*.
3. If *slotables* is the empty list, then append each [slotable](https://dom.spec.whatwg.org/#concept-slotable) [child](https://dom.spec.whatwg.org/#concept-tree-child) of *slot*, in [tree order](https://dom.spec.whatwg.org/#concept-tree-order), to *slotables*.
4. For each *node* in *slotables*, run these substeps:
   1. If *node* is a [slot](https://dom.spec.whatwg.org/#concept-slot), run these subsubsteps:
      1. Let *temporaryResult* be the result of [finding flattened slotables](https://dom.spec.whatwg.org/#find-flattened-slotables) given *node*.
      2. Append each [slotable](https://dom.spec.whatwg.org/#concept-slotable) in *temporaryResult*, in order, to *result*.
   2. Otherwise, append *node* to *result*.
5. Return *result*.

**4.2.2.4. Assigning slotables and slots**

To *assign slotables*, for a [slot](https://dom.spec.whatwg.org/#concept-slot) *slot* with an optional *suppress signaling flag* (unset unless stated otherwise), run these steps:

1. Let *slotables* be the result of [finding slotables](https://dom.spec.whatwg.org/#find-slotables) for *slot*.
2. If *suppress signaling flag* is unset, and *slotables* and *slot*’s [assigned nodes](https://dom.spec.whatwg.org/#slot-assigned-nodes) are not identical, then run [signal a slot change](https://dom.spec.whatwg.org/#signal-a-slot-change) for *slot*.
3. Set *slot*’s [assigned nodes](https://dom.spec.whatwg.org/#slot-assigned-nodes) to *slotables*.
4. For each *slotable* in *slotables*, set *slotable*’s [assigned slot](https://dom.spec.whatwg.org/#slotable-assigned-slot) to *slot*.

To *assign slotables for a tree*, given a [tree](https://dom.spec.whatwg.org/#concept-tree) *tree* and an optional set of [slots](https://dom.spec.whatwg.org/#concept-slot) *noSignalSlots* (empty unless stated otherwise), run these steps for each [slot](https://dom.spec.whatwg.org/#concept-slot) *slot* in *tree*, in [tree order](https://dom.spec.whatwg.org/#concept-tree-order):

1. Let *suppress signaling flag* be set, if *slot* is in *noSignalSlots*, and unset otherwise.
2. Run [assign slotables](https://dom.spec.whatwg.org/#assign-slotables) for *slot* with *suppress signaling flag*.

To *assign a slot*, given a [slotable](https://dom.spec.whatwg.org/#concept-slotable) *slotable*, run these steps:

1. Let *slot* be the result of [finding a slot](https://dom.spec.whatwg.org/#find-a-slot) with *slotable*.
2. If *slot* is non-null, then run [assign slotables](https://dom.spec.whatwg.org/#assign-slotables) for *slot*.

**4.2.2.5. Signaling slot change**

Each [unit of related similar-origin browsing contexts](https://html.spec.whatwg.org/multipage/browsers.html#unit-of-related-similar-origin-browsing-contexts) has a *signal slot list* (a list of [slots](https://dom.spec.whatwg.org/#concept-slot)). Unless stated otherwise it is empty. [[HTML]](https://dom.spec.whatwg.org/#biblio-html)

To *signal a slot change*, for a [slot](https://dom.spec.whatwg.org/#concept-slot) *slot*, run these steps:

1. If *slot* is not in [unit of related similar-origin browsing contexts](https://html.spec.whatwg.org/multipage/browsers.html#unit-of-related-similar-origin-browsing-contexts)' [signal slot list](https://dom.spec.whatwg.org/#signal-slot-list), append *slot* to [unit of related similar-origin browsing contexts](https://html.spec.whatwg.org/multipage/browsers.html#unit-of-related-similar-origin-browsing-contexts)' [signal slot list](https://dom.spec.whatwg.org/#signal-slot-list).
2. If *slot* is [assigned](https://dom.spec.whatwg.org/#slotable-assigned), then run [signal a slot change](https://dom.spec.whatwg.org/#signal-a-slot-change) for *slot*’s [assigned slot](https://dom.spec.whatwg.org/#slotable-assigned-slot).
3. Otherwise, if *slot*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is a [slot](https://dom.spec.whatwg.org/#concept-slot) and *slot*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent)’s [assigned nodes](https://dom.spec.whatwg.org/#slot-assigned-nodes) is the empty list, then run [signal a slot change](https://dom.spec.whatwg.org/#signal-a-slot-change) for *slot*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
4. [Queue a mutation observer compound microtask](https://dom.spec.whatwg.org/#queue-a-mutation-observer-compound-microtask).

**4.2.3. Mutation algorithms**

To *ensure pre-insertion validity* of a *node* into a *parent* before a *child*, run these steps:

1. If *parent* is not a [Document](https://dom.spec.whatwg.org/#document), [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment), or [Element](https://dom.spec.whatwg.org/#element) [node](https://dom.spec.whatwg.org/#concept-node), [throw](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror).
2. If *node* is a [host-including inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-host-including-inclusive-ancestor) of *parent*, [throw](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror).
3. If *child* is not null and its [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is not *parent*, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotFoundError](https://heycam.github.io/webidl/#notfounderror).
4. If *node* is not a [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment), [DocumentType](https://dom.spec.whatwg.org/#documenttype), [Element](https://dom.spec.whatwg.org/#element), [Text](https://dom.spec.whatwg.org/#text), [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction), or [Comment](https://dom.spec.whatwg.org/#comment) [node](https://dom.spec.whatwg.org/#concept-node), [throw](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror).
5. If either *node* is a [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node) and *parent* is a [document](https://dom.spec.whatwg.org/#concept-document), or *node* is a [doctype](https://dom.spec.whatwg.org/#concept-doctype) and *parent* is not a [document](https://dom.spec.whatwg.org/#concept-document), [throw](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror).
6. If *parent* is a [document](https://dom.spec.whatwg.org/#concept-document), and any of the statements below, switched on *node*, are true, [throw](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror).

[DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [node](https://dom.spec.whatwg.org/#concept-node)

If *node* has more than one [element](https://dom.spec.whatwg.org/#concept-element) [child](https://dom.spec.whatwg.org/#concept-tree-child) or has a [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node) [child](https://dom.spec.whatwg.org/#concept-tree-child).

Otherwise, if *node* has one [element](https://dom.spec.whatwg.org/#concept-element) [child](https://dom.spec.whatwg.org/#concept-tree-child) and either *parent* has an [element](https://dom.spec.whatwg.org/#concept-element) [child](https://dom.spec.whatwg.org/#concept-tree-child), *child* is a [doctype](https://dom.spec.whatwg.org/#concept-doctype), or *child* is not null and a [doctype](https://dom.spec.whatwg.org/#concept-doctype) is [following](https://dom.spec.whatwg.org/#concept-tree-following) *child*.

[element](https://dom.spec.whatwg.org/#concept-element)

*parent* has an [element](https://dom.spec.whatwg.org/#concept-element) [child](https://dom.spec.whatwg.org/#concept-tree-child), *child* is a [doctype](https://dom.spec.whatwg.org/#concept-doctype), or *child* is not null and a [doctype](https://dom.spec.whatwg.org/#concept-doctype) is [following](https://dom.spec.whatwg.org/#concept-tree-following) *child*.

[doctype](https://dom.spec.whatwg.org/#concept-doctype)

*parent* has a [doctype](https://dom.spec.whatwg.org/#concept-doctype) [child](https://dom.spec.whatwg.org/#concept-tree-child), *child* is non-null and an [element](https://dom.spec.whatwg.org/#concept-element) is [preceding](https://dom.spec.whatwg.org/#concept-tree-preceding) *child*, or *child* is null and *parent* has an [element](https://dom.spec.whatwg.org/#concept-element) [child](https://dom.spec.whatwg.org/#concept-tree-child).

To *pre-insert* a *node* into a *parent* before a *child*, run these steps:

1. [Ensure pre-insertion validity](https://dom.spec.whatwg.org/#concept-node-ensure-pre-insertion-validity) of *node* into *parent* before *child*.
2. Let *reference child* be *child*.
3. If *reference child* is *node*, set it to *node*’s [next sibling](https://dom.spec.whatwg.org/#concept-tree-next-sibling).
4. [Adopt](https://dom.spec.whatwg.org/#concept-node-adopt) *node* into *parent*’s [node document](https://dom.spec.whatwg.org/#concept-node-document).
5. [Insert](https://dom.spec.whatwg.org/#concept-node-insert) *node* into *parent* before *reference child*.
6. Return *node*.

[Specifications](https://dom.spec.whatwg.org/#other-applicable-specifications) may define *insertion steps* for all or some [nodes](https://dom.spec.whatwg.org/#concept-node). The algorithm is passed *insertedNode*, as indicated in the [insert](https://dom.spec.whatwg.org/#concept-node-insert) algorithm below.

To *insert* a *node* into a *parent* before a *child*, with an optional *suppress observers flag*, run these steps:

1. Let *count* be the number of [children](https://dom.spec.whatwg.org/#concept-tree-child) of *node* if it is a [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [node](https://dom.spec.whatwg.org/#concept-node), and one otherwise.
2. If *child* is non-null, run these substeps:
   1. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [start node](https://dom.spec.whatwg.org/#concept-range-start-node) is *parent* and [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) is greater than *child*’s [index](https://dom.spec.whatwg.org/#concept-tree-index), increase its [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) by *count*.
   2. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [end node](https://dom.spec.whatwg.org/#concept-range-end-node) is *parent* and [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset) is greater than *child*’s [index](https://dom.spec.whatwg.org/#concept-tree-index), increase its [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset) by *count*.
3. Let *nodes* be *node*’s [children](https://dom.spec.whatwg.org/#concept-tree-child) if *node* is a [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [node](https://dom.spec.whatwg.org/#concept-node), and a list containing solely *node* otherwise.
4. If *node* is a [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [node](https://dom.spec.whatwg.org/#concept-node), [remove](https://dom.spec.whatwg.org/#concept-node-remove) its [children](https://dom.spec.whatwg.org/#concept-tree-child) with the *suppress observers flag* set.
5. If *node* is a [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [node](https://dom.spec.whatwg.org/#concept-node), [queue a mutation record](https://dom.spec.whatwg.org/#queue-a-mutation-record) of "childList" for *node* with removedNodes *nodes*.

This step intentionally does not pay attention to the *suppress observers flag*.

1. For each *node* in *nodes*, in [tree order](https://dom.spec.whatwg.org/#concept-tree-order), run these substeps:
   1. Insert *node* into *parent* before *child* or at the end of *parent* if *child* is null.
   2. If *parent* is a [shadow host](https://dom.spec.whatwg.org/#element-shadow-host) and *node* is a [slotable](https://dom.spec.whatwg.org/#concept-slotable), then [assign a slot](https://dom.spec.whatwg.org/#assign-a-slot) for *node*.
   3. If *parent* is a [slot](https://dom.spec.whatwg.org/#concept-slot) whose [assigned nodes](https://dom.spec.whatwg.org/#slot-assigned-nodes) is the empty list, then run [signal a slot change](https://dom.spec.whatwg.org/#signal-a-slot-change) for *parent*.
   4. Run [assign slotables for a tree](https://dom.spec.whatwg.org/#assign-slotables-for-a-tree) with *node*’s [tree](https://dom.spec.whatwg.org/#concept-tree) and a set containing each [inclusive descendant](https://dom.spec.whatwg.org/#concept-tree-inclusive-descendant) of *node* that is a [slot](https://dom.spec.whatwg.org/#concept-slot).
   5. For each [shadow-including inclusive descendant](https://dom.spec.whatwg.org/#concept-shadow-including-inclusive-descendant) *inclusiveDescendant* of *node*, in [shadow-including tree order](https://dom.spec.whatwg.org/#concept-shadow-including-tree-order), run these subsubsteps:
      1. Run the [insertion steps](https://dom.spec.whatwg.org/#concept-node-insert-ext) with *inclusiveDescendant*.
      2. If *inclusiveDescendant* is [in a shadow-including document](https://dom.spec.whatwg.org/#in-a-shadow-including-document), then:
         1. If *inclusiveDescendant* is [custom](https://dom.spec.whatwg.org/#concept-element-custom), then [enqueue a custom element callback reaction](https://html.spec.whatwg.org/multipage/scripting.html#enqueue-a-custom-element-callback-reaction) with *inclusiveDescendant*, callback name "connectedCallback", and an empty argument list.
         2. Otherwise, [try to upgrade](https://html.spec.whatwg.org/multipage/scripting.html#concept-try-upgrade) *inclusiveDescendant*.

If this successfully upgrades *inclusiveDescendant*, its connectedCallback will be enqueued automatically during the [upgrade an element](https://html.spec.whatwg.org/multipage/scripting.html#concept-upgrade-an-element) algorithm.

1. If *suppress observers flag* is unset, [queue a mutation record](https://dom.spec.whatwg.org/#queue-a-mutation-record) of "childList" for *parent* with addedNodes *nodes*, nextSibling *child*, and previousSibling *child*’s [previous sibling](https://dom.spec.whatwg.org/#concept-tree-previous-sibling) or *parent*’s [last child](https://dom.spec.whatwg.org/#concept-tree-last-child) if *child* is null.

To *append* a *node* to a *parent*, [pre-insert](https://dom.spec.whatwg.org/#concept-node-pre-insert) *node* into *parent* before null.

To *replace* a *child* with *node* within a *parent*, run these steps:

1. If *parent* is not a [Document](https://dom.spec.whatwg.org/#document), [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment), or [Element](https://dom.spec.whatwg.org/#element) [node](https://dom.spec.whatwg.org/#concept-node), [throw](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror).
2. If *node* is a [host-including inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-host-including-inclusive-ancestor) of *parent*, [throw](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror).
3. If *child*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is not *parent*, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotFoundError](https://heycam.github.io/webidl/#notfounderror).
4. If *node* is not a [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment), [DocumentType](https://dom.spec.whatwg.org/#documenttype), [Element](https://dom.spec.whatwg.org/#element), [Text](https://dom.spec.whatwg.org/#text), [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction), or [Comment](https://dom.spec.whatwg.org/#comment) [node](https://dom.spec.whatwg.org/#concept-node), [throw](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror).
5. If either *node* is a [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node) and *parent* is a [document](https://dom.spec.whatwg.org/#concept-document), or *node* is a [doctype](https://dom.spec.whatwg.org/#concept-doctype) and *parent* is not a [document](https://dom.spec.whatwg.org/#concept-document), [throw](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror).
6. If *parent* is a [document](https://dom.spec.whatwg.org/#concept-document), and any of the statements below, switched on *node*, are true, [throw](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror).

[DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [node](https://dom.spec.whatwg.org/#concept-node)

If *node* has more than one [element](https://dom.spec.whatwg.org/#concept-element) [child](https://dom.spec.whatwg.org/#concept-tree-child) or has a [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node) [child](https://dom.spec.whatwg.org/#concept-tree-child).

Otherwise, if *node* has one [element](https://dom.spec.whatwg.org/#concept-element) [child](https://dom.spec.whatwg.org/#concept-tree-child) and either *parent* has an [element](https://dom.spec.whatwg.org/#concept-element) [child](https://dom.spec.whatwg.org/#concept-tree-child) that is not *child* or a [doctype](https://dom.spec.whatwg.org/#concept-doctype) is [following](https://dom.spec.whatwg.org/#concept-tree-following) *child*.

[element](https://dom.spec.whatwg.org/#concept-element)

*parent* has an [element](https://dom.spec.whatwg.org/#concept-element) [child](https://dom.spec.whatwg.org/#concept-tree-child) that is not *child* or a [doctype](https://dom.spec.whatwg.org/#concept-doctype) is [following](https://dom.spec.whatwg.org/#concept-tree-following) *child*.

[doctype](https://dom.spec.whatwg.org/#concept-doctype)

*parent* has a [doctype](https://dom.spec.whatwg.org/#concept-doctype) [child](https://dom.spec.whatwg.org/#concept-tree-child) that is not *child*, or an [element](https://dom.spec.whatwg.org/#concept-element) is [preceding](https://dom.spec.whatwg.org/#concept-tree-preceding) *child*.

The above statements differ from the [pre-insert](https://dom.spec.whatwg.org/#concept-node-pre-insert) algorithm.

1. Let *reference child* be *child*’s [next sibling](https://dom.spec.whatwg.org/#concept-tree-next-sibling).
2. If *reference child* is *node*, set it to *node*’s [next sibling](https://dom.spec.whatwg.org/#concept-tree-next-sibling).
3. Let *previousSibling* be *child*’s [previous sibling](https://dom.spec.whatwg.org/#concept-tree-previous-sibling).
4. [Adopt](https://dom.spec.whatwg.org/#concept-node-adopt) *node* into *parent*’s [node document](https://dom.spec.whatwg.org/#concept-node-document).
5. Let *removedNodes* be the empty list.
6. If *child*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is not null, run these substeps:
   1. Set *removedNodes* to a list solely containing *child*.
   2. [Remove](https://dom.spec.whatwg.org/#concept-node-remove) *child* from its *parent* with the *suppress observers flag* set.

The above can only be false if *child* is *node*.

1. Let *nodes* be *node*’s [children](https://dom.spec.whatwg.org/#concept-tree-child) if *node* is a [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [node](https://dom.spec.whatwg.org/#concept-node), and a list containing solely *node* otherwise.
2. [Insert](https://dom.spec.whatwg.org/#concept-node-insert) *node* into *parent* before *reference child* with the *suppress observers flag* set.
3. [Queue a mutation record](https://dom.spec.whatwg.org/#queue-a-mutation-record) of "childList" for target *parent* with addedNodes *nodes*, removedNodes *removedNodes*, nextSibling *reference child*, and previousSibling *previousSibling*.
4. Return *child*.

To *replace all* with a *node* within a *parent*, run these steps:

1. If *node* is not null, [adopt](https://dom.spec.whatwg.org/#concept-node-adopt) *node* into *parent*’s [node document](https://dom.spec.whatwg.org/#concept-node-document).
2. Let *removedNodes* be *parent*’s [children](https://dom.spec.whatwg.org/#concept-tree-child).
3. Let *addedNodes* be the empty list if *node* is null, *node*’s [children](https://dom.spec.whatwg.org/#concept-tree-child) if *node* is a [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [node](https://dom.spec.whatwg.org/#concept-node), and a list containing *node* otherwise.
4. [Remove](https://dom.spec.whatwg.org/#concept-node-remove) all *parent*’s [children](https://dom.spec.whatwg.org/#concept-tree-child), in [tree order](https://dom.spec.whatwg.org/#concept-tree-order), with the *suppress observers flag* set.
5. If *node* is not null, [insert](https://dom.spec.whatwg.org/#concept-node-insert) *node* into *parent* before null with the *suppress observers flag* set.
6. [Queue a mutation record](https://dom.spec.whatwg.org/#queue-a-mutation-record) of "childList" for *parent* with addedNodes *addedNodes* and removedNodes *removedNodes*.

This algorithm does not make any checks with regards to the [node tree](https://dom.spec.whatwg.org/#concept-node-tree) constraints. Specification authors need to use it wisely.

To *pre-remove* a *child* from a *parent*, run these steps:

1. If *child*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is not *parent*, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotFoundError](https://heycam.github.io/webidl/#notfounderror).
2. [Remove](https://dom.spec.whatwg.org/#concept-node-remove) *child* from *parent*.
3. Return *child*.

[Specifications](https://dom.spec.whatwg.org/#other-applicable-specifications) may define *removing steps* for all or some [nodes](https://dom.spec.whatwg.org/#concept-node). The algorithm is passed *removedNode*, and optionally *oldParent*, as indicated in the [remove](https://dom.spec.whatwg.org/#concept-node-remove) algorithm below.

To *remove* a *node* from a *parent*, with an optional *suppress observers flag*, run these steps:

1. Let *index* be *node*’s [index](https://dom.spec.whatwg.org/#concept-tree-index).
2. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [start node](https://dom.spec.whatwg.org/#concept-range-start-node) is an [inclusive descendant](https://dom.spec.whatwg.org/#concept-tree-inclusive-descendant) of *node*, set its [start](https://dom.spec.whatwg.org/#concept-range-start) to (*parent*, *index*).
3. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [end node](https://dom.spec.whatwg.org/#concept-range-end-node) is an [inclusive descendant](https://dom.spec.whatwg.org/#concept-tree-inclusive-descendant) of *node*, set its [end](https://dom.spec.whatwg.org/#concept-range-end) to (*parent*, *index*).
4. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [start node](https://dom.spec.whatwg.org/#concept-range-start-node) is *parent* and [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) is greater than *index*, decrease its [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) by one.
5. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [end node](https://dom.spec.whatwg.org/#concept-range-end-node) is *parent* and [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset) is greater than *index*, decrease its [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset) by one.
6. For each [NodeIterator](https://dom.spec.whatwg.org/#nodeiterator) object *iterator* whose [root](https://dom.spec.whatwg.org/#concept-traversal-root)’s [node document](https://dom.spec.whatwg.org/#concept-node-document) is *node*’s [node document](https://dom.spec.whatwg.org/#concept-node-document), run the [NodeIterator pre-removing steps](https://dom.spec.whatwg.org/#nodeiterator-pre-removing-steps) given *node* and *iterator*.
7. Let *oldPreviousSibling* be *node*’s [previous sibling](https://dom.spec.whatwg.org/#concept-tree-previous-sibling).
8. Let *oldNextSibling* be *node*’s [next sibling](https://dom.spec.whatwg.org/#concept-tree-next-sibling).
9. Remove *node* from its *parent*.
10. If *node* is [assigned](https://dom.spec.whatwg.org/#slotable-assigned), then run [assign slotables](https://dom.spec.whatwg.org/#assign-slotables) for *node*’s [assigned slot](https://dom.spec.whatwg.org/#slotable-assigned-slot).
11. If *parent* is a [slot](https://dom.spec.whatwg.org/#concept-slot) whose [assigned nodes](https://dom.spec.whatwg.org/#slot-assigned-nodes) is the empty list, then run [signal a slot change](https://dom.spec.whatwg.org/#signal-a-slot-change) for *parent*.
12. If *node* has an [inclusive descendant](https://dom.spec.whatwg.org/#concept-tree-inclusive-descendant) that is a [slot](https://dom.spec.whatwg.org/#concept-slot), then:
    1. Run [assign slotables for a tree](https://dom.spec.whatwg.org/#assign-slotables-for-a-tree) with *parent*’s [tree](https://dom.spec.whatwg.org/#concept-tree).
    2. Run [assign slotables for a tree](https://dom.spec.whatwg.org/#assign-slotables-for-a-tree) with *node*’s [tree](https://dom.spec.whatwg.org/#concept-tree) and a set containing each [inclusive descendant](https://dom.spec.whatwg.org/#concept-tree-inclusive-descendant) of *node* that is a [slot](https://dom.spec.whatwg.org/#concept-slot).
13. Run the [removing steps](https://dom.spec.whatwg.org/#concept-node-remove-ext) with *node* and *parent*.
14. If *node* is [custom](https://dom.spec.whatwg.org/#concept-element-custom), then [enqueue a custom element callback reaction](https://html.spec.whatwg.org/multipage/scripting.html#enqueue-a-custom-element-callback-reaction) with *node*, callback name "disconnectedCallback", and an empty argument list.

It is intentional for now that [custom](https://dom.spec.whatwg.org/#concept-element-custom) [elements](https://dom.spec.whatwg.org/#concept-element) do not get *parent* passed. This might change in the future if there is a need.

1. For each [shadow-including descendant](https://dom.spec.whatwg.org/#concept-shadow-including-descendant) *descendant* of *node*, in [shadow-including tree order](https://dom.spec.whatwg.org/#concept-shadow-including-tree-order), run these substeps:
   1. Run the [removing steps](https://dom.spec.whatwg.org/#concept-node-remove-ext) with *descendant*.
   2. If *descendant* is [custom](https://dom.spec.whatwg.org/#concept-element-custom), then [enqueue a custom element callback reaction](https://html.spec.whatwg.org/multipage/scripting.html#enqueue-a-custom-element-callback-reaction) with *descendant*, callback name "disconnectedCallback", and an empty argument list.
2. For each [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) *inclusiveAncestor* of *parent*, if *inclusiveAncestor* has any [registered observers](https://dom.spec.whatwg.org/#registered-observer) whose **options**' [subtree](https://dom.spec.whatwg.org/#dom-mutationobserverinit-subtree) is true, then for each such [registered observer](https://dom.spec.whatwg.org/#registered-observer) *registered*, append a [transient registered observer](https://dom.spec.whatwg.org/#transient-registered-observer) whose **observer** and **options** are identical to those of *registered* and **source** which is *registered* to *node*’s list of [registered observers](https://dom.spec.whatwg.org/#registered-observer).
3. If *suppress observers flag* is unset, [queue a mutation record](https://dom.spec.whatwg.org/#queue-a-mutation-record) of "childList" for *parent* with removedNodes a list solely containing *node*, nextSibling *oldNextSibling*, and previousSibling *oldPreviousSibling*.

**4.2.4. Mixin** [**NonElementParentNode**](https://dom.spec.whatwg.org/#nonelementparentnode)

Web compatibility prevents the [getElementById()](https://dom.spec.whatwg.org/#dom-nonelementparentnode-getelementbyid) method from being exposed on [elements](https://dom.spec.whatwg.org/#concept-element) (and therefore on [ParentNode](https://dom.spec.whatwg.org/#parentnode)).

[NoInterfaceObject,

Exposed=Window]

interface *NonElementParentNode* {

[Element](https://dom.spec.whatwg.org/#element)? [getElementById](https://dom.spec.whatwg.org/#dom-nonelementparentnode-getelementbyid)(DOMString *elementId*);

};

[Document](https://dom.spec.whatwg.org/#document) implements [NonElementParentNode](https://dom.spec.whatwg.org/#nonelementparentnode);

[DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) implements [NonElementParentNode](https://dom.spec.whatwg.org/#nonelementparentnode);

*node* . [getElementById](https://dom.spec.whatwg.org/#dom-nonelementparentnode-getelementbyid)(*elementId*)

Returns the first [element](https://dom.spec.whatwg.org/#concept-element) within *node*’s [descendants](https://dom.spec.whatwg.org/#concept-tree-descendant) whose [ID](https://dom.spec.whatwg.org/#concept-id) is *elementId*.

The *getElementById(elementId)* method, when invoked, must return the first [element](https://dom.spec.whatwg.org/#concept-element), in [tree order](https://dom.spec.whatwg.org/#concept-tree-order), within [context object](https://dom.spec.whatwg.org/#context-object)’s [descendants](https://dom.spec.whatwg.org/#concept-tree-descendant), whose [ID](https://dom.spec.whatwg.org/#concept-id) is *elementId*, and null if there is no such [element](https://dom.spec.whatwg.org/#concept-element) otherwise.

**4.2.5. Mixin** [**DocumentOrShadowRoot**](https://dom.spec.whatwg.org/#documentorshadowroot)

[NoInterfaceObject,

Exposed=Window]

interface *DocumentOrShadowRoot* {

};

[Document](https://dom.spec.whatwg.org/#document) implements [DocumentOrShadowRoot](https://dom.spec.whatwg.org/#documentorshadowroot);

[ShadowRoot](https://dom.spec.whatwg.org/#shadowroot) implements [DocumentOrShadowRoot](https://dom.spec.whatwg.org/#documentorshadowroot);

The [DocumentOrShadowRoot](https://dom.spec.whatwg.org/#documentorshadowroot) mixin is expected to be used by other standards that want to define APIs shared between [documents](https://dom.spec.whatwg.org/#concept-document) and [shadow roots](https://dom.spec.whatwg.org/#concept-shadow-root).

**4.2.6. Mixin** [**ParentNode**](https://dom.spec.whatwg.org/#parentnode)

To *convert nodes into a node*, given *nodes* and *document*, run these steps:

1. Let *node* be null.
2. Replace each string in *nodes* with a new [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node) whose [data](https://dom.spec.whatwg.org/#concept-cd-data) is the string and [node document](https://dom.spec.whatwg.org/#concept-node-document) is *document*.
3. If *nodes* contains one [node](https://dom.spec.whatwg.org/#concept-node), set *node* to that [node](https://dom.spec.whatwg.org/#concept-node).
4. Otherwise, set *node* to a new [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) whose [node document](https://dom.spec.whatwg.org/#concept-node-document) is *document*, and then [append](https://dom.spec.whatwg.org/#concept-node-append) each [node](https://dom.spec.whatwg.org/#concept-node) in *nodes*, if any, to it. Rethrow any exceptions.
5. Return *node*.

[NoInterfaceObject,

Exposed=Window]

interface *ParentNode* {

[SameObject] readonly attribute [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) [children](https://dom.spec.whatwg.org/#dom-parentnode-children);

readonly attribute [Element](https://dom.spec.whatwg.org/#element)? [firstElementChild](https://dom.spec.whatwg.org/#dom-parentnode-firstelementchild);

readonly attribute [Element](https://dom.spec.whatwg.org/#element)? [lastElementChild](https://dom.spec.whatwg.org/#dom-parentnode-lastelementchild);

readonly attribute unsigned long [childElementCount](https://dom.spec.whatwg.org/#dom-parentnode-childelementcount);

[CEReactions, Unscopable] void [prepend](https://dom.spec.whatwg.org/#dom-parentnode-prepend)(([Node](https://dom.spec.whatwg.org/#node) or DOMString)... *nodes*);

[CEReactions, Unscopable] void [append](https://dom.spec.whatwg.org/#dom-parentnode-append)(([Node](https://dom.spec.whatwg.org/#node) or DOMString)... *nodes*);

[Element](https://dom.spec.whatwg.org/#element)? [querySelector](https://dom.spec.whatwg.org/#dom-parentnode-queryselector)(DOMString *selectors*);

[NewObject] [NodeList](https://dom.spec.whatwg.org/#nodelist) [querySelectorAll](https://dom.spec.whatwg.org/#dom-parentnode-queryselectorall)(DOMString *selectors*);

};

[Document](https://dom.spec.whatwg.org/#document) implements [ParentNode](https://dom.spec.whatwg.org/#parentnode);

[DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) implements [ParentNode](https://dom.spec.whatwg.org/#parentnode);

[Element](https://dom.spec.whatwg.org/#element) implements [ParentNode](https://dom.spec.whatwg.org/#parentnode);

*collection* = *node* . [children](https://dom.spec.whatwg.org/#dom-parentnode-children)

Returns the [child](https://dom.spec.whatwg.org/#concept-tree-child) [elements](https://dom.spec.whatwg.org/#concept-element).

*element* = *node* . [firstElementChild](https://dom.spec.whatwg.org/#dom-parentnode-firstelementchild)

Returns the first [child](https://dom.spec.whatwg.org/#concept-tree-child) that is an [element](https://dom.spec.whatwg.org/#concept-element), and null otherwise.

*element* = *node* . [lastElementChild](https://dom.spec.whatwg.org/#dom-parentnode-lastelementchild)

Returns the last [child](https://dom.spec.whatwg.org/#concept-tree-child) that is an [element](https://dom.spec.whatwg.org/#concept-element), and null otherwise.

*node* . [prepend](https://dom.spec.whatwg.org/#dom-parentnode-prepend)(*nodes*)

Inserts *nodes* before the [first child](https://dom.spec.whatwg.org/#concept-tree-first-child) of *node*, while replacing strings in *nodes* with equivalent [Text](https://dom.spec.whatwg.org/#text) [nodes](https://dom.spec.whatwg.org/#concept-node).

[Throws](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror) if the constraints of the [node tree](https://dom.spec.whatwg.org/#concept-node-tree) are violated.

*node* . [append](https://dom.spec.whatwg.org/#dom-parentnode-append)(*nodes*)

Inserts *nodes* after the [last child](https://dom.spec.whatwg.org/#concept-tree-last-child) of *node*, while replacing strings in *nodes* with equivalent [Text](https://dom.spec.whatwg.org/#text) [nodes](https://dom.spec.whatwg.org/#concept-node).

[Throws](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror) if the constraints of the [node tree](https://dom.spec.whatwg.org/#concept-node-tree) are violated.

*node* . [querySelector](https://dom.spec.whatwg.org/#dom-parentnode-queryselector)(*selectors*)

Returns the first [element](https://dom.spec.whatwg.org/#concept-element) that is a [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) of *node* that matches *selectors*.

*node* . [querySelectorAll](https://dom.spec.whatwg.org/#dom-parentnode-queryselectorall)(*selectors*)

Returns all [element](https://dom.spec.whatwg.org/#concept-element) [descendants](https://dom.spec.whatwg.org/#concept-tree-descendant) of *node* that match *selectors*.

The *children* attribute must return an [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) [collection](https://dom.spec.whatwg.org/#concept-collection) rooted at the [context object](https://dom.spec.whatwg.org/#context-object) matching only [element](https://dom.spec.whatwg.org/#concept-element) [children](https://dom.spec.whatwg.org/#concept-tree-child).

The *firstElementChild* attribute must return the first [child](https://dom.spec.whatwg.org/#concept-tree-child) that is an [element](https://dom.spec.whatwg.org/#concept-element), and null otherwise.

The *lastElementChild* attribute must return the last [child](https://dom.spec.whatwg.org/#concept-tree-child) that is an [element](https://dom.spec.whatwg.org/#concept-element), and null otherwise.

The *childElementCount* attribute must return the number of [children](https://dom.spec.whatwg.org/#concept-tree-child) of the [context object](https://dom.spec.whatwg.org/#context-object) that are [elements](https://dom.spec.whatwg.org/#concept-element).

The *prepend(nodes)* method must run these steps:

1. Let *node* be the result of [converting nodes into a node](https://dom.spec.whatwg.org/#converting-nodes-into-a-node) given *nodes* and [context object](https://dom.spec.whatwg.org/#context-object)’s [node document](https://dom.spec.whatwg.org/#concept-node-document). Rethrow any exceptions.
2. [Pre-insert](https://dom.spec.whatwg.org/#concept-node-pre-insert) *node* into [context object](https://dom.spec.whatwg.org/#context-object) before the [context object](https://dom.spec.whatwg.org/#context-object)’s [first child](https://dom.spec.whatwg.org/#concept-tree-first-child). Rethrow any exceptions.

The *append(nodes)* method must run these steps:

1. Let *node* be the result of [converting nodes into a node](https://dom.spec.whatwg.org/#converting-nodes-into-a-node) given *nodes* and [context object](https://dom.spec.whatwg.org/#context-object)’s [node document](https://dom.spec.whatwg.org/#concept-node-document). Rethrow any exceptions.
2. [Append](https://dom.spec.whatwg.org/#concept-node-append) *node* to [context object](https://dom.spec.whatwg.org/#context-object). Rethrow any exceptions.

The *querySelector(selectors)* method, when invoked, must return the first result of running [scope-match a selectors string](https://dom.spec.whatwg.org/#scope-match-a-selectors-string) *selectors* against the [context object](https://dom.spec.whatwg.org/#context-object), and null if the result is an empty list otherwise.

The *querySelectorAll(selectors)* method, when invoked, must return the [static](https://dom.spec.whatwg.org/#concept-collection-static) result of running [scope-match a selectors string](https://dom.spec.whatwg.org/#scope-match-a-selectors-string) *selectors* against the [context object](https://dom.spec.whatwg.org/#context-object).

**4.2.7. Mixin** [**NonDocumentTypeChildNode**](https://dom.spec.whatwg.org/#nondocumenttypechildnode)

Web compatibility prevents the [previousElementSibling](https://dom.spec.whatwg.org/#dom-nondocumenttypechildnode-previouselementsibling) and [nextElementSibling](https://dom.spec.whatwg.org/#dom-nondocumenttypechildnode-nextelementsibling) attributes from being exposed on [doctypes](https://dom.spec.whatwg.org/#concept-doctype) (and therefore on [ChildNode](https://dom.spec.whatwg.org/#childnode)).

[NoInterfaceObject,

Exposed=Window]

interface *NonDocumentTypeChildNode* {

readonly attribute [Element](https://dom.spec.whatwg.org/#element)? [previousElementSibling](https://dom.spec.whatwg.org/#dom-nondocumenttypechildnode-previouselementsibling);

readonly attribute [Element](https://dom.spec.whatwg.org/#element)? [nextElementSibling](https://dom.spec.whatwg.org/#dom-nondocumenttypechildnode-nextelementsibling);

};

[Element](https://dom.spec.whatwg.org/#element) implements [NonDocumentTypeChildNode](https://dom.spec.whatwg.org/#nondocumenttypechildnode);

[CharacterData](https://dom.spec.whatwg.org/#characterdata) implements [NonDocumentTypeChildNode](https://dom.spec.whatwg.org/#nondocumenttypechildnode);

*element* = *node* . [previousElementSibling](https://dom.spec.whatwg.org/#dom-nondocumenttypechildnode-previouselementsibling)

Returns the first [preceding](https://dom.spec.whatwg.org/#concept-tree-preceding) [sibling](https://dom.spec.whatwg.org/#concept-tree-sibling) that is an [element](https://dom.spec.whatwg.org/#concept-element), and null otherwise.

*element* = *node* . [nextElementSibling](https://dom.spec.whatwg.org/#dom-nondocumenttypechildnode-nextelementsibling)

Returns the first [following](https://dom.spec.whatwg.org/#concept-tree-following) [sibling](https://dom.spec.whatwg.org/#concept-tree-sibling) that is an [element](https://dom.spec.whatwg.org/#concept-element), and null otherwise.

The *previousElementSibling* attribute must return the first [preceding](https://dom.spec.whatwg.org/#concept-tree-preceding) [sibling](https://dom.spec.whatwg.org/#concept-tree-sibling) that is an [element](https://dom.spec.whatwg.org/#concept-element), and null otherwise.

The *nextElementSibling* attribute must return the first [following](https://dom.spec.whatwg.org/#concept-tree-following) [sibling](https://dom.spec.whatwg.org/#concept-tree-sibling) that is an [element](https://dom.spec.whatwg.org/#concept-element), and null otherwise.

**4.2.8. Mixin** [**ChildNode**](https://dom.spec.whatwg.org/#childnode)

[NoInterfaceObject,

Exposed=Window]

interface *ChildNode* {

[CEReactions, Unscopable] void [before](https://dom.spec.whatwg.org/#dom-childnode-before)(([Node](https://dom.spec.whatwg.org/#node) or DOMString)... *nodes*);

[CEReactions, Unscopable] void [after](https://dom.spec.whatwg.org/#dom-childnode-after)(([Node](https://dom.spec.whatwg.org/#node) or DOMString)... *nodes*);

[CEReactions, Unscopable] void [replaceWith](https://dom.spec.whatwg.org/#dom-childnode-replacewith)(([Node](https://dom.spec.whatwg.org/#node) or DOMString)... *nodes*);

[CEReactions, Unscopable] void [remove](https://dom.spec.whatwg.org/#dom-childnode-remove)();

};

[DocumentType](https://dom.spec.whatwg.org/#documenttype) implements [ChildNode](https://dom.spec.whatwg.org/#childnode);

[Element](https://dom.spec.whatwg.org/#element) implements [ChildNode](https://dom.spec.whatwg.org/#childnode);

[CharacterData](https://dom.spec.whatwg.org/#characterdata) implements [ChildNode](https://dom.spec.whatwg.org/#childnode);

*node* . [before(nodes)](https://dom.spec.whatwg.org/#dom-childnode-before)

Inserts *nodes* just before *node*, while replacing strings in *nodes* with equivalent [Text](https://dom.spec.whatwg.org/#text) [nodes](https://dom.spec.whatwg.org/#concept-node).

[Throws](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror) if the constraints of the [node tree](https://dom.spec.whatwg.org/#concept-node-tree) are violated.

*node* . [after(nodes)](https://dom.spec.whatwg.org/#dom-childnode-after)

Inserts *nodes* just after *node*, while replacing strings in *nodes* with equivalent [Text](https://dom.spec.whatwg.org/#text) [nodes](https://dom.spec.whatwg.org/#concept-node).

[Throws](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror) if the constraints of the [node tree](https://dom.spec.whatwg.org/#concept-node-tree) are violated.

*node* . [replaceWith(nodes)](https://dom.spec.whatwg.org/#dom-childnode-replacewith)

Replaces *node* with *nodes*, while replacing strings in *nodes* with equivalent [Text](https://dom.spec.whatwg.org/#text) [nodes](https://dom.spec.whatwg.org/#concept-node).

[Throws](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror) if the constraints of the [node tree](https://dom.spec.whatwg.org/#concept-node-tree) are violated.

*node* . [remove()](https://dom.spec.whatwg.org/#dom-childnode-remove)

Removes *node*.

The *before(nodes)* method, when invoked, must run these steps:

1. Let *parent* be [context object](https://dom.spec.whatwg.org/#context-object)’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
2. If *parent* is null, terminate these steps.
3. Let *viablePreviousSibling* be [context object](https://dom.spec.whatwg.org/#context-object)’s first [preceding](https://dom.spec.whatwg.org/#concept-tree-preceding) [sibling](https://dom.spec.whatwg.org/#concept-tree-sibling) not in *nodes*, and null otherwise.
4. Let *node* be the result of [converting nodes into a node](https://dom.spec.whatwg.org/#converting-nodes-into-a-node), given *nodes* and [context object](https://dom.spec.whatwg.org/#context-object)’s [node document](https://dom.spec.whatwg.org/#concept-node-document). Rethrow any exceptions.
5. If *viablePreviousSibling* is null, set it to *parent*’s [first child](https://dom.spec.whatwg.org/#concept-tree-first-child), and to *viablePreviousSibling*’s [next sibling](https://dom.spec.whatwg.org/#concept-tree-next-sibling) otherwise.
6. [Pre-insert](https://dom.spec.whatwg.org/#concept-node-pre-insert) *node* into *parent* before *viablePreviousSibling*. Rethrow any exceptions.

The *after(nodes)* method, when invoked, must run these steps:

1. Let *parent* be [context object](https://dom.spec.whatwg.org/#context-object)’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
2. If *parent* is null, terminate these steps.
3. Let *viableNextSibling* be [context object](https://dom.spec.whatwg.org/#context-object)’s first [following](https://dom.spec.whatwg.org/#concept-tree-following) [sibling](https://dom.spec.whatwg.org/#concept-tree-sibling) not in *nodes*, and null otherwise.
4. Let *node* be the result of [converting nodes into a node](https://dom.spec.whatwg.org/#converting-nodes-into-a-node), given *nodes* and [context object](https://dom.spec.whatwg.org/#context-object)’s [node document](https://dom.spec.whatwg.org/#concept-node-document). Rethrow any exceptions.
5. [Pre-insert](https://dom.spec.whatwg.org/#concept-node-pre-insert) *node* into *parent* before *viableNextSibling*. Rethrow any exceptions.

The *replaceWith(nodes)* method, when invoked, must run these steps:

1. Let *parent* be [context object](https://dom.spec.whatwg.org/#context-object)’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
2. If *parent* is null, terminate these steps.
3. Let *viableNextSibling* be [context object](https://dom.spec.whatwg.org/#context-object)’s first [following](https://dom.spec.whatwg.org/#concept-tree-following) [sibling](https://dom.spec.whatwg.org/#concept-tree-sibling) not in *nodes*, and null otherwise.
4. Let *node* be the result of [converting nodes into a node](https://dom.spec.whatwg.org/#converting-nodes-into-a-node), given *nodes* and [context object](https://dom.spec.whatwg.org/#context-object)’s [node document](https://dom.spec.whatwg.org/#concept-node-document). Rethrow any exceptions.
5. If [context object](https://dom.spec.whatwg.org/#context-object)’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is *parent*, [replace](https://dom.spec.whatwg.org/#concept-node-replace) the [context object](https://dom.spec.whatwg.org/#context-object) with *node* within *parent*. Rethrow any exceptions.

[Context object](https://dom.spec.whatwg.org/#context-object) could have been inserted into *node*.

1. Otherwise, [pre-insert](https://dom.spec.whatwg.org/#concept-node-pre-insert) *node* into *parent* before *viableNextSibling*. Rethrow any exceptions.

The *remove()* method, when invoked, must run these steps:

1. If [context object](https://dom.spec.whatwg.org/#context-object)’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is null, terminate these steps.
2. [Remove](https://dom.spec.whatwg.org/#concept-node-remove) the [context object](https://dom.spec.whatwg.org/#context-object) from [context object](https://dom.spec.whatwg.org/#context-object)’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent).

**4.2.9. Mixin:** [**Slotable**](https://dom.spec.whatwg.org/#slotable)

[NoInterfaceObject,

Exposed=Window]

interface *Slotable* {

readonly attribute [HTMLSlotElement](https://html.spec.whatwg.org/multipage/scripting.html#htmlslotelement)? [assignedSlot](https://dom.spec.whatwg.org/#dom-slotable-assignedslot);

};

[Element](https://dom.spec.whatwg.org/#element) implements [Slotable](https://dom.spec.whatwg.org/#slotable);

[Text](https://dom.spec.whatwg.org/#text) implements [Slotable](https://dom.spec.whatwg.org/#slotable);

The *assignedSlot* attribute’s getter must return the result of [find a slot](https://dom.spec.whatwg.org/#find-a-slot) given [context object](https://dom.spec.whatwg.org/#context-object) and with the *open flag* set.

**4.2.10. Old-style collections:** [**NodeList**](https://dom.spec.whatwg.org/#nodelist) **and** [**HTMLCollection**](https://dom.spec.whatwg.org/#htmlcollection)

A *collection* is an object that represents a lists of DOM nodes. A [collection](https://dom.spec.whatwg.org/#concept-collection) can be either *live* or *static*. Unless otherwise stated, a [collection](https://dom.spec.whatwg.org/#concept-collection) must be [live](https://dom.spec.whatwg.org/#concept-collection-live).

If a [collection](https://dom.spec.whatwg.org/#concept-collection) is [live](https://dom.spec.whatwg.org/#concept-collection-live), then the attributes and methods on that object must operate on the actual underlying data, not a snapshot of the data.

When a [collection](https://dom.spec.whatwg.org/#concept-collection) is created, a filter and a root are associated with it.

The [collection](https://dom.spec.whatwg.org/#concept-collection) then *represents* a view of the subtree rooted at the [collection’s](https://dom.spec.whatwg.org/#concept-collection) root, containing only nodes that match the given filter. The view is linear. In the absence of specific requirements to the contrary, the nodes within the [collection](https://dom.spec.whatwg.org/#concept-collection) must be sorted in [tree order](https://dom.spec.whatwg.org/#concept-tree-order).

**4.2.10.1. Interface** [**NodeList**](https://dom.spec.whatwg.org/#nodelist)

A [NodeList](https://dom.spec.whatwg.org/#nodelist) object is a [collection](https://dom.spec.whatwg.org/#concept-collection) of [nodes](https://dom.spec.whatwg.org/#concept-node).

[Exposed=Window]

interface *NodeList* {

getter [Node](https://dom.spec.whatwg.org/#node)? [item](https://dom.spec.whatwg.org/#dom-nodelist-item)(unsigned long *index*);

readonly attribute unsigned long [length](https://dom.spec.whatwg.org/#dom-nodelist-length);

iterable<[Node](https://dom.spec.whatwg.org/#node)>;

};

*collection* . [length](https://dom.spec.whatwg.org/#dom-nodelist-length)

Returns the number of [nodes](https://dom.spec.whatwg.org/#concept-node) in the [collection](https://dom.spec.whatwg.org/#concept-collection).

*element* = *collection* . [item(index)](https://dom.spec.whatwg.org/#dom-nodelist-item)

*element* = *collection*[*index*]

Returns the [node](https://dom.spec.whatwg.org/#concept-node) with index *index* from the [collection](https://dom.spec.whatwg.org/#concept-collection). The [nodes](https://dom.spec.whatwg.org/#concept-node) are sorted in [tree order](https://dom.spec.whatwg.org/#concept-tree-order).

The object’s [supported property indices](https://heycam.github.io/webidl/#dfn-supported-property-indices) are the numbers in the range zero to one less than the number of nodes [represented by the collection](https://dom.spec.whatwg.org/#represented-by-the-collection). If there are no such elements, then there are no [supported property indices](https://heycam.github.io/webidl/#dfn-supported-property-indices).

The *length* attribute must return the number of nodes [represented by the collection](https://dom.spec.whatwg.org/#represented-by-the-collection).

The *item(index)* method must return the *index*th [node](https://dom.spec.whatwg.org/#concept-node) in the [collection](https://dom.spec.whatwg.org/#concept-collection). If there is no *index*th [node](https://dom.spec.whatwg.org/#concept-node) in the [collection](https://dom.spec.whatwg.org/#concept-collection), then the method must return null.

**4.2.10.2. Interface** [**HTMLCollection**](https://dom.spec.whatwg.org/#htmlcollection)

[Exposed=Window, LegacyUnenumerableNamedProperties]

interface *HTMLCollection* {

readonly attribute unsigned long [length](https://dom.spec.whatwg.org/#dom-htmlcollection-length);

getter [Element](https://dom.spec.whatwg.org/#element)? [item](https://dom.spec.whatwg.org/#dom-htmlcollection-item)(unsigned long *index*);

getter [Element](https://dom.spec.whatwg.org/#element)? [namedItem](https://dom.spec.whatwg.org/#dom-htmlcollection-nameditem)(DOMString *name*);

};

An [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) object is a [collection](https://dom.spec.whatwg.org/#concept-collection) of [elements](https://dom.spec.whatwg.org/#concept-element).

[HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) is an historical artifact we cannot rid the web of. While developers are of course welcome to keep using it, new API standard designers ought not to use it (use sequence<T> in IDL instead).

*collection* . [length](https://dom.spec.whatwg.org/#dom-htmlcollection-length)

Returns the number of [elements](https://dom.spec.whatwg.org/#concept-element) in the [collection](https://dom.spec.whatwg.org/#concept-collection).

*element* = *collection* . [item(index)](https://dom.spec.whatwg.org/#dom-htmlcollection-item)

*element* = *collection*[*index*]

Returns the [element](https://dom.spec.whatwg.org/#concept-element) with index *index* from the [collection](https://dom.spec.whatwg.org/#concept-collection). The [elements](https://dom.spec.whatwg.org/#concept-element) are sorted in [tree order](https://dom.spec.whatwg.org/#concept-tree-order).

*element* = *collection* . [namedItem(name)](https://dom.spec.whatwg.org/#dom-htmlcollection-nameditem)

*element* = *collection*[*name*]

Returns the first [element](https://dom.spec.whatwg.org/#concept-element) with [ID](https://dom.spec.whatwg.org/#concept-id) or name *name* from the collection.

The object’s [supported property indices](https://heycam.github.io/webidl/#dfn-supported-property-indices) are the numbers in the range zero to one less than the number of elements [represented by the collection](https://dom.spec.whatwg.org/#represented-by-the-collection). If there are no such elements, then there are no [supported property indices](https://heycam.github.io/webidl/#dfn-supported-property-indices).

The *length* attribute’s getter must return the number of nodes [represented by the collection](https://dom.spec.whatwg.org/#represented-by-the-collection).

The *item(index)* method, when invoked, must return the *index*th [element](https://dom.spec.whatwg.org/#concept-element) in the [collection](https://dom.spec.whatwg.org/#concept-collection). If there is no *index*th [element](https://dom.spec.whatwg.org/#concept-element) in the [collection](https://dom.spec.whatwg.org/#concept-collection), then the method must return null.

The [supported property names](https://heycam.github.io/webidl/#dfn-supported-property-names) are the values from the list returned by these steps:

1. Let *result* be an empty list.
2. For each *element* [represented by the collection](https://dom.spec.whatwg.org/#represented-by-the-collection), in [tree order](https://dom.spec.whatwg.org/#concept-tree-order), run these substeps:
   1. If *element* has an [ID](https://dom.spec.whatwg.org/#concept-id) which is not in *result*, append *element*’s [ID](https://dom.spec.whatwg.org/#concept-id) to *result*.
   2. If *element* is in the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace) and [has](https://dom.spec.whatwg.org/#concept-element-attribute-has) a [name attribute](https://dom.spec.whatwg.org/#concept-named-attribute) whose [value](https://dom.spec.whatwg.org/#concept-attribute-value) is neither the empty string nor is in *result*, append *element*’s [name attribute](https://dom.spec.whatwg.org/#concept-named-attribute) [value](https://dom.spec.whatwg.org/#concept-attribute-value) to *result*.
3. Return *result*.

The *namedItem(key)* method, when invoked, must run these steps:

1. If *key* is the empty string, return null.
2. Return the first [element](https://dom.spec.whatwg.org/#concept-element) in the [collection](https://dom.spec.whatwg.org/#concept-collection) for which at least one of the following is true:
   * it has an [ID](https://dom.spec.whatwg.org/#concept-id) which is *key*;
   * it is in the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace) and [has](https://dom.spec.whatwg.org/#concept-element-attribute-has) a [name attribute](https://dom.spec.whatwg.org/#concept-named-attribute) whose [value](https://dom.spec.whatwg.org/#concept-attribute-value) is *key*;

or null if there is no such [element](https://dom.spec.whatwg.org/#concept-element).

**4.3. Mutation observers**

Each [unit of related similar-origin browsing contexts](https://html.spec.whatwg.org/multipage/browsers.html#unit-of-related-similar-origin-browsing-contexts) has a *mutation observer compound microtask queued flag*, which is initially unset, and an associated list of [MutationObserver](https://dom.spec.whatwg.org/#mutationobserver) objects, which is initially empty. [[HTML]](https://dom.spec.whatwg.org/#biblio-html)

To *queue a mutation observer compound microtask*, run these steps:

1. If [mutation observer compound microtask queued flag](https://dom.spec.whatwg.org/#mutation-observer-compound-microtask-queued-flag) is set, terminate these steps.
2. Set [mutation observer compound microtask queued flag](https://dom.spec.whatwg.org/#mutation-observer-compound-microtask-queued-flag).
3. [Queue](https://html.spec.whatwg.org/multipage/webappapis.html#queue-a-microtask) a [compound microtask](https://html.spec.whatwg.org/multipage/webappapis.html#compound-microtask) to [notify mutation observers](https://dom.spec.whatwg.org/#notify-mutation-observers).

To *notify mutation observers*, run these steps:

1. Unset [mutation observer compound microtask queued flag](https://dom.spec.whatwg.org/#mutation-observer-compound-microtask-queued-flag).
2. Let *notify list* be a copy of [unit of related similar-origin browsing contexts](https://html.spec.whatwg.org/multipage/browsers.html#unit-of-related-similar-origin-browsing-contexts)' list of [MutationObserver](https://dom.spec.whatwg.org/#mutationobserver) objects.
3. For each [MutationObserver](https://dom.spec.whatwg.org/#mutationobserver) object *mo* in *notify list*, [execute a compound microtask subtask](https://html.spec.whatwg.org/multipage/webappapis.html#execute-a-compound-microtask-subtask) to run these steps: [[HTML]](https://dom.spec.whatwg.org/#biblio-html)
   1. Let *queue* be a copy of *mo*’s [record queue](https://dom.spec.whatwg.org/#concept-mo-queue).
   2. Empty *mo*’s [record queue](https://dom.spec.whatwg.org/#concept-mo-queue).
   3. Remove all [transient registered observers](https://dom.spec.whatwg.org/#transient-registered-observer) whose **observer** is *mo*.
   4. If *queue* is non-empty, call *mo*’s [callback](https://dom.spec.whatwg.org/#concept-mo-callback) with *queue* as first argument, and *mo* (itself) as second argument and [callback this value](https://heycam.github.io/webidl/#dfn-callback-this-value). If this throws an exception, [report the exception](https://html.spec.whatwg.org/multipage/webappapis.html#report-the-exception).
4. Let *signalList* be a copy of [unit of related similar-origin browsing contexts](https://html.spec.whatwg.org/multipage/browsers.html#unit-of-related-similar-origin-browsing-contexts)' [signal slot list](https://dom.spec.whatwg.org/#signal-slot-list).
5. Empty [unit of related similar-origin browsing contexts](https://html.spec.whatwg.org/multipage/browsers.html#unit-of-related-similar-origin-browsing-contexts)' [signal slot list](https://dom.spec.whatwg.org/#signal-slot-list).
6. For each [slot](https://dom.spec.whatwg.org/#concept-slot) *slot* in *signalList*, in order, [fire an event](https://dom.spec.whatwg.org/#concept-event-fire) named slotchange, with its [bubbles](https://dom.spec.whatwg.org/#dom-event-bubbles) attribute set to true, at *slot*.

Each [node](https://dom.spec.whatwg.org/#concept-node) has an associated list of [registered observers](https://dom.spec.whatwg.org/#registered-observer).

A *registered observer* consists of an **observer** (a [MutationObserver](https://dom.spec.whatwg.org/#mutationobserver) object) and **options** (a [MutationObserverInit](https://dom.spec.whatwg.org/#dictdef-mutationobserverinit) dictionary). A *transient registered observer* is a specific type of [registered observer](https://dom.spec.whatwg.org/#registered-observer) that has a **source** which is a [registered observer](https://dom.spec.whatwg.org/#registered-observer).

[Transient registered observers](https://dom.spec.whatwg.org/#transient-registered-observer) are used to track mutations within a given [node](https://dom.spec.whatwg.org/#concept-node)’s [descendants](https://dom.spec.whatwg.org/#concept-tree-descendant) after [node](https://dom.spec.whatwg.org/#concept-node) has been removed so they do not get lost when subtree is set to true on [node](https://dom.spec.whatwg.org/#concept-node)’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent).

**4.3.1. Interface** [**MutationObserver**](https://dom.spec.whatwg.org/#mutationobserver)

[[Constructor](https://dom.spec.whatwg.org/#dom-mutationobserver-mutationobserver)([MutationCallback](https://dom.spec.whatwg.org/#callbackdef-mutationcallback) callback)]

interface *MutationObserver* {

void [observe](https://dom.spec.whatwg.org/#dom-mutationobserver-observe)([Node](https://dom.spec.whatwg.org/#node) *target*, [MutationObserverInit](https://dom.spec.whatwg.org/#dictdef-mutationobserverinit) *options*);

void [disconnect](https://dom.spec.whatwg.org/#dom-mutationobserver-disconnect)();

sequence<[MutationRecord](https://dom.spec.whatwg.org/#mutationrecord)> [takeRecords](https://dom.spec.whatwg.org/#dom-mutationobserver-takerecords)();

};

callback *MutationCallback* = void (sequence<[MutationRecord](https://dom.spec.whatwg.org/#mutationrecord)> *mutations*, [MutationObserver](https://dom.spec.whatwg.org/#mutationobserver) *observer*);

dictionary *MutationObserverInit* {

boolean *childList* = false;

boolean *attributes*;

boolean *characterData*;

boolean *subtree* = false;

boolean *attributeOldValue*;

boolean *characterDataOldValue*;

sequence<DOMString> *attributeFilter*;

};

A [MutationObserver](https://dom.spec.whatwg.org/#mutationobserver) object can be used to observe mutations to the [tree](https://dom.spec.whatwg.org/#concept-tree) of [nodes](https://dom.spec.whatwg.org/#concept-node).

Each [MutationObserver](https://dom.spec.whatwg.org/#mutationobserver) object has these associated concepts:

* A *callback* set on creation.
* A list of [nodes](https://dom.spec.whatwg.org/#concept-node) on which it is a [registered observer](https://dom.spec.whatwg.org/#registered-observer)’s **observer** that is initially empty.
* A list of [MutationRecord](https://dom.spec.whatwg.org/#mutationrecord) objects called the *record queue* that is initially empty.

*observer* = new [MutationObserver(callback)](https://dom.spec.whatwg.org/#dom-mutationobserver-mutationobserver)

Constructs a [MutationObserver](https://dom.spec.whatwg.org/#mutationobserver) object and sets its [callback](https://dom.spec.whatwg.org/#concept-mo-callback) to *callback*. The *callback* is invoked with a list of [MutationRecord](https://dom.spec.whatwg.org/#mutationrecord) objects as first argument and the constructed [MutationObserver](https://dom.spec.whatwg.org/#mutationobserver) object as second argument. It is invoked after [nodes](https://dom.spec.whatwg.org/#concept-node) registered with the [observe()](https://dom.spec.whatwg.org/#dom-mutationobserver-observe) method, are mutated.

*observer* . [observe(target, options)](https://dom.spec.whatwg.org/#dom-mutationobserver-observe)

Instructs the user agent to observe a given *target* (a [node](https://dom.spec.whatwg.org/#concept-node)) and report any mutations based on the criteria given by *options* (an object).

The *options* argument allows for setting mutation observation options via object members. These are the object members that can be used:

[childList](https://dom.spec.whatwg.org/#dom-mutationobserverinit-childlist)

Set to true if mutations to *target*’s [children](https://dom.spec.whatwg.org/#concept-tree-child) are to be observed.

[attributes](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributes)

Set to true if mutations to *target*’s [attributes](https://dom.spec.whatwg.org/#concept-attribute) are to be observed. Can be omitted if [attributeOldValue](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributeoldvalue) and/or [attributeFilter](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributefilter) is specified.

[characterData](https://dom.spec.whatwg.org/#dom-mutationobserverinit-characterdata)

Set to true if mutations to *target*’s [data](https://dom.spec.whatwg.org/#concept-cd-data) are to be observed. Can be omitted if [characterDataOldValue](https://dom.spec.whatwg.org/#dom-mutationobserverinit-characterdataoldvalue) is specified.

[subtree](https://dom.spec.whatwg.org/#dom-mutationobserverinit-subtree)

Set to true if mutations to not just *target*, but also *target*’s [descendants](https://dom.spec.whatwg.org/#concept-tree-descendant) are to be observed.

[attributeOldValue](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributeoldvalue)

Set to true if [attributes](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributes) is true or omitted and *target*’s [attribute](https://dom.spec.whatwg.org/#concept-attribute) [value](https://dom.spec.whatwg.org/#concept-attribute-value) before the mutation needs to be recorded.

[characterDataOldValue](https://dom.spec.whatwg.org/#dom-mutationobserverinit-characterdataoldvalue)

Set to true if [characterData](https://dom.spec.whatwg.org/#dom-mutationobserverinit-characterdata) is set to true or omitted and *target*’s [data](https://dom.spec.whatwg.org/#concept-cd-data) before the mutation needs to be recorded.

[attributeFilter](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributefilter)

Set to a list of [attribute](https://dom.spec.whatwg.org/#concept-attribute) [local names](https://dom.spec.whatwg.org/#concept-attribute-local-name) (without [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace)) if not all [attribute](https://dom.spec.whatwg.org/#concept-attribute) mutations need to be observed and [attributes](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributes) is true or omitted.

*observer* . [disconnect()](https://dom.spec.whatwg.org/#dom-mutationobserver-disconnect)

Stops *observer* from observing any mutations. Until the [observe()](https://dom.spec.whatwg.org/#dom-mutationobserver-observe) method is used again, *observer*’s [callback](https://dom.spec.whatwg.org/#concept-mo-callback) will not be invoked.

*observer* . [takeRecords()](https://dom.spec.whatwg.org/#dom-mutationobserver-takerecords)

Empties the [record queue](https://dom.spec.whatwg.org/#concept-mo-queue) and returns what was in there.

The *MutationObserver(callback)* constructor must create a new [MutationObserver](https://dom.spec.whatwg.org/#mutationobserver) object with [callback](https://dom.spec.whatwg.org/#concept-mo-callback) set to *callback*, append it to the [unit of related similar-origin browsing contexts](https://html.spec.whatwg.org/multipage/browsers.html#unit-of-related-similar-origin-browsing-contexts)' list of [MutationObserver](https://dom.spec.whatwg.org/#mutationobserver) objects, and then return it.

The *observe(target, options)* method, when invoked, must run these steps:

1. If either *options*’ [attributeOldValue](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributeoldvalue) or [attributeFilter](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributefilter) is present and *options*’ [attributes](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributes) is omitted, set *options*’ [attributes](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributes) to true.
2. If *options*’ [characterDataOldValue](https://dom.spec.whatwg.org/#dom-mutationobserverinit-characterdataoldvalue) is present and *options*’ [characterData](https://dom.spec.whatwg.org/#dom-mutationobserverinit-characterdata) is omitted, set *options*’ [characterData](https://dom.spec.whatwg.org/#dom-mutationobserverinit-characterdata) to true.
3. If none of *options*’ [childList](https://dom.spec.whatwg.org/#dom-mutationobserverinit-childlist) [attributes](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributes), and [characterData](https://dom.spec.whatwg.org/#dom-mutationobserverinit-characterdata) is true, [throw](https://heycam.github.io/webidl/#dfn-throw) a TypeError.
4. If *options*’ [attributeOldValue](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributeoldvalue) is true and *options*’ [attributes](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributes) is false, [throw](https://heycam.github.io/webidl/#dfn-throw) a TypeError.
5. If *options*’ [attributeFilter](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributefilter) is present and *options*’ [attributes](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributes) is false, [throw](https://heycam.github.io/webidl/#dfn-throw) a TypeError.
6. If *options*’ [characterDataOldValue](https://dom.spec.whatwg.org/#dom-mutationobserverinit-characterdataoldvalue) is true and *options*’ [characterData](https://dom.spec.whatwg.org/#dom-mutationobserverinit-characterdata) is false, [throw](https://heycam.github.io/webidl/#dfn-throw) a TypeError.
7. For each [registered observer](https://dom.spec.whatwg.org/#registered-observer) *registered* in *target*’s list of [registered observers](https://dom.spec.whatwg.org/#registered-observer) whose **observer** is the [context object](https://dom.spec.whatwg.org/#context-object):
   1. Remove all [transient registered observers](https://dom.spec.whatwg.org/#transient-registered-observer) whose **source** is *registered*.
   2. Replace *registered*’s **options** with *options*.
8. Otherwise, add a new [registered observer](https://dom.spec.whatwg.org/#registered-observer) to *target*’s list of [registered observers](https://dom.spec.whatwg.org/#registered-observer) with the [context object](https://dom.spec.whatwg.org/#context-object) as the **observer** and *options* as the **options**, and add *target* to [context object](https://dom.spec.whatwg.org/#context-object)’s list of [nodes](https://dom.spec.whatwg.org/#concept-node) on which it is registered.

The *disconnect()* method must, for each [node](https://dom.spec.whatwg.org/#concept-node) *node* in the [context object](https://dom.spec.whatwg.org/#context-object)’s list of [nodes](https://dom.spec.whatwg.org/#concept-node), remove any [registered observer](https://dom.spec.whatwg.org/#registered-observer) on *node* for which the [context object](https://dom.spec.whatwg.org/#context-object) is the **observer**, and also empty [context object](https://dom.spec.whatwg.org/#context-object)’s [record queue](https://dom.spec.whatwg.org/#concept-mo-queue).

The *takeRecords()* method must return a copy of the [record queue](https://dom.spec.whatwg.org/#concept-mo-queue) and then empty the [record queue](https://dom.spec.whatwg.org/#concept-mo-queue).

**4.3.2. Queuing a mutation record**

To *queue a mutation record* of *type* for *target* with one or more of (depends on *type*) name *name*, namespace *namespace*, oldValue *oldValue*, addedNodes *addedNodes*, removedNodes *removedNodes*, previousSibling *previousSibling*, and nextSibling *nextSibling*, run these steps:

1. Let *interested observers* be an initially empty set of [MutationObserver](https://dom.spec.whatwg.org/#mutationobserver) objects optionally paired with a string.
2. Let *nodes* be the [inclusive ancestors](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of *target*.
3. Then, for each *node* in *nodes*, and then for each *registered observer* (with *registered observer*’s **options** as *options*) in *node*’s list of [registered observers](https://dom.spec.whatwg.org/#registered-observer), run these substeps:
   1. If none of the following are true
      * *node* is not *target* and *options*’ subtree is false
      * *type* is "attributes" and *options*’ attributes is not true
      * *type* is "attributes", *options*’ attributeFilter is present, and *options*’ attributeFilter does not contain *name* or *namespace* is non-null
      * *type* is "characterData" and *options*’ characterData is not true
      * *type* is "childList" and *options*’ childList is false

then run these subsubsteps:

* + - If *registered observer*’s **observer** is not in *interested observers*, append *registered observer*’s **observer** to *interested observers*.
    - If either *type* is "attributes" and *options*’ attributeOldValue is true, or *type* is "characterData" and *options*’ characterDataOldValue is true, set the paired string of *registered observer*’s **observer** in *interested observers* to *oldValue*.

1. Then, for each *observer* in *interested observers*, run these substeps:
   1. Let *record* be a new [MutationRecord](https://dom.spec.whatwg.org/#mutationrecord) object with its [type](https://dom.spec.whatwg.org/#dom-mutationrecord-type) set to *type* and [target](https://dom.spec.whatwg.org/#dom-mutationrecord-target) set to *target*.
   2. If *name* and *namespace* are given, set *record*’s [attributeName](https://dom.spec.whatwg.org/#dom-mutationrecord-attributename) to *name*, and *record*’s [attributeNamespace](https://dom.spec.whatwg.org/#dom-mutationrecord-attributenamespace) to *namespace*.
   3. If *addedNodes* is given, set *record*’s [addedNodes](https://dom.spec.whatwg.org/#dom-mutationrecord-addednodes) to *addedNodes*.
   4. If *removedNodes* is given, set *record*’s [removedNodes](https://dom.spec.whatwg.org/#dom-mutationrecord-removednodes) to *removedNodes*,
   5. If *previousSibling* is given, set *record*’s [previousSibling](https://dom.spec.whatwg.org/#dom-mutationrecord-previoussibling) to *previousSibling*.
   6. If *nextSibling* is given, set *record*’s [nextSibling](https://dom.spec.whatwg.org/#dom-mutationrecord-nextsibling) to *nextSibling*.
   7. If *observer* has a paired string, set *record*’s [oldValue](https://dom.spec.whatwg.org/#dom-mutationrecord-oldvalue) to *observer*’s paired string.
   8. Append *record* to *observer*’s [record queue](https://dom.spec.whatwg.org/#concept-mo-queue).
2. [Queue a mutation observer compound microtask](https://dom.spec.whatwg.org/#queue-a-mutation-observer-compound-microtask).

**4.3.3. Interface** [**MutationRecord**](https://dom.spec.whatwg.org/#mutationrecord)

[Exposed=Window]

interface *MutationRecord* {

readonly attribute DOMString [type](https://dom.spec.whatwg.org/#dom-mutationrecord-type);

[SameObject] readonly attribute [Node](https://dom.spec.whatwg.org/#node) [target](https://dom.spec.whatwg.org/#dom-mutationrecord-target);

[SameObject] readonly attribute [NodeList](https://dom.spec.whatwg.org/#nodelist) [addedNodes](https://dom.spec.whatwg.org/#dom-mutationrecord-addednodes);

[SameObject] readonly attribute [NodeList](https://dom.spec.whatwg.org/#nodelist) [removedNodes](https://dom.spec.whatwg.org/#dom-mutationrecord-removednodes);

readonly attribute [Node](https://dom.spec.whatwg.org/#node)? [previousSibling](https://dom.spec.whatwg.org/#dom-mutationrecord-previoussibling);

readonly attribute [Node](https://dom.spec.whatwg.org/#node)? [nextSibling](https://dom.spec.whatwg.org/#dom-mutationrecord-nextsibling);

readonly attribute DOMString? [attributeName](https://dom.spec.whatwg.org/#dom-mutationrecord-attributename);

readonly attribute DOMString? [attributeNamespace](https://dom.spec.whatwg.org/#dom-mutationrecord-attributenamespace);

readonly attribute DOMString? [oldValue](https://dom.spec.whatwg.org/#dom-mutationrecord-oldvalue);

};

*record* . [type](https://dom.spec.whatwg.org/#dom-mutationrecord-type)

Returns "attributes" if it was an [attribute](https://dom.spec.whatwg.org/#concept-attribute) mutation. "characterData" if it was a mutation to a [CharacterData](https://dom.spec.whatwg.org/#characterdata) [node](https://dom.spec.whatwg.org/#concept-node). And "childList" if it was a mutation to the [tree](https://dom.spec.whatwg.org/#concept-tree) of [nodes](https://dom.spec.whatwg.org/#concept-node).

*record* . [target](https://dom.spec.whatwg.org/#dom-mutationrecord-target)

Returns the [node](https://dom.spec.whatwg.org/#concept-node) the mutation affected, depending on the [type](https://dom.spec.whatwg.org/#dom-mutationrecord-type). For "attributes", it is the [element](https://dom.spec.whatwg.org/#concept-element) whose [attribute](https://dom.spec.whatwg.org/#concept-attribute) changed. For "characterData", it is the [CharacterData](https://dom.spec.whatwg.org/#characterdata) [node](https://dom.spec.whatwg.org/#concept-node). For "childList", it is the [node](https://dom.spec.whatwg.org/#concept-node) whose [children](https://dom.spec.whatwg.org/#concept-tree-child) changed.

*record* . [addedNodes](https://dom.spec.whatwg.org/#dom-mutationrecord-addednodes)

*record* . [removedNodes](https://dom.spec.whatwg.org/#dom-mutationrecord-removednodes)

Return the [nodes](https://dom.spec.whatwg.org/#concept-node) added and removed respectively.

*record* . [previousSibling](https://dom.spec.whatwg.org/#dom-mutationrecord-previoussibling)

*record* . [nextSibling](https://dom.spec.whatwg.org/#dom-mutationrecord-nextsibling)

Return the [previous](https://dom.spec.whatwg.org/#concept-tree-previous-sibling) and [next sibling](https://dom.spec.whatwg.org/#concept-tree-next-sibling) respectively of the added or removed [nodes](https://dom.spec.whatwg.org/#concept-node), and null otherwise.

*record* . [attributeName](https://dom.spec.whatwg.org/#dom-mutationrecord-attributename)

Returns the [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name) of the changed [attribute](https://dom.spec.whatwg.org/#concept-attribute), and null otherwise.

*record* . [attributeNamespace](https://dom.spec.whatwg.org/#dom-mutationrecord-attributenamespace)

Returns the [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace) of the changed [attribute](https://dom.spec.whatwg.org/#concept-attribute), and null otherwise.

*record* . [oldValue](https://dom.spec.whatwg.org/#dom-mutationrecord-oldvalue)

The return value depends on [type](https://dom.spec.whatwg.org/#dom-mutationrecord-type). For "attributes", it is the [value](https://dom.spec.whatwg.org/#concept-attribute-value) of the changed [attribute](https://dom.spec.whatwg.org/#concept-attribute) before the change. For "characterData", it is the [data](https://dom.spec.whatwg.org/#concept-cd-data) of the changed [node](https://dom.spec.whatwg.org/#concept-node) before the change. For "childList", it is null.

The *type* and *target* attributes must return the values they were initialized to.

The *addedNodes* and *removedNodes* attributes must return the values they were initialized to. Unless stated otherwise, when a [MutationRecord](https://dom.spec.whatwg.org/#mutationrecord) object is created, they must both be initialized to an empty [NodeList](https://dom.spec.whatwg.org/#nodelist).

The *previousSibling*, *nextSibling*, *attributeName*, *attributeNamespace*, and *oldValue* attributes must return the values they were initialized to. Unless stated otherwise, when a [MutationRecord](https://dom.spec.whatwg.org/#mutationrecord) object is created, they must be initialized to null.

**4.3.4. Garbage collection**

[Nodes](https://dom.spec.whatwg.org/#concept-node) have a strong reference to [registered observers](https://dom.spec.whatwg.org/#registered-observer) in their list of [registered observers](https://dom.spec.whatwg.org/#registered-observer).

[Registered observers](https://dom.spec.whatwg.org/#registered-observer) in a [node](https://dom.spec.whatwg.org/#concept-node)’s list of [registered observers](https://dom.spec.whatwg.org/#registered-observer) have a weak reference to the [node](https://dom.spec.whatwg.org/#concept-node).

**4.4. Interface** [**Node**](https://dom.spec.whatwg.org/#node)

[Exposed=Window]

interface *Node* : [EventTarget](https://dom.spec.whatwg.org/#eventtarget) {

const unsigned short [ELEMENT\_NODE](https://dom.spec.whatwg.org/#dom-node-element_node) = 1;

const unsigned short *ATTRIBUTE\_NODE* = 2; // historical

const unsigned short [TEXT\_NODE](https://dom.spec.whatwg.org/#dom-node-text_node) = 3;

const unsigned short *CDATA\_SECTION\_NODE* = 4; // historical

const unsigned short *ENTITY\_REFERENCE\_NODE* = 5; // historical

const unsigned short *ENTITY\_NODE* = 6; // historical

const unsigned short [PROCESSING\_INSTRUCTION\_NODE](https://dom.spec.whatwg.org/#dom-node-processing_instruction_node) = 7;

const unsigned short [COMMENT\_NODE](https://dom.spec.whatwg.org/#dom-node-comment_node) = 8;

const unsigned short [DOCUMENT\_NODE](https://dom.spec.whatwg.org/#dom-node-document_node) = 9;

const unsigned short [DOCUMENT\_TYPE\_NODE](https://dom.spec.whatwg.org/#dom-node-document_type_node) = 10;

const unsigned short [DOCUMENT\_FRAGMENT\_NODE](https://dom.spec.whatwg.org/#dom-node-document_fragment_node) = 11;

const unsigned short *NOTATION\_NODE* = 12; // historical

readonly attribute unsigned short [nodeType](https://dom.spec.whatwg.org/#dom-node-nodetype);

readonly attribute DOMString [nodeName](https://dom.spec.whatwg.org/#dom-node-nodename);

readonly attribute DOMString [baseURI](https://dom.spec.whatwg.org/#dom-node-baseuri);

readonly attribute boolean [isConnected](https://dom.spec.whatwg.org/#dom-node-isconnected);

readonly attribute [Document](https://dom.spec.whatwg.org/#document)? [ownerDocument](https://dom.spec.whatwg.org/#dom-node-ownerdocument);

readonly attribute [Node](https://dom.spec.whatwg.org/#node) [rootNode](https://dom.spec.whatwg.org/#dom-node-rootnode);

readonly attribute [Node](https://dom.spec.whatwg.org/#node)? [parentNode](https://dom.spec.whatwg.org/#dom-node-parentnode);

readonly attribute [Element](https://dom.spec.whatwg.org/#element)? [parentElement](https://dom.spec.whatwg.org/#dom-node-parentelement);

boolean [hasChildNodes](https://dom.spec.whatwg.org/#dom-node-haschildnodes)();

[SameObject] readonly attribute [NodeList](https://dom.spec.whatwg.org/#nodelist) [childNodes](https://dom.spec.whatwg.org/#dom-node-childnodes);

readonly attribute [Node](https://dom.spec.whatwg.org/#node)? [firstChild](https://dom.spec.whatwg.org/#dom-node-firstchild);

readonly attribute [Node](https://dom.spec.whatwg.org/#node)? [lastChild](https://dom.spec.whatwg.org/#dom-node-lastchild);

readonly attribute [Node](https://dom.spec.whatwg.org/#node)? [previousSibling](https://dom.spec.whatwg.org/#dom-node-previoussibling);

readonly attribute [Node](https://dom.spec.whatwg.org/#node)? [nextSibling](https://dom.spec.whatwg.org/#dom-node-nextsibling);

[CEReactions] attribute DOMString? [nodeValue](https://dom.spec.whatwg.org/#dom-node-nodevalue);

[CEReactions] attribute DOMString? [textContent](https://dom.spec.whatwg.org/#dom-node-textcontent);

[CEReactions] void [normalize](https://dom.spec.whatwg.org/#dom-node-normalize)();

[CEReactions, NewObject] [Node](https://dom.spec.whatwg.org/#node) [cloneNode](https://dom.spec.whatwg.org/#dom-node-clonenode)(optional boolean *deep* = false);

boolean [isEqualNode](https://dom.spec.whatwg.org/#dom-node-isequalnode)([Node](https://dom.spec.whatwg.org/#node)? *otherNode*);

boolean [isSameNode](https://dom.spec.whatwg.org/#dom-node-issamenode)([Node](https://dom.spec.whatwg.org/#node)? *otherNode*); // historical alias of ===

const unsigned short [DOCUMENT\_POSITION\_DISCONNECTED](https://dom.spec.whatwg.org/#dom-node-document_position_disconnected) = 0x01;

const unsigned short [DOCUMENT\_POSITION\_PRECEDING](https://dom.spec.whatwg.org/#dom-node-document_position_preceding) = 0x02;

const unsigned short [DOCUMENT\_POSITION\_FOLLOWING](https://dom.spec.whatwg.org/#dom-node-document_position_following) = 0x04;

const unsigned short [DOCUMENT\_POSITION\_CONTAINS](https://dom.spec.whatwg.org/#dom-node-document_position_contains) = 0x08;

const unsigned short [DOCUMENT\_POSITION\_CONTAINED\_BY](https://dom.spec.whatwg.org/#dom-node-document_position_contained_by) = 0x10;

const unsigned short [DOCUMENT\_POSITION\_IMPLEMENTATION\_SPECIFIC](https://dom.spec.whatwg.org/#dom-node-document_position_implementation_specific) = 0x20;

unsigned short [compareDocumentPosition](https://dom.spec.whatwg.org/#dom-node-comparedocumentposition)([Node](https://dom.spec.whatwg.org/#node) *other*);

boolean [contains](https://dom.spec.whatwg.org/#dom-node-contains)([Node](https://dom.spec.whatwg.org/#node)? *other*);

DOMString? [lookupPrefix](https://dom.spec.whatwg.org/#dom-node-lookupprefix)(DOMString? *namespace*);

DOMString? [lookupNamespaceURI](https://dom.spec.whatwg.org/#dom-node-lookupnamespaceuri)(DOMString? *prefix*);

boolean [isDefaultNamespace](https://dom.spec.whatwg.org/#dom-node-isdefaultnamespace)(DOMString? *namespace*);

[CEReactions] [Node](https://dom.spec.whatwg.org/#node) [insertBefore](https://dom.spec.whatwg.org/#dom-node-insertbefore)([Node](https://dom.spec.whatwg.org/#node) *node*, [Node](https://dom.spec.whatwg.org/#node)? *child*);

[CEReactions] [Node](https://dom.spec.whatwg.org/#node) [appendChild](https://dom.spec.whatwg.org/#dom-node-appendchild)([Node](https://dom.spec.whatwg.org/#node) *node*);

[CEReactions] [Node](https://dom.spec.whatwg.org/#node) [replaceChild](https://dom.spec.whatwg.org/#dom-node-replacechild)([Node](https://dom.spec.whatwg.org/#node) *node*, [Node](https://dom.spec.whatwg.org/#node) *child*);

[CEReactions] [Node](https://dom.spec.whatwg.org/#node) [removeChild](https://dom.spec.whatwg.org/#dom-node-removechild)([Node](https://dom.spec.whatwg.org/#node) *child*);

};

[Node](https://dom.spec.whatwg.org/#node) is an abstract interface and does not exist as [node](https://dom.spec.whatwg.org/#concept-node). It is used by all [nodes](https://dom.spec.whatwg.org/#concept-node) ([Document](https://dom.spec.whatwg.org/#document), [DocumentType](https://dom.spec.whatwg.org/#documenttype), [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment), [ShadowRoot](https://dom.spec.whatwg.org/#shadowroot), [Element](https://dom.spec.whatwg.org/#element), [Text](https://dom.spec.whatwg.org/#text), [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction), and [Comment](https://dom.spec.whatwg.org/#comment)).

Each [node](https://dom.spec.whatwg.org/#concept-node) has an associated *node document*, set upon creation, that is a [document](https://dom.spec.whatwg.org/#concept-document).

A [node](https://dom.spec.whatwg.org/#concept-node)’s [node document](https://dom.spec.whatwg.org/#concept-node-document) can be changed by the [adopt](https://dom.spec.whatwg.org/#concept-node-adopt) algorithm.

A [node](https://dom.spec.whatwg.org/#concept-node)’s [get the parent](https://dom.spec.whatwg.org/#get-the-parent) algorithm, given an *event*, returns the [node](https://dom.spec.whatwg.org/#concept-node)’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent).

*node* . [nodeType](https://dom.spec.whatwg.org/#dom-node-nodetype)

Returns the type of *node*, represented by a number from the following list:

[Node](https://dom.spec.whatwg.org/#node) . [ELEMENT\_NODE](https://dom.spec.whatwg.org/#dom-node-element_node) (1)

*node* is an [element](https://dom.spec.whatwg.org/#concept-element).

[Node](https://dom.spec.whatwg.org/#node) . [TEXT\_NODE](https://dom.spec.whatwg.org/#dom-node-text_node) (3)

*node* is a [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node).

[Node](https://dom.spec.whatwg.org/#node) . [PROCESSING\_INSTRUCTION\_NODE](https://dom.spec.whatwg.org/#dom-node-processing_instruction_node) (7)

*node* is a [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction) [node](https://dom.spec.whatwg.org/#concept-node).

[Node](https://dom.spec.whatwg.org/#node) . [COMMENT\_NODE](https://dom.spec.whatwg.org/#dom-node-comment_node) (8)

*node* is a [Comment](https://dom.spec.whatwg.org/#comment) [node](https://dom.spec.whatwg.org/#concept-node).

[Node](https://dom.spec.whatwg.org/#node) . [DOCUMENT\_NODE](https://dom.spec.whatwg.org/#dom-node-document_node) (9)

*node* is a [document](https://dom.spec.whatwg.org/#concept-document).

[Node](https://dom.spec.whatwg.org/#node) . [DOCUMENT\_TYPE\_NODE](https://dom.spec.whatwg.org/#dom-node-document_type_node) (10)

*node* is a [doctype](https://dom.spec.whatwg.org/#concept-doctype).

[Node](https://dom.spec.whatwg.org/#node) . [DOCUMENT\_FRAGMENT\_NODE](https://dom.spec.whatwg.org/#dom-node-document_fragment_node) (11)

*node* is a [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) or [ShadowRoot](https://dom.spec.whatwg.org/#shadowroot) [node](https://dom.spec.whatwg.org/#concept-node).

*node* . [nodeName](https://dom.spec.whatwg.org/#dom-node-nodename)

Returns a string appropriate for the type of *node*, as follows:

[Element](https://dom.spec.whatwg.org/#element)

Its [tagName](https://dom.spec.whatwg.org/#dom-element-tagname) attribute value.

[Text](https://dom.spec.whatwg.org/#text)

"#text".

[ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction)

Its [target](https://dom.spec.whatwg.org/#concept-pi-target).

[Comment](https://dom.spec.whatwg.org/#comment)

"#comment".

[Document](https://dom.spec.whatwg.org/#document)

"#document".

[DocumentType](https://dom.spec.whatwg.org/#documenttype)

Its [name](https://dom.spec.whatwg.org/#concept-doctype-name).

[DocumentFragment](https://dom.spec.whatwg.org/#documentfragment)

[ShadowRoot](https://dom.spec.whatwg.org/#shadowroot)

"#document-fragment".

The *nodeType* attribute’s getter, when invoked, must return the first matching statement, switching on the [context object](https://dom.spec.whatwg.org/#context-object):

[Element](https://dom.spec.whatwg.org/#element)

*ELEMENT\_NODE* (1)

[Text](https://dom.spec.whatwg.org/#text)

*TEXT\_NODE* (3);

[ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction)

*PROCESSING\_INSTRUCTION\_NODE* (7);

[Comment](https://dom.spec.whatwg.org/#comment)

*COMMENT\_NODE* (8);

[Document](https://dom.spec.whatwg.org/#document)

*DOCUMENT\_NODE* (9);

[DocumentType](https://dom.spec.whatwg.org/#documenttype)

*DOCUMENT\_TYPE\_NODE* (10);

[DocumentFragment](https://dom.spec.whatwg.org/#documentfragment)

[ShadowRoot](https://dom.spec.whatwg.org/#shadowroot)

*DOCUMENT\_FRAGMENT\_NODE* (11).

The *nodeName* attribute’s getter, when invoked, must return the first matching statement, switching on the [context object](https://dom.spec.whatwg.org/#context-object):

[Element](https://dom.spec.whatwg.org/#element)

Its [tagName](https://dom.spec.whatwg.org/#dom-element-tagname) attribute value.

[Text](https://dom.spec.whatwg.org/#text)

"#text".

[ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction)

Its [target](https://dom.spec.whatwg.org/#concept-pi-target).

[Comment](https://dom.spec.whatwg.org/#comment)

"#comment".

[Document](https://dom.spec.whatwg.org/#document)

"#document".

[DocumentType](https://dom.spec.whatwg.org/#documenttype)

Its [name](https://dom.spec.whatwg.org/#concept-doctype-name).

[DocumentFragment](https://dom.spec.whatwg.org/#documentfragment)

[ShadowRoot](https://dom.spec.whatwg.org/#shadowroot)

"#document-fragment".

*node* . [baseURI](https://dom.spec.whatwg.org/#dom-node-baseuri)

Returns *node*’s [node document](https://dom.spec.whatwg.org/#concept-node-document)’s [base URL](https://dom.spec.whatwg.org/#concept-document-base-url).

The *baseURI* attribute’s getter must return [node document](https://dom.spec.whatwg.org/#concept-node-document)’s [base URL](https://dom.spec.whatwg.org/#concept-document-base-url), [serialized](https://url.spec.whatwg.org/#concept-url-serializer).

*node* . [isConnected](https://dom.spec.whatwg.org/#dom-node-isconnected)

Returns true if *node* is [in a shadow-including document](https://dom.spec.whatwg.org/#in-a-shadow-including-document) and false otherwise.

*node* . [ownerDocument](https://dom.spec.whatwg.org/#dom-node-ownerdocument)

Returns the [node document](https://dom.spec.whatwg.org/#concept-node-document). Returns null for [documents](https://dom.spec.whatwg.org/#concept-document).

*node* . [rootNode](https://dom.spec.whatwg.org/#dom-node-rootnode)

Returns the [root](https://dom.spec.whatwg.org/#concept-tree-root).

*node* . [parentNode](https://dom.spec.whatwg.org/#dom-node-parentnode)

Returns the [parent](https://dom.spec.whatwg.org/#concept-tree-parent).

*node* . [parentElement](https://dom.spec.whatwg.org/#dom-node-parentelement)

Returns the [parent element](https://dom.spec.whatwg.org/#parent-element).

*node* . [hasChildNodes()](https://dom.spec.whatwg.org/#dom-node-haschildnodes)

Returns whether *node* has [children](https://dom.spec.whatwg.org/#concept-tree-child).

*node* . [childNodes](https://dom.spec.whatwg.org/#dom-node-childnodes)

Returns the [children](https://dom.spec.whatwg.org/#concept-tree-child).

*node* . [firstChild](https://dom.spec.whatwg.org/#dom-node-firstchild)

Returns the [first child](https://dom.spec.whatwg.org/#concept-tree-first-child).

*node* . [lastChild](https://dom.spec.whatwg.org/#dom-node-lastchild)

Returns the [last child](https://dom.spec.whatwg.org/#concept-tree-last-child).

*node* . [previousSibling](https://dom.spec.whatwg.org/#dom-node-previoussibling)

Returns the [previous sibling](https://dom.spec.whatwg.org/#concept-tree-previous-sibling).

*node* . [nextSibling](https://dom.spec.whatwg.org/#dom-node-nextsibling)

Returns the [next sibling](https://dom.spec.whatwg.org/#concept-tree-next-sibling).

The *isConnected* attribute’s getter must return true, if [context object](https://dom.spec.whatwg.org/#context-object) is [in a shadow-including document](https://dom.spec.whatwg.org/#in-a-shadow-including-document), and false otherwise.

The *ownerDocument* attribute’s getter must return null, if the [context object](https://dom.spec.whatwg.org/#context-object) is a [document](https://dom.spec.whatwg.org/#concept-document), and the [context object](https://dom.spec.whatwg.org/#context-object)’s [node document](https://dom.spec.whatwg.org/#concept-node-document) otherwise.

The [node document](https://dom.spec.whatwg.org/#concept-node-document) of a [document](https://dom.spec.whatwg.org/#concept-document) is that [document](https://dom.spec.whatwg.org/#concept-document) itself. All [nodes](https://dom.spec.whatwg.org/#concept-node) have a [node document](https://dom.spec.whatwg.org/#concept-node-document) at all times.

The *rootNode* attribute’s getter must return [context object](https://dom.spec.whatwg.org/#context-object)’s [root](https://dom.spec.whatwg.org/#concept-tree-root).

The *parentNode* attribute’s getter must return the [context object](https://dom.spec.whatwg.org/#context-object)’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent).

The *parentElement* attribute’s getter must return the [context object](https://dom.spec.whatwg.org/#context-object)’s [parent element](https://dom.spec.whatwg.org/#parent-element).

The *hasChildNodes()* method, when invoked, must return true if the [context object](https://dom.spec.whatwg.org/#context-object) has [children](https://dom.spec.whatwg.org/#concept-tree-child), and false otherwise.

The *childNodes* attribute’s getter must return a [NodeList](https://dom.spec.whatwg.org/#nodelist) rooted at the [context object](https://dom.spec.whatwg.org/#context-object) matching only [children](https://dom.spec.whatwg.org/#concept-tree-child).

The *firstChild* attribute’s getter must return the [context object](https://dom.spec.whatwg.org/#context-object)’s [first child](https://dom.spec.whatwg.org/#concept-tree-first-child).

The *lastChild* attribute’s getter must return the [context object](https://dom.spec.whatwg.org/#context-object)’s [last child](https://dom.spec.whatwg.org/#concept-tree-last-child).

The *previousSibling* attribute’s getter must return the [context object](https://dom.spec.whatwg.org/#context-object)’s [previous sibling](https://dom.spec.whatwg.org/#concept-tree-previous-sibling).

The *nextSibling* attribute’s getter must return the [context object](https://dom.spec.whatwg.org/#context-object)’s [next sibling](https://dom.spec.whatwg.org/#concept-tree-next-sibling).

The *nodeValue* attribute must return the following, depending on the [context object](https://dom.spec.whatwg.org/#context-object):

[Text](https://dom.spec.whatwg.org/#text)

[ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction)

[Comment](https://dom.spec.whatwg.org/#comment)

The [context object](https://dom.spec.whatwg.org/#context-object)’s [data](https://dom.spec.whatwg.org/#concept-cd-data).

Any other node

Null.

The [nodeValue](https://dom.spec.whatwg.org/#dom-node-nodevalue) attribute must, on setting, if the new value is null, act as if it was the empty string instead, and then do as described below, depending on the [context object](https://dom.spec.whatwg.org/#context-object):

[Text](https://dom.spec.whatwg.org/#text)

[ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction)

[Comment](https://dom.spec.whatwg.org/#comment)

[Replace data](https://dom.spec.whatwg.org/#concept-cd-replace) with node [context object](https://dom.spec.whatwg.org/#context-object), offset 0, count [length](https://dom.spec.whatwg.org/#dom-characterdata-length) attribute value, and data new value.

Any other node

Do nothing.

The *textContent* attribute’s getter must return the following, switching on [context object](https://dom.spec.whatwg.org/#context-object):

[DocumentFragment](https://dom.spec.whatwg.org/#documentfragment)

[ShadowRoot](https://dom.spec.whatwg.org/#shadowroot)

[Element](https://dom.spec.whatwg.org/#element)

The concatenation of [data](https://dom.spec.whatwg.org/#concept-cd-data) of all the [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node) [descendants](https://dom.spec.whatwg.org/#concept-tree-descendant) of the [context object](https://dom.spec.whatwg.org/#context-object), in [tree order](https://dom.spec.whatwg.org/#concept-tree-order).

[Text](https://dom.spec.whatwg.org/#text)

[ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction)

[Comment](https://dom.spec.whatwg.org/#comment)

The [context object](https://dom.spec.whatwg.org/#context-object)’s [data](https://dom.spec.whatwg.org/#concept-cd-data).

Any other node

Null.

The [textContent](https://dom.spec.whatwg.org/#dom-node-textcontent) attribute’s setter must, if the given value is null, act as if it was the empty string instead, and then do as described below, switching on [context object](https://dom.spec.whatwg.org/#context-object):

[DocumentFragment](https://dom.spec.whatwg.org/#documentfragment)

[ShadowRoot](https://dom.spec.whatwg.org/#shadowroot)

[Element](https://dom.spec.whatwg.org/#element)

1. Let *node* be null.
2. If the given value is not the empty string, set *node* to a new [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node) whose [data](https://dom.spec.whatwg.org/#concept-cd-data) is the given value and [node document](https://dom.spec.whatwg.org/#concept-node-document) is [context object](https://dom.spec.whatwg.org/#context-object)’s [node document](https://dom.spec.whatwg.org/#concept-node-document).
3. [Replace all](https://dom.spec.whatwg.org/#concept-node-replace-all) with *node* within the [context object](https://dom.spec.whatwg.org/#context-object).

[Text](https://dom.spec.whatwg.org/#text)

[ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction)

[Comment](https://dom.spec.whatwg.org/#comment)

[Replace data](https://dom.spec.whatwg.org/#concept-cd-replace) with node [context object](https://dom.spec.whatwg.org/#context-object), offset 0, count [length](https://dom.spec.whatwg.org/#dom-characterdata-length) attribute value, and data the given value.

Any other node

Do nothing.

*node* . [normalize()](https://dom.spec.whatwg.org/#dom-node-normalize)

Removes [empty](https://dom.spec.whatwg.org/#concept-node-empty) [Text](https://dom.spec.whatwg.org/#text) [nodes](https://dom.spec.whatwg.org/#concept-node) and concatenates the [data](https://dom.spec.whatwg.org/#concept-cd-data) of remaining [contiguous Text nodes](https://dom.spec.whatwg.org/#contiguous-text-nodes) into the first of their [nodes](https://dom.spec.whatwg.org/#concept-node).

The *normalize()* method must run these steps:

For each [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node) [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) of the [context object](https://dom.spec.whatwg.org/#context-object):

1. Let *node* be the [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node) [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant).
2. Let *length* be *node*’s [length](https://dom.spec.whatwg.org/#dom-characterdata-length) attribute value.
3. If *length* is zero, [remove](https://dom.spec.whatwg.org/#concept-node-remove) *node* and continue with the next [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node), if any.
4. Let *data* be the concatenation of the [data](https://dom.spec.whatwg.org/#concept-cd-data) of *node*’s [contiguous Text nodes](https://dom.spec.whatwg.org/#contiguous-text-nodes) (excluding itself), in [tree order](https://dom.spec.whatwg.org/#concept-tree-order).
5. [Replace data](https://dom.spec.whatwg.org/#concept-cd-replace) with node *node*, offset *length*, count 0, and data *data*.
6. Let *current node* be *node*’s [next sibling](https://dom.spec.whatwg.org/#concept-tree-next-sibling).
7. While *current node* is a [Text](https://dom.spec.whatwg.org/#text) node:
   1. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [start node](https://dom.spec.whatwg.org/#concept-range-start-node) is *current node*, add *length* to its [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) and set its [start node](https://dom.spec.whatwg.org/#concept-range-start-node) to *node*.
   2. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [end node](https://dom.spec.whatwg.org/#concept-range-end-node) is *current node*, add *length* to its [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset) and set its [end node](https://dom.spec.whatwg.org/#concept-range-end-node) to *node*.
   3. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [start node](https://dom.spec.whatwg.org/#concept-range-start-node) is *current node*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) and [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) is *current node*’s [index](https://dom.spec.whatwg.org/#concept-tree-index), set its [start node](https://dom.spec.whatwg.org/#concept-range-start-node) to *node* and its [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) to *length*.
   4. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [end node](https://dom.spec.whatwg.org/#concept-range-end-node) is *current node*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) and [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset) is *current node*’s [index](https://dom.spec.whatwg.org/#concept-tree-index), set its [end node](https://dom.spec.whatwg.org/#concept-range-end-node) to *node* and its [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset) to *length*.
   5. Add *current node*’s [length](https://dom.spec.whatwg.org/#dom-characterdata-length) attribute value to *length*.
   6. Set *current node* to its [next sibling](https://dom.spec.whatwg.org/#concept-tree-next-sibling).
8. [Remove](https://dom.spec.whatwg.org/#concept-node-remove) *node*’s [contiguous Text nodes](https://dom.spec.whatwg.org/#contiguous-text-nodes) (excluding itself), in [tree order](https://dom.spec.whatwg.org/#concept-tree-order).

*node* . [cloneNode([*deep* = false])](https://dom.spec.whatwg.org/#dom-node-clonenode)

Returns a copy of *node*. If *deep* is true, the copy also includes the *node*’s [descendants](https://dom.spec.whatwg.org/#concept-tree-descendant).

*node* . [isEqualNode(otherNode)](https://dom.spec.whatwg.org/#dom-node-isequalnode)

Returns whether *node* and *otherNode* have the same properties.

[Specifications](https://dom.spec.whatwg.org/#other-applicable-specifications) may define *cloning steps* for all or some [nodes](https://dom.spec.whatwg.org/#concept-node). The algorithm is passed *copy*, *node*, *document*, and an optional *clone children flag*, as indicated in the [clone](https://dom.spec.whatwg.org/#concept-node-clone) algorithm.

HTML defines [cloning steps](https://dom.spec.whatwg.org/#concept-node-clone-ext) for [script](https://html.spec.whatwg.org/multipage/scripting.html#script) and [input](https://html.spec.whatwg.org/multipage/forms.html#the-input-element) elements. SVG ought to do the same for its [script](https://html.spec.whatwg.org/multipage/scripting.html#script) elements, but does not call this out at the moment.

To *clone* a *node*, with an optional *document* and *clone children flag*, run these steps:

1. If *document* is not given, let *document* be *node*’s [node document](https://dom.spec.whatwg.org/#concept-node-document).
2. If *node* is an [element](https://dom.spec.whatwg.org/#concept-element), then:
   1. Let *copy* be the result of [creating an element](https://dom.spec.whatwg.org/#concept-create-element), given *document*, *node*’s [local name](https://dom.spec.whatwg.org/#concept-element-local-name), *node*’s [namespace](https://dom.spec.whatwg.org/#concept-element-namespace), *node*’s [namespace prefix](https://dom.spec.whatwg.org/#concept-element-namespace-prefix), and the value of *node*’s is attribute if present (or null if not). The *synchronous custom elements flag* should be unset.
   2. For each *attribute* in *node*’s [attribute list](https://dom.spec.whatwg.org/#concept-element-attribute), in order, run these substeps:
      1. Let *copyAttribute* be a new [attribute](https://dom.spec.whatwg.org/#concept-attribute).
      2. Set *copyAttribute*’s [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace), [namespace prefix](https://dom.spec.whatwg.org/#concept-attribute-namespace-prefix), [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name), and [value](https://dom.spec.whatwg.org/#concept-attribute-value), to those of *attribute*.
      3. [Append](https://dom.spec.whatwg.org/#concept-element-attributes-append) *copyAttribute* to *copy*.
3. Otherwise, let *copy* be a [node](https://dom.spec.whatwg.org/#concept-node) that implements the same interfaces as *node*, and fulfills these additional requirements, switching on *node*:

[Document](https://dom.spec.whatwg.org/#document)

Set *copy*’s [encoding](https://dom.spec.whatwg.org/#concept-document-encoding), [content type](https://dom.spec.whatwg.org/#concept-document-content-type), [URL](https://dom.spec.whatwg.org/#concept-document-url), [type](https://dom.spec.whatwg.org/#concept-document-type), and [mode](https://dom.spec.whatwg.org/#concept-document-mode), to those of *node*.

[DocumentType](https://dom.spec.whatwg.org/#documenttype)

Set *copy*’s [name](https://dom.spec.whatwg.org/#concept-doctype-name), [public ID](https://dom.spec.whatwg.org/#concept-doctype-publicid), and [system ID](https://dom.spec.whatwg.org/#concept-doctype-systemid), to those of *node*.

[Text](https://dom.spec.whatwg.org/#text)

[Comment](https://dom.spec.whatwg.org/#comment)

Set *copy*’s [data](https://dom.spec.whatwg.org/#concept-cd-data), to that of *node*.

[ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction)

Set *copy*’s [target](https://dom.spec.whatwg.org/#concept-pi-target) and [data](https://dom.spec.whatwg.org/#concept-cd-data) to those of *node*.

Any other node

—

1. Set *copy*’s [node document](https://dom.spec.whatwg.org/#concept-node-document) and *document* to *copy*, if *copy* is a [document](https://dom.spec.whatwg.org/#concept-document), and set *copy*’s [node document](https://dom.spec.whatwg.org/#concept-node-document) to *document* otherwise.
2. Run any [cloning steps](https://dom.spec.whatwg.org/#concept-node-clone-ext) defined for *node* in [other applicable specifications](https://dom.spec.whatwg.org/#other-applicable-specifications) and pass *copy*, *node*, *document* and the *clone children flag* if set, as parameters.
3. If the *clone children flag* is set, [clone](https://dom.spec.whatwg.org/#concept-node-clone) all the [children](https://dom.spec.whatwg.org/#concept-tree-child) of *node* and append them to *copy*, with *document* as specified and the *clone children flag* being set.
4. Return *copy*.

The *cloneNode(deep)* method, when invoked, must run these steps:

1. If [context object](https://dom.spec.whatwg.org/#context-object) is a [shadow root](https://dom.spec.whatwg.org/#concept-shadow-root), then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotSupportedError](https://heycam.github.io/webidl/#notsupportederror).
2. Return a [clone](https://dom.spec.whatwg.org/#concept-node-clone) of the [context object](https://dom.spec.whatwg.org/#context-object), with the *clone children flag* set if *deep* is true.

A [node](https://dom.spec.whatwg.org/#concept-node) *A* *equals* a [node](https://dom.spec.whatwg.org/#concept-node) *B* if all of the following conditions are true:

* *A* and *B*’s [nodeType](https://dom.spec.whatwg.org/#dom-node-nodetype) attribute value is identical.
* The following are also equal, depending on *A*:

[DocumentType](https://dom.spec.whatwg.org/#documenttype)

Its [name](https://dom.spec.whatwg.org/#concept-doctype-name), [public ID](https://dom.spec.whatwg.org/#concept-doctype-publicid), and [system ID](https://dom.spec.whatwg.org/#concept-doctype-systemid).

[Element](https://dom.spec.whatwg.org/#element)

Its [namespace](https://dom.spec.whatwg.org/#concept-element-namespace), [namespace prefix](https://dom.spec.whatwg.org/#concept-element-namespace-prefix), [local name](https://dom.spec.whatwg.org/#concept-element-local-name), and its number of [attributes](https://dom.spec.whatwg.org/#concept-attribute) in its [attribute list](https://dom.spec.whatwg.org/#concept-element-attribute).

[ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction)

Its [target](https://dom.spec.whatwg.org/#concept-pi-target) and [data](https://dom.spec.whatwg.org/#concept-cd-data).

[Text](https://dom.spec.whatwg.org/#text)

[Comment](https://dom.spec.whatwg.org/#comment)

Its [data](https://dom.spec.whatwg.org/#concept-cd-data).

Any other node

—

* If *A* is an [element](https://dom.spec.whatwg.org/#concept-element), each [attribute](https://dom.spec.whatwg.org/#concept-attribute) in its [attribute list](https://dom.spec.whatwg.org/#concept-element-attribute) has an [attribute](https://dom.spec.whatwg.org/#concept-attribute) with the same [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace), [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name), and [value](https://dom.spec.whatwg.org/#concept-attribute-value) in *B*’s [attribute list](https://dom.spec.whatwg.org/#concept-element-attribute).
* *A* and *B* have the same number of [children](https://dom.spec.whatwg.org/#concept-tree-child).
* Each [child](https://dom.spec.whatwg.org/#concept-tree-child) of *A* [equals](https://dom.spec.whatwg.org/#concept-node-equals) the [child](https://dom.spec.whatwg.org/#concept-tree-child) of *B* at the identical [index](https://dom.spec.whatwg.org/#concept-tree-index).

The *isEqualNode(otherNode)* method, when invoked, must return true if *otherNode* is non-null and [context object](https://dom.spec.whatwg.org/#context-object) [equals](https://dom.spec.whatwg.org/#concept-node-equals) *otherNode*, and false otherwise.

The *isSameNode(otherNode)* method, when invoked, must return true if *otherNode* is [context object](https://dom.spec.whatwg.org/#context-object), and false otherwise.

*node* . [compareDocumentPosition(other)](https://dom.spec.whatwg.org/#dom-node-comparedocumentposition)

Returns a bitmask indicating the position of *other* relative to *node*. These are the bits that can be set:

[Node](https://dom.spec.whatwg.org/#node) . [DOCUMENT\_POSITION\_DISCONNECTED](https://dom.spec.whatwg.org/#dom-node-document_position_disconnected) (1)

Set when *node* and *other* are not in the same [tree](https://dom.spec.whatwg.org/#concept-tree).

[Node](https://dom.spec.whatwg.org/#node) . [DOCUMENT\_POSITION\_PRECEDING](https://dom.spec.whatwg.org/#dom-node-document_position_preceding) (2)

Set when *other* is [preceding](https://dom.spec.whatwg.org/#concept-tree-preceding) *node*.

[Node](https://dom.spec.whatwg.org/#node) . [DOCUMENT\_POSITION\_FOLLOWING](https://dom.spec.whatwg.org/#dom-node-document_position_following) (4)

Set when *other* is [following](https://dom.spec.whatwg.org/#concept-tree-following) *node*.

[Node](https://dom.spec.whatwg.org/#node) . [DOCUMENT\_POSITION\_CONTAINS](https://dom.spec.whatwg.org/#dom-node-document_position_contains) (8)

Set when *other* is an [ancestor](https://dom.spec.whatwg.org/#concept-tree-ancestor) of *node*.

[Node](https://dom.spec.whatwg.org/#node) . [DOCUMENT\_POSITION\_CONTAINED\_BY](https://dom.spec.whatwg.org/#dom-node-document_position_contained_by) (16, 10 in hexadecimal)

Set when *other* is a [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) of *node*.

*node* . [contains(other)](https://dom.spec.whatwg.org/#dom-node-contains)

Returns true if *other* is an [inclusive descendant](https://dom.spec.whatwg.org/#concept-tree-inclusive-descendant) of *node*, and false otherwise.

These are the constants [compareDocumentPosition()](https://dom.spec.whatwg.org/#dom-node-comparedocumentposition) returns as mask:

* *DOCUMENT\_POSITION\_DISCONNECTED* (1);
* *DOCUMENT\_POSITION\_PRECEDING* (2);
* *DOCUMENT\_POSITION\_FOLLOWING* (4);
* *DOCUMENT\_POSITION\_CONTAINS* (8);
* *DOCUMENT\_POSITION\_CONTAINED\_BY* (16, 10 in hexadecimal);
* *DOCUMENT\_POSITION\_IMPLEMENTATION\_SPECIFIC* (32, 20 in hexadecimal).

The *compareDocumentPosition(other)* method must run these steps:

1. Let *reference* be the [context object](https://dom.spec.whatwg.org/#context-object).
2. If *other* and *reference* are the same object, return zero.
3. If *other* and *reference* are not in the same [tree](https://dom.spec.whatwg.org/#concept-tree), return the result of adding [DOCUMENT\_POSITION\_DISCONNECTED](https://dom.spec.whatwg.org/#dom-node-document_position_disconnected), [DOCUMENT\_POSITION\_IMPLEMENTATION\_SPECIFIC](https://dom.spec.whatwg.org/#dom-node-document_position_implementation_specific), and either [DOCUMENT\_POSITION\_PRECEDING](https://dom.spec.whatwg.org/#dom-node-document_position_preceding) or [DOCUMENT\_POSITION\_FOLLOWING](https://dom.spec.whatwg.org/#dom-node-document_position_following), with the constraint that this is to be consistent, together.

Whether to return [DOCUMENT\_POSITION\_PRECEDING](https://dom.spec.whatwg.org/#dom-node-document_position_preceding) or [DOCUMENT\_POSITION\_FOLLOWING](https://dom.spec.whatwg.org/#dom-node-document_position_following) is typically implemented via pointer comparison. In JavaScript implementations Math.random() can be used.

1. If *other* is an [ancestor](https://dom.spec.whatwg.org/#concept-tree-ancestor) of *reference*, return the result of adding [DOCUMENT\_POSITION\_CONTAINS](https://dom.spec.whatwg.org/#dom-node-document_position_contains) to [DOCUMENT\_POSITION\_PRECEDING](https://dom.spec.whatwg.org/#dom-node-document_position_preceding).
2. If *other* is a [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) of *reference*, return the result of adding [DOCUMENT\_POSITION\_CONTAINED\_BY](https://dom.spec.whatwg.org/#dom-node-document_position_contained_by) to [DOCUMENT\_POSITION\_FOLLOWING](https://dom.spec.whatwg.org/#dom-node-document_position_following).
3. If *other* is [preceding](https://dom.spec.whatwg.org/#concept-tree-preceding) *reference* return [DOCUMENT\_POSITION\_PRECEDING](https://dom.spec.whatwg.org/#dom-node-document_position_preceding).
4. Return [DOCUMENT\_POSITION\_FOLLOWING](https://dom.spec.whatwg.org/#dom-node-document_position_following).

The *contains(other)* method must return true if *other* is an [inclusive descendant](https://dom.spec.whatwg.org/#concept-tree-inclusive-descendant) of the [context object](https://dom.spec.whatwg.org/#context-object), and false otherwise (including when *other* is null).

To *locate a namespace prefix* for an *element* using *namespace* run these steps:

1. If *element*’s [namespace](https://dom.spec.whatwg.org/#concept-element-namespace) is *namespace* and its [namespace prefix](https://dom.spec.whatwg.org/#concept-element-namespace-prefix) is not null, return its [namespace prefix](https://dom.spec.whatwg.org/#concept-element-namespace-prefix).
2. If *element* [has](https://dom.spec.whatwg.org/#concept-element-attribute-has) an [attribute](https://dom.spec.whatwg.org/#concept-attribute) whose [namespace prefix](https://dom.spec.whatwg.org/#concept-attribute-namespace-prefix) is "xmlns" and [value](https://dom.spec.whatwg.org/#concept-attribute-value) is *namespace*, then return *element*’s first such [attribute](https://dom.spec.whatwg.org/#concept-attribute)’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name).
3. If *element*’s [parent element](https://dom.spec.whatwg.org/#parent-element) is not null, return the result of running [locate a namespace prefix](https://dom.spec.whatwg.org/#locate-a-namespace-prefix) on that [element](https://dom.spec.whatwg.org/#concept-element) using *namespace*. Otherwise, return null.

To *locate a namespace* for a *node* using *prefix* depends on *node*:

[Element](https://dom.spec.whatwg.org/#element)

1. If its [namespace](https://dom.spec.whatwg.org/#concept-element-namespace) is not null and its [namespace prefix](https://dom.spec.whatwg.org/#concept-element-namespace-prefix) is *prefix*, return [namespace](https://dom.spec.whatwg.org/#concept-element-namespace).
2. If it [has](https://dom.spec.whatwg.org/#concept-element-attribute-has) an [attribute](https://dom.spec.whatwg.org/#concept-attribute) whose [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace) is the [XMLNS namespace](https://dom.spec.whatwg.org/#xmlns-namespace), [namespace prefix](https://dom.spec.whatwg.org/#concept-attribute-namespace-prefix) is "xmlns" and [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name) is *prefix*, or if *prefix* is null and it [has](https://dom.spec.whatwg.org/#concept-element-attribute-has) an [attribute](https://dom.spec.whatwg.org/#concept-attribute) whose [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace) is the [XMLNS namespace](https://dom.spec.whatwg.org/#xmlns-namespace), [namespace prefix](https://dom.spec.whatwg.org/#concept-attribute-namespace-prefix) is null and [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name) is "xmlns":
   1. Let *value* be its [value](https://dom.spec.whatwg.org/#concept-attribute-value) if it is not the empty string, and null otherwise.
   2. Return *value*.
3. If its [parent element](https://dom.spec.whatwg.org/#parent-element) is null, return null.
4. Return the result of running [locate a namespace](https://dom.spec.whatwg.org/#locate-a-namespace) on its [parent element](https://dom.spec.whatwg.org/#parent-element) using *prefix*.

[Document](https://dom.spec.whatwg.org/#document)

1. If its [document element](https://dom.spec.whatwg.org/#document-element) is null, return null.
2. Return the result of running [locate a namespace](https://dom.spec.whatwg.org/#locate-a-namespace) on its [document element](https://dom.spec.whatwg.org/#document-element) using *prefix*.

[DocumentType](https://dom.spec.whatwg.org/#documenttype)

[DocumentFragment](https://dom.spec.whatwg.org/#documentfragment)

[ShadowRoot](https://dom.spec.whatwg.org/#shadowroot)

Return null.

Any other node

1. If its [parent element](https://dom.spec.whatwg.org/#parent-element) is null, return null.
2. Return the result of running [locate a namespace](https://dom.spec.whatwg.org/#locate-a-namespace) on its [parent element](https://dom.spec.whatwg.org/#parent-element) using *prefix*.

The *lookupPrefix(namespace)* method must run these steps:

1. If *namespace* is null or the empty string, return null.
2. Otherwise it depends on the [context object](https://dom.spec.whatwg.org/#context-object):

[Element](https://dom.spec.whatwg.org/#element)

Return the result of [locating a namespace prefix](https://dom.spec.whatwg.org/#locate-a-namespace-prefix) for the node using *namespace*.

[Document](https://dom.spec.whatwg.org/#document)

Return the result of [locating a namespace prefix](https://dom.spec.whatwg.org/#locate-a-namespace-prefix) for its [document element](https://dom.spec.whatwg.org/#document-element), if that is not null, and null otherwise.

[DocumentType](https://dom.spec.whatwg.org/#documenttype)

[DocumentFragment](https://dom.spec.whatwg.org/#documentfragment)

[ShadowRoot](https://dom.spec.whatwg.org/#shadowroot)

Return null.

Any other node

Return the result of [locating a namespace prefix](https://dom.spec.whatwg.org/#locate-a-namespace-prefix) for its [parent element](https://dom.spec.whatwg.org/#parent-element), or if that is null, null.

The *lookupNamespaceURI(prefix)* method must run these steps:

1. If *prefix* is the empty string, set it to null.
2. Return the result of running [locate a namespace](https://dom.spec.whatwg.org/#locate-a-namespace) for the [context object](https://dom.spec.whatwg.org/#context-object) using *prefix*.

The *isDefaultNamespace(namespace)* method must run these steps:

1. If *namespace* is the empty string, set it to null.
2. Let *defaultNamespace* be the result of running [locate a namespace](https://dom.spec.whatwg.org/#locate-a-namespace) for the [context object](https://dom.spec.whatwg.org/#context-object) using null.
3. Return true if *defaultNamespace* is the same as *namespace*, and false otherwise.

The *insertBefore(node, child)* method must return the result of [pre-inserting](https://dom.spec.whatwg.org/#concept-node-pre-insert) *node* into the [context object](https://dom.spec.whatwg.org/#context-object) before *child*.

The *appendChild(node)* method must return the result of [appending](https://dom.spec.whatwg.org/#concept-node-append) *node* to the [context object](https://dom.spec.whatwg.org/#context-object).

The *replaceChild(node, child)* method must return the result of [replacing](https://dom.spec.whatwg.org/#concept-node-replace) *child* with *node* within the [context object](https://dom.spec.whatwg.org/#context-object).

The *removeChild(child)* method must return the result of [pre-removing](https://dom.spec.whatwg.org/#concept-node-pre-remove) *child* from the [context object](https://dom.spec.whatwg.org/#context-object).

The *list of elements with qualified name qualifiedName* for a [node](https://dom.spec.whatwg.org/#concept-node) *root* is the [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) returned by the following algorithm:

1. If *qualifiedName* is "\*" (U+002A), return a [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) rooted at *root*, whose filter matches only [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) [elements](https://dom.spec.whatwg.org/#concept-element).
2. Otherwise, if *root*’s [node document](https://dom.spec.whatwg.org/#concept-node-document) is an [HTML document](https://dom.spec.whatwg.org/#html-document), return a [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) rooted at *root*, whose filter matches the following [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) [elements](https://dom.spec.whatwg.org/#concept-element):
   * Whose [namespace](https://dom.spec.whatwg.org/#concept-element-namespace) is the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace) and whose [qualified name](https://dom.spec.whatwg.org/#concept-element-qualified-name) is *qualifiedName*, [converted to ASCII lowercase](https://dom.spec.whatwg.org/#converted-to-ascii-lowercase).
   * Whose [namespace](https://dom.spec.whatwg.org/#concept-element-namespace) is *not* the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace) and whose [qualified name](https://dom.spec.whatwg.org/#concept-element-qualified-name) is *qualifiedName*.
3. Otherwise, return a [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) rooted at *root*, whose filter matches [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) [elements](https://dom.spec.whatwg.org/#concept-element) whose [qualified name](https://dom.spec.whatwg.org/#concept-element-qualified-name) is *qualifiedName*.

When invoked with the same argument, and as long as *root*’s [node document](https://dom.spec.whatwg.org/#concept-node-document)’s [type](https://dom.spec.whatwg.org/#concept-document-type) has not changed, the same [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) object may be returned as returned by an earlier call.

The *list of elements with namespace namespace and local name localName* for a [node](https://dom.spec.whatwg.org/#concept-node) *root* is the [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) returned by the following algorithm:

1. If *namespace* is the empty string, set it to null.
2. If both *namespace* and *localName* are "\*" (U+002A), return a [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) rooted at *root*, whose filter matches [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) [elements](https://dom.spec.whatwg.org/#concept-element).
3. Otherwise, if *namespace* is "\*" (U+002A), return a [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) rooted at *root*, whose filter matches [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) [elements](https://dom.spec.whatwg.org/#concept-element) whose [local name](https://dom.spec.whatwg.org/#concept-element-local-name) is *localName*.
4. Otherwise, if *localName* is "\*" (U+002A), return a [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) rooted at *root*, whose filter matches [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) [elements](https://dom.spec.whatwg.org/#concept-element) whose [namespace](https://dom.spec.whatwg.org/#concept-element-namespace) is *namespace*.
5. Otherwise, return a [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) rooted at *root*, whose filter matches [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) [elements](https://dom.spec.whatwg.org/#concept-element) whose [namespace](https://dom.spec.whatwg.org/#concept-element-namespace) is *namespace* and [local name](https://dom.spec.whatwg.org/#concept-element-local-name) is *localName*.

When invoked with the same arguments, the same [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) object may be returned as returned by an earlier call.

The *list of elements with class names classNames* for a [node](https://dom.spec.whatwg.org/#concept-node) *root* is the [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) returned by the following algorithm:

1. Let *classes* be the result of running the [ordered set parser](https://dom.spec.whatwg.org/#concept-ordered-set-parser) on *classNames*.
2. If *classes* is the empty set, return an empty [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection).
3. Return a [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) rooted at *root*, whose filter matches [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) [elements](https://dom.spec.whatwg.org/#concept-element) that have all their [classes](https://dom.spec.whatwg.org/#concept-class) in *classes*.

The comparisons for the [classes](https://dom.spec.whatwg.org/#concept-class) must be done in an [ASCII case-insensitive](https://dom.spec.whatwg.org/#ascii-case-insensitive) manner if *root*’s [node document](https://dom.spec.whatwg.org/#concept-node-document)’s [mode](https://dom.spec.whatwg.org/#concept-document-mode) is "quirks", and in a [case-sensitive](https://dom.spec.whatwg.org/#case-sensitive) manner otherwise.

When invoked with the same argument, the same [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) object may be returned as returned by an earlier call.

**4.5. Interface** [**Document**](https://dom.spec.whatwg.org/#document)

[[Constructor](https://dom.spec.whatwg.org/#dom-document-document),

Exposed=Window]

interface *Document* : [Node](https://dom.spec.whatwg.org/#node) {

[SameObject] readonly attribute [DOMImplementation](https://dom.spec.whatwg.org/#domimplementation) [implementation](https://dom.spec.whatwg.org/#dom-document-implementation);

readonly attribute DOMString [URL](https://dom.spec.whatwg.org/#dom-document-url);

readonly attribute DOMString [documentURI](https://dom.spec.whatwg.org/#dom-document-documenturi);

readonly attribute DOMString [origin](https://dom.spec.whatwg.org/#dom-document-origin);

readonly attribute DOMString [compatMode](https://dom.spec.whatwg.org/#dom-document-compatmode);

readonly attribute DOMString [characterSet](https://dom.spec.whatwg.org/#dom-document-characterset);

readonly attribute DOMString [charset](https://dom.spec.whatwg.org/#dom-document-charset); // historical alias of .characterSet

readonly attribute DOMString [inputEncoding](https://dom.spec.whatwg.org/#dom-document-inputencoding); // historical alias of .characterSet

readonly attribute DOMString [contentType](https://dom.spec.whatwg.org/#dom-document-contenttype);

readonly attribute [DocumentType](https://dom.spec.whatwg.org/#documenttype)? [doctype](https://dom.spec.whatwg.org/#dom-document-doctype);

readonly attribute [Element](https://dom.spec.whatwg.org/#element)? [documentElement](https://dom.spec.whatwg.org/#dom-document-documentelement);

[HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) [getElementsByTagName](https://dom.spec.whatwg.org/#dom-document-getelementsbytagname)(DOMString *qualifiedName*);

[HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) [getElementsByTagNameNS](https://dom.spec.whatwg.org/#dom-document-getelementsbytagnamens)(DOMString? *namespace*, DOMString *localName*);

[HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) [getElementsByClassName](https://dom.spec.whatwg.org/#dom-document-getelementsbyclassname)(DOMString *classNames*);

[NewObject] [Element](https://dom.spec.whatwg.org/#element) [createElement](https://dom.spec.whatwg.org/#dom-document-createelement)(DOMString *localName*, optional [ElementCreationOptions](https://dom.spec.whatwg.org/#dictdef-elementcreationoptions) *options*);

[NewObject] [Element](https://dom.spec.whatwg.org/#element) [createElementNS](https://dom.spec.whatwg.org/#dom-document-createelementns)(DOMString? *namespace*, DOMString *qualifiedName*, optional [ElementCreationOptions](https://dom.spec.whatwg.org/#dictdef-elementcreationoptions) *options*);

[NewObject] [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [createDocumentFragment](https://dom.spec.whatwg.org/#dom-document-createdocumentfragment)();

[NewObject] [Text](https://dom.spec.whatwg.org/#text) [createTextNode](https://dom.spec.whatwg.org/#dom-document-createtextnode)(DOMString *data*);

[NewObject] [Comment](https://dom.spec.whatwg.org/#comment) [createComment](https://dom.spec.whatwg.org/#dom-document-createcomment)(DOMString *data*);

[NewObject] [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction) [createProcessingInstruction](https://dom.spec.whatwg.org/#dom-document-createprocessinginstruction)(DOMString *target*, DOMString *data*);

[CEReactions, NewObject] [Node](https://dom.spec.whatwg.org/#node) [importNode](https://dom.spec.whatwg.org/#dom-document-importnode)([Node](https://dom.spec.whatwg.org/#node) *node*, optional boolean *deep* = false);

[CEReactions] [Node](https://dom.spec.whatwg.org/#node) [adoptNode](https://dom.spec.whatwg.org/#dom-document-adoptnode)([Node](https://dom.spec.whatwg.org/#node) *node*);

[NewObject] [Attr](https://dom.spec.whatwg.org/#attr) [createAttribute](https://dom.spec.whatwg.org/#dom-document-createattribute)(DOMString *localName*);

[NewObject] [Attr](https://dom.spec.whatwg.org/#attr) [createAttributeNS](https://dom.spec.whatwg.org/#dom-document-createattributens)(DOMString? *namespace*, DOMString *qualifiedName*);

[NewObject] [Event](https://dom.spec.whatwg.org/#event) [createEvent](https://dom.spec.whatwg.org/#dom-document-createevent)(DOMString interface);

[NewObject] [Range](https://dom.spec.whatwg.org/#range) [createRange](https://dom.spec.whatwg.org/#dom-document-createrange)();

// NodeFilter.SHOW\_ALL = 0xFFFFFFFF

[NewObject] [NodeIterator](https://dom.spec.whatwg.org/#nodeiterator) [createNodeIterator](https://dom.spec.whatwg.org/#dom-document-createnodeiterator)([Node](https://dom.spec.whatwg.org/#node) *root*, optional unsigned long *whatToShow* = 0xFFFFFFFF, optional [NodeFilter](https://dom.spec.whatwg.org/#callbackdef-nodefilter)? *filter* = null);

[NewObject] [TreeWalker](https://dom.spec.whatwg.org/#treewalker) [createTreeWalker](https://dom.spec.whatwg.org/#dom-document-createtreewalker)([Node](https://dom.spec.whatwg.org/#node) *root*, optional unsigned long *whatToShow* = 0xFFFFFFFF, optional [NodeFilter](https://dom.spec.whatwg.org/#callbackdef-nodefilter)? *filter* = null);

};

[Exposed=Window]

interface *XMLDocument* : [Document](https://dom.spec.whatwg.org/#document) {};

dictionary *ElementCreationOptions* {

DOMString *is*;

};

[Document](https://dom.spec.whatwg.org/#document) [nodes](https://dom.spec.whatwg.org/#concept-node) are simply known as *documents*.

Each [document](https://dom.spec.whatwg.org/#concept-document) has an associated *base URL* (a [URL](https://url.spec.whatwg.org/#concept-url)), *encoding* (an [encoding](https://encoding.spec.whatwg.org/#encoding)), *content type* (a string), *URL* (a [URL](https://url.spec.whatwg.org/#concept-url)), *type* ("xml" or "html"), and *mode* ("no-quirks", "quirks", or "limited-quirks"). [[ENCODING]](https://dom.spec.whatwg.org/#biblio-encoding) [[URL]](https://dom.spec.whatwg.org/#biblio-url)

Unless stated otherwise, a [document](https://dom.spec.whatwg.org/#concept-document)’s [base URL](https://dom.spec.whatwg.org/#concept-document-base-url) is "about:blank", [encoding](https://dom.spec.whatwg.org/#concept-document-encoding) is the [utf-8](https://encoding.spec.whatwg.org/#utf-8) [encoding](https://encoding.spec.whatwg.org/#encoding), [content type](https://dom.spec.whatwg.org/#concept-document-content-type) is "application/xml", [URL](https://dom.spec.whatwg.org/#concept-document-url) is "about:blank", [type](https://dom.spec.whatwg.org/#concept-document-type) is "xml", and its [mode](https://dom.spec.whatwg.org/#concept-document-mode) is "no-quirks".

Unless stated otherwise, a [document](https://dom.spec.whatwg.org/#concept-document)’s [origin](https://html.spec.whatwg.org/multipage/browsers.html#concept-origin) is a new [opaque origin](https://html.spec.whatwg.org/multipage/browsers.html#concept-origin-opaque). [[HTML]](https://dom.spec.whatwg.org/#biblio-html)

A [document](https://dom.spec.whatwg.org/#concept-document) is said to be an *XML document* if its [type](https://dom.spec.whatwg.org/#concept-document-type) is "xml", and an *HTML document* otherwise. Whether a [document](https://dom.spec.whatwg.org/#concept-document) is an [HTML document](https://dom.spec.whatwg.org/#html-document) or an [XML document](https://dom.spec.whatwg.org/#xml-document) affects the behavior of certain APIs.

A [document](https://dom.spec.whatwg.org/#concept-document) is said to be in *no-quirks mode* if its [mode](https://dom.spec.whatwg.org/#concept-document-mode) is "no-quirks", *quirks mode* if its [mode](https://dom.spec.whatwg.org/#concept-document-mode) is "quirks", and *limited-quirks mode* if its [mode](https://dom.spec.whatwg.org/#concept-document-mode) is "limited-quirks".

The mode is only ever changed from the default if the [document](https://dom.spec.whatwg.org/#concept-document) is created by the [HTML parser](https://html.spec.whatwg.org/multipage/syntax.html#html-parser), based on the presence, absence, or value of the DOCTYPE string. [[HTML]](https://dom.spec.whatwg.org/#biblio-html)

[No-quirks mode](https://dom.spec.whatwg.org/#concept-document-no-quirks) was originally known as "standards mode" and [limited-quirks mode](https://dom.spec.whatwg.org/#concept-document-limited-quirks) was once known as "almost standards mode". They have been renamed because their details are now defined by standards. (And because Ian Hickson vetoed their original names on the basis that they are nonsensical.)

A [document](https://dom.spec.whatwg.org/#concept-document)’s [get the parent](https://dom.spec.whatwg.org/#get-the-parent) algorithm, given an *event*, returns null if *event*’s type attribute value is "load" or [document](https://dom.spec.whatwg.org/#concept-document) does not have a [browsing context](https://html.spec.whatwg.org/multipage/browsers.html#concept-document-bc), and the [document](https://dom.spec.whatwg.org/#concept-document)’s [associated](https://html.spec.whatwg.org/multipage/browsers.html#concept-document-window) [Window](https://html.spec.whatwg.org/multipage/browsers.html#window) object otherwise.

*document* = new [Document()](https://dom.spec.whatwg.org/#dom-document-document)

Returns a new [document](https://dom.spec.whatwg.org/#concept-document).

*document* . [implementation](https://dom.spec.whatwg.org/#dom-document-implementation)

Returns *document*’s [DOMImplementation](https://dom.spec.whatwg.org/#domimplementation) object.

*document* . [URL](https://dom.spec.whatwg.org/#dom-document-url)

*document* . [documentURI](https://dom.spec.whatwg.org/#dom-document-documenturi)

Returns *document*’s [URL](https://dom.spec.whatwg.org/#concept-document-url).

*document* . [origin](https://dom.spec.whatwg.org/#dom-document-origin)

Returns *document*’s [origin](https://html.spec.whatwg.org/multipage/browsers.html#concept-origin).

*document* . [compatMode](https://dom.spec.whatwg.org/#dom-document-compatmode)

Returns the string "BackCompat" if *document*’s [mode](https://dom.spec.whatwg.org/#concept-document-mode) is "quirks", and "CSS1Compat" otherwise.

*document* . [characterSet](https://dom.spec.whatwg.org/#dom-document-characterset)

Returns *document*’s [encoding](https://dom.spec.whatwg.org/#concept-document-encoding).

*document* . [contentType](https://dom.spec.whatwg.org/#dom-document-contenttype)

Returns *document*’s [content type](https://dom.spec.whatwg.org/#concept-document-content-type).

The *Document()* constructor, when invoked, must return a new [document](https://dom.spec.whatwg.org/#concept-document) whose [origin](https://html.spec.whatwg.org/multipage/browsers.html#concept-origin) is the [origin](https://html.spec.whatwg.org/multipage/browsers.html#concept-origin) of the global object’s associated [document](https://dom.spec.whatwg.org/#concept-document). [[HTML]](https://dom.spec.whatwg.org/#biblio-html)

Unlike [createDocument()](https://dom.spec.whatwg.org/#dom-domimplementation-createdocument) this constructor does not return an [XMLDocument](https://dom.spec.whatwg.org/#xmldocument) object, but a [document](https://dom.spec.whatwg.org/#concept-document) ([Document](https://dom.spec.whatwg.org/#document) object).

The *implementation* attribute’s getter must return the [DOMImplementation](https://dom.spec.whatwg.org/#domimplementation) object that is associated with the [document](https://dom.spec.whatwg.org/#concept-document).

The *URL* attribute’s getter and *documentURI* attribute’s getter must return the [URL](https://dom.spec.whatwg.org/#concept-document-url), [serialized](https://url.spec.whatwg.org/#concept-url-serializer).

The *origin* attribute’s getter must return the [Unicode serialization](https://html.spec.whatwg.org/multipage/browsers.html#unicode-serialisation-of-an-origin) of [context object](https://dom.spec.whatwg.org/#context-object)’s [origin](https://html.spec.whatwg.org/multipage/browsers.html#concept-origin).

The *compatMode* attribute’s getter must return "BackCompat" if [context object](https://dom.spec.whatwg.org/#context-object)’s [mode](https://dom.spec.whatwg.org/#concept-document-mode) is "quirks", and "CSS1Compat" otherwise.

The *characterSet* attribute’s getter, *charset* attribute’s getter, and *inputEncoding* attribute’s getter, must return [context object](https://dom.spec.whatwg.org/#context-object)’s [encoding](https://dom.spec.whatwg.org/#concept-document-encoding)’s [name](https://encoding.spec.whatwg.org/#name).

The *contentType* attribute’s getter must return the [content type](https://dom.spec.whatwg.org/#concept-document-content-type).

*document* . [doctype](https://dom.spec.whatwg.org/#dom-document-doctype)

Returns the [doctype](https://dom.spec.whatwg.org/#concept-doctype) or null if there is none.

*document* . [documentElement](https://dom.spec.whatwg.org/#dom-document-documentelement)

Returns the [document element](https://dom.spec.whatwg.org/#document-element).

*collection* = *document* . [getElementsByTagName(qualifiedName)](https://dom.spec.whatwg.org/#dom-document-getelementsbytagname)

If *qualifiedName* is "\*" returns a [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) of all [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) [elements](https://dom.spec.whatwg.org/#concept-element).

Otherwise, returns a [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) of all [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) [elements](https://dom.spec.whatwg.org/#concept-element) whose [qualified name](https://dom.spec.whatwg.org/#concept-element-qualified-name) is *qualifiedName*. (Matches case-insensitively against [elements](https://dom.spec.whatwg.org/#concept-element) in the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace) within an [HTML document](https://dom.spec.whatwg.org/#html-document).)

*collection* = *document* . [getElementsByTagNameNS(namespace, localName)](https://dom.spec.whatwg.org/#dom-document-getelementsbytagnamens)

If *namespace* and *localName* are "\*" returns a [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) of all [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) [elements](https://dom.spec.whatwg.org/#concept-element).

If only *namespace* is "\*" returns a [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) of all [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) [elements](https://dom.spec.whatwg.org/#concept-element) whose [local name](https://dom.spec.whatwg.org/#concept-element-local-name) is *localName*.

If only *localName* is "\*" returns a [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) of all [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) [elements](https://dom.spec.whatwg.org/#concept-element) whose [namespace](https://dom.spec.whatwg.org/#concept-element-namespace) is *namespace*.

Otherwise, returns a [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) of all [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) [elements](https://dom.spec.whatwg.org/#concept-element) whose [namespace](https://dom.spec.whatwg.org/#concept-element-namespace) is *namespace* and [local name](https://dom.spec.whatwg.org/#concept-element-local-name) is *localName*.

*collection* = *document* . [getElementsByClassName(classNames)](https://dom.spec.whatwg.org/#dom-document-getelementsbyclassname)

*collection* = *element* . [getElementsByClassName(classNames)](https://dom.spec.whatwg.org/#dom-element-getelementsbyclassname)

Returns a [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) of the [elements](https://dom.spec.whatwg.org/#concept-element) in the object on which the method was invoked (a [document](https://dom.spec.whatwg.org/#concept-document) or an [element](https://dom.spec.whatwg.org/#concept-element)) that have all the classes given by *classNames*. The *classNames* argument is interpreted as a space-separated list of classes.

The *doctype* attribute’s getter must return the [child](https://dom.spec.whatwg.org/#concept-tree-child) of the [document](https://dom.spec.whatwg.org/#concept-document) that is a [doctype](https://dom.spec.whatwg.org/#concept-doctype), and null otherwise.

The *documentElement* attribute’s getter must return the [document element](https://dom.spec.whatwg.org/#document-element).

The *getElementsByTagName(qualifiedName)* method, when invoked, must return the [list of elements with qualified name *qualifiedName*](https://dom.spec.whatwg.org/#concept-getelementsbytagname) for the [context object](https://dom.spec.whatwg.org/#context-object).

Thus, in an [HTML document](https://dom.spec.whatwg.org/#html-document), document.getElementsByTagName("FOO") will match <FOO> elements that are not in the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace), and <foo> elements that are in the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace), but not <FOO> elements that are in the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace).

The *getElementsByTagNameNS(namespace, localName)* method, when invoked, must return the [list of elements with namespace *namespace* and local name *localName*](https://dom.spec.whatwg.org/#concept-getelementsbytagnamens) for the [context object](https://dom.spec.whatwg.org/#context-object).

The *getElementsByClassName(classNames)* method, when invoked, must return the [list of elements with class names *classNames*](https://dom.spec.whatwg.org/#concept-getelementsbyclassname) for the [context object](https://dom.spec.whatwg.org/#context-object).

Given the following XHTML fragment:

<div id="example">

<p id="p1" class="aaa bbb"/>

<p id="p2" class="aaa ccc"/>

<p id="p3" class="bbb ccc"/>

</div>

A call to document.getElementById("example").getElementsByClassName("aaa") would return a [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) with the two paragraphs p1 and p2 in it.

A call to getElementsByClassName("ccc bbb") would only return one node, however, namely p3. A call to document.getElementById("example").getElementsByClassName("bbb  ccc ") would return the same thing.

A call to getElementsByClassName("aaa,bbb") would return no nodes; none of the elements above are in the aaa,bbb class.

*element* = *document* . [createElement(localName [, options])](https://dom.spec.whatwg.org/#dom-document-createelement)

Returns an [element](https://dom.spec.whatwg.org/#concept-element) in the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace) with *localName* as [local name](https://dom.spec.whatwg.org/#concept-element-local-name). (In an [HTML document](https://dom.spec.whatwg.org/#html-document) *localName* is lowercased.)

If *localName* does not match the [Name](https://www.w3.org/TR/xml/#NT-Name) production an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror) will be thrown.

When supplied, *options*’ is member can be used to create a [customized built-in element](https://html.spec.whatwg.org/multipage/scripting.html#customized-built-in-element).

*element* = *document* . [createElementNS(namespace, qualifiedName [, options])](https://dom.spec.whatwg.org/#dom-document-createelementns)

Returns an [element](https://dom.spec.whatwg.org/#concept-element) with [namespace](https://dom.spec.whatwg.org/#concept-element-namespace) *namespace*. Its [namespace prefix](https://dom.spec.whatwg.org/#concept-element-namespace-prefix) will be everything before ":" (U+003E) in *qualifiedName* or null. Its [local name](https://dom.spec.whatwg.org/#concept-element-local-name) will be everything after ":" (U+003E) in *qualifiedName* or *qualifiedName*.

If *localName* does not match the [Name](https://www.w3.org/TR/xml/#NT-Name) production an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror) will be thrown.

If one of the following conditions is true a [NamespaceError](https://heycam.github.io/webidl/#namespaceerror) will be thrown:

* *localName* does not match the [QName](https://www.w3.org/TR/xml-names/#NT-QName) production.
* [Namespace prefix](https://dom.spec.whatwg.org/#concept-element-namespace-prefix) is not null and *namespace* is the empty string.
* [Namespace prefix](https://dom.spec.whatwg.org/#concept-element-namespace-prefix) is "xml" and *namespace* is not the [XML namespace](https://dom.spec.whatwg.org/#xml-namespace).
* *qualifiedName* or [namespace prefix](https://dom.spec.whatwg.org/#concept-element-namespace-prefix) is "xmlns" and *namespace* is not the [XMLNS namespace](https://dom.spec.whatwg.org/#xmlns-namespace).
* *namespace* is the [XMLNS namespace](https://dom.spec.whatwg.org/#xmlns-namespace) and neither *qualifiedName* nor [namespace prefix](https://dom.spec.whatwg.org/#concept-element-namespace-prefix) is "xmlns".

When supplied, *options*’ is member can be used to create a [customized built-in element](https://html.spec.whatwg.org/multipage/scripting.html#customized-built-in-element).

*documentFragment* = *document* . [createDocumentFragment()](https://dom.spec.whatwg.org/#dom-document-createdocumentfragment)

Returns a [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [node](https://dom.spec.whatwg.org/#concept-node).

*text* = *document* . [createTextNode(data)](https://dom.spec.whatwg.org/#dom-document-createtextnode)

Returns a [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node) whose [data](https://dom.spec.whatwg.org/#concept-cd-data) is *data*.

*comment* = *document* . [createComment(data)](https://dom.spec.whatwg.org/#dom-document-createcomment)

Returns a [Comment](https://dom.spec.whatwg.org/#comment) [node](https://dom.spec.whatwg.org/#concept-node) whose [data](https://dom.spec.whatwg.org/#concept-cd-data) is *data*.

*processingInstruction* = *document* . [createProcessingInstruction(target, data)](https://dom.spec.whatwg.org/#dom-document-createprocessinginstruction)

Returns a [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction) [node](https://dom.spec.whatwg.org/#concept-node) whose [target](https://dom.spec.whatwg.org/#concept-pi-target) is *target* and [data](https://dom.spec.whatwg.org/#concept-cd-data) is *data*. If *target* does not match the [Name](https://www.w3.org/TR/xml/#NT-Name) production an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror) will be thrown. If *data* contains "?>" an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror) will be thrown.

The *element interface* for any *name* and *namespace* is [Element](https://dom.spec.whatwg.org/#element), unless stated otherwise.

The HTML Standard will e.g. define that for html and the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace), the [HTMLHtmlElement](https://html.spec.whatwg.org/multipage/semantics.html#htmlhtmlelement) interface is used. [[HTML]](https://dom.spec.whatwg.org/#biblio-html)

The *createElement(localName, options)* method, when invoked, must run these steps:

1. If *localName* does not match the [Name](https://www.w3.org/TR/xml/#NT-Name) production, then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror).
2. If the [context object](https://dom.spec.whatwg.org/#context-object) is an [HTML document](https://dom.spec.whatwg.org/#html-document), let *localName* be [converted to ASCII lowercase](https://dom.spec.whatwg.org/#converted-to-ascii-lowercase).
3. Let *is* be the value of is member of *options*, or null if no such member exists.
4. Let *definition* be the result of [looking up a custom element definition](https://html.spec.whatwg.org/multipage/scripting.html#look-up-a-custom-element-definition), given the [context object](https://dom.spec.whatwg.org/#context-object), the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace), *localName*, and *is*.
5. If *is* is non-null and *definition* is null, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotFoundError](https://heycam.github.io/webidl/#notfounderror).
6. Let *namespace* be the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace), if the [context object](https://dom.spec.whatwg.org/#context-object)’s [content type](https://dom.spec.whatwg.org/#concept-document-content-type) is "text/html" or "application/xhtml+xml", and null otherwise.
7. Let *element* be the result of [creating an element](https://dom.spec.whatwg.org/#concept-create-element) given the [context object](https://dom.spec.whatwg.org/#context-object), *localName*, *namespace*, null, *is*, and with the *synchronous custom elements* flag set. Rethrow any exceptions.
8. If *is* is non-null, then [set an attribute value](https://dom.spec.whatwg.org/#concept-element-attributes-set-value) for *element* using "is" and *is*.
9. Return *element*.

The *internal createElementNS steps*, given *document*, *namespace*, *qualifiedName*, and *options*, are as follows:

1. Let *namespace*, *prefix*, and *localName* be the result of passing *namespace* and *qualifiedName* to [validate and extract](https://dom.spec.whatwg.org/#validate-and-extract). Rethrow any exceptions.
2. Let *is* be the value of is member of *options*, or null if no such member exists.
3. Let *definition* be the result of [looking up a custom element definition](https://html.spec.whatwg.org/multipage/scripting.html#look-up-a-custom-element-definition), given the [context object](https://dom.spec.whatwg.org/#context-object), *namespace*, *localName*, and *is*.
4. If *is* is non-null and *definition* is null, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotFoundError](https://heycam.github.io/webidl/#notfounderror).
5. Let *element* be the result of [creating an element](https://dom.spec.whatwg.org/#concept-create-element) given *document*, *localName*, *namespace*, *prefix*, *is*, and with the *synchronous custom elements* flag set. Rethrow any exceptions.
6. If *is* is non-null, then [set an attribute value](https://dom.spec.whatwg.org/#concept-element-attributes-set-value) for *element* using "is" and *is*.
7. Return *element*.

The *createElementNS(namespace, qualifiedName, options)* method, when invoked, must return the result of running the [internal createElementNS steps](https://dom.spec.whatwg.org/#internal-createelementns-steps), given [context object](https://dom.spec.whatwg.org/#context-object), *namespace*, *qualifiedName*, and *options*.

The *createDocumentFragment()* method, when invoked, must return a new [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [node](https://dom.spec.whatwg.org/#concept-node) with its [node document](https://dom.spec.whatwg.org/#concept-node-document) set to the [context object](https://dom.spec.whatwg.org/#context-object).

The *createTextNode(data)* method, when invoked, must return a new [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node) with its [data](https://dom.spec.whatwg.org/#concept-cd-data) set to *data* and [node document](https://dom.spec.whatwg.org/#concept-node-document) set to the [context object](https://dom.spec.whatwg.org/#context-object).

No check is performed that *data* consists of characters that match the [Char](https://www.w3.org/TR/xml/#NT-Char) production.

The *createComment(data)* method, when invoked, must return a new [Comment](https://dom.spec.whatwg.org/#comment) [node](https://dom.spec.whatwg.org/#concept-node) with its [data](https://dom.spec.whatwg.org/#concept-cd-data) set to *data* and [node document](https://dom.spec.whatwg.org/#concept-node-document) set to the [context object](https://dom.spec.whatwg.org/#context-object).

No check is performed that *data* consists of characters that match the [Char](https://www.w3.org/TR/xml/#NT-Char) production or that it contains two adjacent hyphens or ends with a hyphen.

The *createProcessingInstruction(target, data)* method, when invoked, must run these steps:

1. If *target* does not match the [Name](https://www.w3.org/TR/xml/#NT-Name) production, then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror).
2. If *data* contains the string "?>", then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror).
3. Return a new [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction) [node](https://dom.spec.whatwg.org/#concept-node), with [target](https://dom.spec.whatwg.org/#concept-pi-target) set to *target*, [data](https://dom.spec.whatwg.org/#concept-cd-data) set to *data*, and [node document](https://dom.spec.whatwg.org/#concept-node-document) set to the [context object](https://dom.spec.whatwg.org/#context-object).

No check is performed that *target* contains "xml" or ":", or that *data* contains characters that match the [Char](https://www.w3.org/TR/xml/#NT-Char) production.

*clone* = *document* . [importNode(*node* [, *deep* = false])](https://dom.spec.whatwg.org/#dom-document-importnode)

Returns a copy of *node*. If *deep* is true, the copy also includes the *node*’s [descendants](https://dom.spec.whatwg.org/#concept-tree-descendant).

If *node* is a [document](https://dom.spec.whatwg.org/#concept-document) or a [shadow root](https://dom.spec.whatwg.org/#concept-shadow-root), throws a [NotSupportedError](https://heycam.github.io/webidl/#notsupportederror).

*node* = *document* . [adoptNode(node)](https://dom.spec.whatwg.org/#dom-document-adoptnode)

Moves *node* from another [document](https://dom.spec.whatwg.org/#concept-document) and returns it.

If *node* is a [document](https://dom.spec.whatwg.org/#concept-document), throws a [NotSupportedError](https://heycam.github.io/webidl/#notsupportederror) or, if *node* is a [shadow root](https://dom.spec.whatwg.org/#concept-shadow-root), throws a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror).

The *importNode(node, deep)* method, when invoked, must run these steps:

1. If *node* is a [document](https://dom.spec.whatwg.org/#concept-document) or [shadow root](https://dom.spec.whatwg.org/#concept-shadow-root), then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotSupportedError](https://heycam.github.io/webidl/#notsupportederror).
2. Return a [clone](https://dom.spec.whatwg.org/#concept-node-clone) of *node*, with [context object](https://dom.spec.whatwg.org/#context-object) and the *clone children flag* set if *deep* is true.

[Specifications](https://dom.spec.whatwg.org/#other-applicable-specifications) may define *adopting steps* for all or some [nodes](https://dom.spec.whatwg.org/#concept-node). The algorithm is passed *node* and *oldDocument*, as indicated in the [adopt](https://dom.spec.whatwg.org/#concept-node-adopt) algorithm.

To *adopt* a *node* into a *document*, run these steps:

1. Let *oldDocument* be *node*’s [node document](https://dom.spec.whatwg.org/#concept-node-document).
2. If *node*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is not null, [remove](https://dom.spec.whatwg.org/#concept-node-remove) *node* from its [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
3. If *document* is not the same as *oldDocument*, run these substeps:
   1. For each *inclusiveDescendant* in *node*’s [shadow-including inclusive descendants](https://dom.spec.whatwg.org/#concept-shadow-including-inclusive-descendant), in [shadow-including tree order](https://dom.spec.whatwg.org/#concept-shadow-including-tree-order), set *inclusiveDescendant*’s [node document](https://dom.spec.whatwg.org/#concept-node-document) to *document*.
   2. For each *inclusiveDescendant* in *node*’s [shadow-including inclusive descendants](https://dom.spec.whatwg.org/#concept-shadow-including-inclusive-descendant), in [shadow-including tree order](https://dom.spec.whatwg.org/#concept-shadow-including-tree-order), run the [adopting steps](https://dom.spec.whatwg.org/#concept-node-adopt-ext) with *inclusiveDescendant* and *oldDocument*.

The *adoptNode(node)* method, when invoked, must run these steps:

1. If *node* is a [document](https://dom.spec.whatwg.org/#concept-document), then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotSupportedError](https://heycam.github.io/webidl/#notsupportederror).
2. If *node* is a [shadow root](https://dom.spec.whatwg.org/#concept-shadow-root), then [throw](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror).
3. [Adopt](https://dom.spec.whatwg.org/#concept-node-adopt) *node* into the [context object](https://dom.spec.whatwg.org/#context-object).
4. Return *node*.

The *createAttribute(localName)* method, when invoked, must run these steps:

1. If *localName* does not match the [Name](https://www.w3.org/TR/xml/#NT-Name) production in XML, then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror).
2. If the [context object](https://dom.spec.whatwg.org/#context-object) is an [HTML document](https://dom.spec.whatwg.org/#html-document), let *localName* be [converted to ASCII lowercase](https://dom.spec.whatwg.org/#converted-to-ascii-lowercase).
3. Return a new [attribute](https://dom.spec.whatwg.org/#concept-attribute) whose [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name) is *localName*.

The *createAttributeNS(namespace, qualifiedName)* method, when invoked, must run these steps:

1. Let *namespace*, *prefix*, and *localName* be the result of passing *namespace* and *qualifiedName* to [validate and extract](https://dom.spec.whatwg.org/#validate-and-extract). Rethrow any exceptions.
2. Return a new [attribute](https://dom.spec.whatwg.org/#concept-attribute) whose [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace) is *namespace*, [namespace prefix](https://dom.spec.whatwg.org/#concept-attribute-namespace-prefix) is *prefix*, and [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name) is *localName*.

The *createEvent(interface)* method, when invoked, must run these steps:

1. Let *constructor* be null.
2. If *interface* is an [ASCII case-insensitive](https://dom.spec.whatwg.org/#ascii-case-insensitive) match for any of the strings in the first column in the following table, then set *constructor* to the interface in the second column on the same row as the matching string:

| **String** | **Interface** | Notes |
| --- | --- | --- |
| "animationevent" | [AnimationEvent](https://drafts.csswg.org/css-animations-1/#animationevent) | [[CSS3-ANIMATIONS]](https://dom.spec.whatwg.org/#biblio-css3-animations) |
| "beforeunloadevent" | [BeforeUnloadEvent](https://html.spec.whatwg.org/multipage/browsers.html#beforeunloadevent) | [[HTML]](https://dom.spec.whatwg.org/#biblio-html) |
| "closeevent" | [CloseEvent](https://html.spec.whatwg.org/multipage/comms.html#closeevent) |
| "compositionevent" | CompositionEvent | [[UIEVENTS]](https://dom.spec.whatwg.org/#biblio-uievents) |
| "customevent" | [CustomEvent](https://dom.spec.whatwg.org/#customevent) |  |
| "devicemotionevent" | DeviceMotionEvent | [[DEVICE-ORIENTATION]](https://dom.spec.whatwg.org/#biblio-device-orientation) |
| "deviceorientationevent" | DeviceOrientationEvent |
| "dragevent" | [DragEvent](https://html.spec.whatwg.org/multipage/interaction.html#dragevent) | [[HTML]](https://dom.spec.whatwg.org/#biblio-html) |
| "errorevent" | [ErrorEvent](https://html.spec.whatwg.org/multipage/webappapis.html#errorevent) |
| "event" | [Event](https://dom.spec.whatwg.org/#event) |  |
| "events" |
| "focusevent" | FocusEvent | [[UIEVENTS]](https://dom.spec.whatwg.org/#biblio-uievents) |
| "hashchangeevent" | [HashChangeEvent](https://html.spec.whatwg.org/multipage/browsers.html#hashchangeevent) | [[HTML]](https://dom.spec.whatwg.org/#biblio-html) |
| "htmlevents" | [Event](https://dom.spec.whatwg.org/#event) |  |
| "idbversionchangeevent" | IDBVersionChangeEvent | [[INDEXEDDB]](https://dom.spec.whatwg.org/#biblio-indexeddb) |
| "keyboardevent" | [KeyboardEvent](https://w3c.github.io/uievents/#interface-keyboardevent) | [[UIEVENTS]](https://dom.spec.whatwg.org/#biblio-uievents) |
| "messageevent" | [MessageEvent](https://html.spec.whatwg.org/multipage/comms.html#messageevent) | [[HTML]](https://dom.spec.whatwg.org/#biblio-html) |
| "mouseevent" | [MouseEvent](https://w3c.github.io/uievents/#interface-mouseevent) | [[UIEVENTS]](https://dom.spec.whatwg.org/#biblio-uievents) |
| "mouseevents" |
| "pagetransitionevent" | [PageTransitionEvent](https://html.spec.whatwg.org/multipage/browsers.html#pagetransitionevent) | [[HTML]](https://dom.spec.whatwg.org/#biblio-html) |
| "popstateevent" | [PopStateEvent](https://html.spec.whatwg.org/multipage/browsers.html#popstateevent) |
| "progressevent" | ProgressEvent | [[XHR]](https://dom.spec.whatwg.org/#biblio-xhr) |
| "storageevent" | [StorageEvent](https://html.spec.whatwg.org/multipage/webstorage.html#storageevent) | [[HTML]](https://dom.spec.whatwg.org/#biblio-html) |
| "svgevents" | [Event](https://dom.spec.whatwg.org/#event) |  |
| "svgzoomevent" | [SVGZoomEvent](https://www.w3.org/TR/SVG/script.html#InterfaceSVGZoomEvent) | [[SVG]](https://dom.spec.whatwg.org/#biblio-svg) |
| "svgzoomevents" |
| "textevent" | CompositionEvent | [[UIEVENTS]](https://dom.spec.whatwg.org/#biblio-uievents) |
| "touchevent" | [TouchEvent](https://www.w3.org/TR/touch-events/#touchevent-interface) | [[TOUCH-EVENTS]](https://dom.spec.whatwg.org/#biblio-touch-events) |
| "trackevent" | [TrackEvent](https://html.spec.whatwg.org/multipage/embedded-content.html#trackevent) | [[HTML]](https://dom.spec.whatwg.org/#biblio-html) |
| "transitionevent" | [TransitionEvent](https://drafts.csswg.org/css-transitions-1/#Events-TransitionEvent) | [[CSS3-TRANSITIONS]](https://dom.spec.whatwg.org/#biblio-css3-transitions) |
| "uievent" | [UIEvent](https://w3c.github.io/uievents/#interface-uievent) | [[UIEVENTS]](https://dom.spec.whatwg.org/#biblio-uievents) |
| "uievents" |
| "webglcontextevent" | WebGLContextEvent | [[WEBGL]](https://dom.spec.whatwg.org/#biblio-webgl) |
| "wheelevent" | WheelEvent | [[UIEVENTS]](https://dom.spec.whatwg.org/#biblio-uievents) |

1. If *constructor* is null, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotSupportedError](https://heycam.github.io/webidl/#notsupportederror).
2. If the initial value of *constructor* is undefined, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotSupportedError](https://heycam.github.io/webidl/#notsupportederror).

Typically user agents disable support for touch events in some configurations, in which case the initial value of [TouchEvent](https://www.w3.org/TR/touch-events/#touchevent-interface) is undefined.

1. Let *event* be the result of [invoking](https://dom.spec.whatwg.org/#concept-event-constructor) the initial value of *constructor* with the empty string as argument.
2. Unset *event*’s [initialized flag](https://dom.spec.whatwg.org/#initialized-flag).
3. Return *event*.

[Event](https://dom.spec.whatwg.org/#concept-event) constructors ought to be used instead.

The *createRange()* method, when invoked, must return a new [range](https://dom.spec.whatwg.org/#concept-range) with ([context object](https://dom.spec.whatwg.org/#context-object), 0) as its [start](https://dom.spec.whatwg.org/#concept-range-start) and [end](https://dom.spec.whatwg.org/#concept-range-end).

The [Range()](https://dom.spec.whatwg.org/#dom-range-range) constructor ought to be used instead.

The *createNodeIterator(root, whatToShow, filter)* method, when invoked, must run these steps:

1. Create a [NodeIterator](https://dom.spec.whatwg.org/#nodeiterator) object.
2. Set [root](https://dom.spec.whatwg.org/#concept-traversal-root) to *root* and initialize the [referenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-referencenode) attribute to *root*.
3. Initialize the [pointerBeforeReferenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-pointerbeforereferencenode) attribute to true.
4. Set [whatToShow](https://dom.spec.whatwg.org/#concept-traversal-whattoshow) to *whatToShow*.
5. Set [filter](https://dom.spec.whatwg.org/#concept-traversal-filter) to *filter*.
6. Return the newly created [NodeIterator](https://dom.spec.whatwg.org/#nodeiterator) object.

The *createTreeWalker(root, whatToShow, filter)* method, when invoked, must run these steps:

1. Create a [TreeWalker](https://dom.spec.whatwg.org/#treewalker) object.
2. Set [root](https://dom.spec.whatwg.org/#concept-traversal-root) to *root* and initialize the [currentNode](https://dom.spec.whatwg.org/#dom-treewalker-currentnode) attribute to *root*.
3. Set [whatToShow](https://dom.spec.whatwg.org/#concept-traversal-whattoshow) to *whatToShow*.
4. Set [filter](https://dom.spec.whatwg.org/#concept-traversal-filter) to *filter*.
5. Return the newly created [TreeWalker](https://dom.spec.whatwg.org/#treewalker) object.

**4.5.1. Interface** [**DOMImplementation**](https://dom.spec.whatwg.org/#domimplementation)

User agents must create a [DOMImplementation](https://dom.spec.whatwg.org/#domimplementation) object whenever a [document](https://dom.spec.whatwg.org/#concept-document) is created and associate it with that [document](https://dom.spec.whatwg.org/#concept-document).

[Exposed=Window]

interface *DOMImplementation* {

[NewObject] [DocumentType](https://dom.spec.whatwg.org/#documenttype) [createDocumentType](https://dom.spec.whatwg.org/#dom-domimplementation-createdocumenttype)(DOMString *qualifiedName*, DOMString *publicId*, DOMString *systemId*);

[NewObject] [XMLDocument](https://dom.spec.whatwg.org/#xmldocument) [createDocument](https://dom.spec.whatwg.org/#dom-domimplementation-createdocument)(DOMString? *namespace*, [TreatNullAs=EmptyString] DOMString *qualifiedName*, optional [DocumentType](https://dom.spec.whatwg.org/#documenttype)? *doctype* = null);

[NewObject] [Document](https://dom.spec.whatwg.org/#document) [createHTMLDocument](https://dom.spec.whatwg.org/#dom-domimplementation-createhtmldocument)(optional DOMString *title*);

boolean [hasFeature](https://dom.spec.whatwg.org/#dom-domimplementation-hasfeature)(); // useless; always returns true

};

*doctype* = *document* . [implementation](https://dom.spec.whatwg.org/#dom-document-implementation) . [createDocumentType(qualifiedName, publicId, systemId)](https://dom.spec.whatwg.org/#dom-domimplementation-createdocumenttype)

Returns a [doctype](https://dom.spec.whatwg.org/#concept-doctype), with the given *qualifiedName*, *publicId*, and *systemId*. If *qualifiedName* does not match the [Name](https://www.w3.org/TR/xml/#NT-Name) production, an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror) is thrown, and if it does not match the [QName](https://www.w3.org/TR/xml-names/#NT-QName) production, a [NamespaceError](https://heycam.github.io/webidl/#namespaceerror) is thrown.

*doc* = *document* . [implementation](https://dom.spec.whatwg.org/#dom-document-implementation) . [createDocument(*namespace*, *qualifiedName* [, *doctype* = null])](https://dom.spec.whatwg.org/#dom-domimplementation-createdocument)

Returns an [XMLDocument](https://dom.spec.whatwg.org/#xmldocument), with a [document element](https://dom.spec.whatwg.org/#document-element) whose [local name](https://dom.spec.whatwg.org/#concept-element-local-name) is *qualifiedName* and whose [namespace](https://dom.spec.whatwg.org/#concept-element-namespace) is *namespace* (unless *qualifiedName* is the empty string), and with *doctype*, if it is given, as its [doctype](https://dom.spec.whatwg.org/#concept-doctype).

This method throws the same exceptions as the [createElementNS()](https://dom.spec.whatwg.org/#dom-document-createelementns) method, when invoked with *namespace* and *qualifiedName*.

*doc* = *document* . [implementation](https://dom.spec.whatwg.org/#dom-document-implementation) . [createHTMLDocument([*title*])](https://dom.spec.whatwg.org/#dom-domimplementation-createhtmldocument)

Returns a [document](https://dom.spec.whatwg.org/#concept-document), with a basic [tree](https://dom.spec.whatwg.org/#concept-tree) already constructed including a [title](https://html.spec.whatwg.org/multipage/semantics.html#the-title-element) element, unless the *title* argument is omitted.

The *createDocumentType(qualifiedName, publicId, systemId)* method, when invoked, must run these steps:

1. [Validate](https://dom.spec.whatwg.org/#validate) *qualifiedName*. Rethrow any exceptions.
2. Return a new [doctype](https://dom.spec.whatwg.org/#concept-doctype), with *qualifiedName* as its [name](https://dom.spec.whatwg.org/#concept-doctype-name), *publicId* as its [public ID](https://dom.spec.whatwg.org/#concept-doctype-publicid), and *systemId* as its [system ID](https://dom.spec.whatwg.org/#concept-doctype-systemid), and with its [node document](https://dom.spec.whatwg.org/#concept-node-document) set to the associated [document](https://dom.spec.whatwg.org/#concept-document) of the [context object](https://dom.spec.whatwg.org/#context-object).

No check is performed that *publicId* code points match the [PubidChar](https://www.w3.org/TR/xml/#NT-PubidChar) production or that *systemId* does not contain both a '"' and a "'".

The *createDocument(namespace, qualifiedName, doctype)* method, when invoked, must run these steps:

1. Let *document* be a new [XMLDocument](https://dom.spec.whatwg.org/#xmldocument).

This method creates an [XMLDocument](https://dom.spec.whatwg.org/#xmldocument) rather than a normal [document](https://dom.spec.whatwg.org/#concept-document). They are identical except for the addition of the [load()](https://html.spec.whatwg.org/multipage/dom.html#dom-xmldocument-load) method deployed content relies upon. [[HTML]](https://dom.spec.whatwg.org/#biblio-html)

1. Let *element* be null.
2. If *qualifiedName* is not the empty string, then set *element* to the result of running the [internal createElementNS steps](https://dom.spec.whatwg.org/#internal-createelementns-steps), given *document*, *namespace*, *qualifiedName*, and an empty dictionary. Rethrow any exceptions.
3. If *doctype* is non-null, [append](https://dom.spec.whatwg.org/#concept-node-append) *doctype* to *document*.
4. If *element* is non-null, [append](https://dom.spec.whatwg.org/#concept-node-append) *element* to *document*.
5. *document*’s [origin](https://html.spec.whatwg.org/multipage/browsers.html#concept-origin) is the [origin](https://html.spec.whatwg.org/multipage/browsers.html#concept-origin) of the [context object](https://dom.spec.whatwg.org/#context-object)’s associated [document](https://dom.spec.whatwg.org/#concept-document). [[HTML]](https://dom.spec.whatwg.org/#biblio-html)
6. *document*’s [content type](https://dom.spec.whatwg.org/#concept-document-content-type) is determined by *namespace*:

[HTML namespace](https://dom.spec.whatwg.org/#html-namespace)

application/xhtml+xml

[SVG namespace](https://dom.spec.whatwg.org/#svg-namespace)

image/svg+xml

Any other namespace

application/xml

1. Return *document*.

The *createHTMLDocument(title)* method, when invoked, must run these steps:

1. Let *doc* be a new [document](https://dom.spec.whatwg.org/#concept-document) that is an [HTML document](https://dom.spec.whatwg.org/#html-document).
2. Set *doc*’s [content type](https://dom.spec.whatwg.org/#concept-document-content-type) to "text/html".
3. [Append](https://dom.spec.whatwg.org/#concept-node-append) a new [doctype](https://dom.spec.whatwg.org/#concept-doctype), with "html" as its [name](https://dom.spec.whatwg.org/#concept-doctype-name) and with its [node document](https://dom.spec.whatwg.org/#concept-node-document) set to *doc*, to *doc*.
4. [Append](https://dom.spec.whatwg.org/#concept-node-append) the result of [creating an element](https://dom.spec.whatwg.org/#concept-create-element) given *doc*, [html](https://html.spec.whatwg.org/multipage/semantics.html#the-html-element), and the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace), to *doc*.
5. [Append](https://dom.spec.whatwg.org/#concept-node-append) the result of [creating an element](https://dom.spec.whatwg.org/#concept-create-element) given *doc*, [head](https://html.spec.whatwg.org/multipage/semantics.html#the-head-element), and the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace), to the [html](https://html.spec.whatwg.org/multipage/semantics.html#the-html-element) element created earlier.
6. If *title* is given:
   1. [Append](https://dom.spec.whatwg.org/#concept-node-append) the result of [creating an element](https://dom.spec.whatwg.org/#concept-create-element) given *doc*, [title](https://html.spec.whatwg.org/multipage/semantics.html#the-title-element), and the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace), to the [head](https://html.spec.whatwg.org/multipage/semantics.html#the-head-element) element created earlier.
   2. [Append](https://dom.spec.whatwg.org/#concept-node-append) a new [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node), with its [data](https://dom.spec.whatwg.org/#concept-cd-data) set to *title* (which could be the empty string) and its [node document](https://dom.spec.whatwg.org/#concept-node-document) set to *doc*, to the [title](https://html.spec.whatwg.org/multipage/semantics.html#the-title-element) element created earlier.
7. [Append](https://dom.spec.whatwg.org/#concept-node-append) the result of [creating an element](https://dom.spec.whatwg.org/#concept-create-element) given *doc*, [body](https://html.spec.whatwg.org/multipage/semantics.html#the-body-element), and the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace), to the [html](https://html.spec.whatwg.org/multipage/semantics.html#the-html-element) element created earlier.
8. *doc*’s [origin](https://html.spec.whatwg.org/multipage/browsers.html#concept-origin) is the [origin](https://html.spec.whatwg.org/multipage/browsers.html#concept-origin) of the [context object](https://dom.spec.whatwg.org/#context-object)’s associated [document](https://dom.spec.whatwg.org/#concept-document). [[HTML]](https://dom.spec.whatwg.org/#biblio-html)
9. Return *doc*.

The *hasFeature()* method, when invoked, must return true.

[hasFeature()](https://dom.spec.whatwg.org/#dom-domimplementation-hasfeature) originally would report whether the user agent claimed to support a given DOM feature, but experience proved it was not nearly as reliable or granular as simply checking whether the desired objects, attributes, or methods existed. As such, it is no longer to be used, but continues to exist (and simply returns true) so that old pages don’t stop working.

**4.6. Interface** [**DocumentType**](https://dom.spec.whatwg.org/#documenttype)

[Exposed=Window]

interface *DocumentType* : [Node](https://dom.spec.whatwg.org/#node) {

readonly attribute DOMString [name](https://dom.spec.whatwg.org/#dom-documenttype-name);

readonly attribute DOMString [publicId](https://dom.spec.whatwg.org/#dom-documenttype-publicid);

readonly attribute DOMString [systemId](https://dom.spec.whatwg.org/#dom-documenttype-systemid);

};

[DocumentType](https://dom.spec.whatwg.org/#documenttype) [nodes](https://dom.spec.whatwg.org/#concept-node) are simply known as *doctypes*.

[Doctypes](https://dom.spec.whatwg.org/#concept-doctype) have an associated *name*, *public ID*, and *system ID*.

When a [doctype](https://dom.spec.whatwg.org/#concept-doctype) is created, its [name](https://dom.spec.whatwg.org/#concept-doctype-name) is always given. Unless explicitly given when a [doctype](https://dom.spec.whatwg.org/#concept-doctype) is created, its [public ID](https://dom.spec.whatwg.org/#concept-doctype-publicid) and [system ID](https://dom.spec.whatwg.org/#concept-doctype-systemid) are the empty string.

The *name* attribute’s getter must return the [context object](https://dom.spec.whatwg.org/#context-object)’s [name](https://dom.spec.whatwg.org/#concept-doctype-name).

The *publicId* attribute’s getter must return the [context object](https://dom.spec.whatwg.org/#context-object)’s [public ID](https://dom.spec.whatwg.org/#concept-doctype-publicid).

The *systemId* attribute’s getter must return the [context object](https://dom.spec.whatwg.org/#context-object)’s [system ID](https://dom.spec.whatwg.org/#concept-doctype-systemid).

**4.7. Interface** [**DocumentFragment**](https://dom.spec.whatwg.org/#documentfragment)

[[Constructor](https://dom.spec.whatwg.org/#dom-documentfragment-documentfragment),

Exposed=Window]

interface *DocumentFragment* : [Node](https://dom.spec.whatwg.org/#node) {

};

A [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [node](https://dom.spec.whatwg.org/#concept-node) has an associated *host* (null or an [element](https://dom.spec.whatwg.org/#concept-element) in a different [node tree](https://dom.spec.whatwg.org/#concept-node-tree)). It is null unless otherwise stated.

An object *A* is a *host-including inclusive ancestor* of an object *B*, if either *A* is an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of *B*, or if *B*’s [root](https://dom.spec.whatwg.org/#concept-tree-root) has a non-null [host](https://dom.spec.whatwg.org/#concept-documentfragment-host) and *A* is a [host-including inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-host-including-inclusive-ancestor) of *B*’s [root](https://dom.spec.whatwg.org/#concept-tree-root)’s [host](https://dom.spec.whatwg.org/#concept-documentfragment-host).

The [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [node](https://dom.spec.whatwg.org/#concept-node)’s [host](https://dom.spec.whatwg.org/#concept-documentfragment-host) concept is useful for HTML’s [template](https://html.spec.whatwg.org/multipage/scripting.html#the-template-element) element and for [shadow roots](https://dom.spec.whatwg.org/#concept-shadow-root), and impacts the [pre-insert](https://dom.spec.whatwg.org/#concept-node-pre-insert) and [replace](https://dom.spec.whatwg.org/#concept-node-replace) algorithms.

*tree* = new [DocumentFragment()](https://dom.spec.whatwg.org/#dom-documentfragment-documentfragment)

Returns a new [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [node](https://dom.spec.whatwg.org/#concept-node).

The *DocumentFragment()* constructor, when invoked, must return a new [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [node](https://dom.spec.whatwg.org/#concept-node) whose [node document](https://dom.spec.whatwg.org/#concept-node-document) is the global object’s associated [document](https://dom.spec.whatwg.org/#concept-document).

**4.8. Interface** [**ShadowRoot**](https://dom.spec.whatwg.org/#shadowroot)

[Exposed=Window]

interface *ShadowRoot* : [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) {

readonly attribute [ShadowRootMode](https://dom.spec.whatwg.org/#enumdef-shadowrootmode) [mode](https://dom.spec.whatwg.org/#dom-shadowroot-mode);

readonly attribute [Element](https://dom.spec.whatwg.org/#element) [host](https://dom.spec.whatwg.org/#dom-shadowroot-host);

};

enum *ShadowRootMode* { *"open"*, *"closed"* };

[ShadowRoot](https://dom.spec.whatwg.org/#shadowroot) [nodes](https://dom.spec.whatwg.org/#concept-node) are simply known as *shadow roots*.

[Shadow roots](https://dom.spec.whatwg.org/#concept-shadow-root) have an associated *mode* ("open" or "closed").

[Shadow roots](https://dom.spec.whatwg.org/#concept-shadow-root)’s associated [host](https://dom.spec.whatwg.org/#concept-documentfragment-host) is never null.

The *mode* attribute’s getter must return the [context object](https://dom.spec.whatwg.org/#context-object)’s [mode](https://dom.spec.whatwg.org/#shadowroot-mode).

The *host* attribute’s getter must return the [context object](https://dom.spec.whatwg.org/#context-object)’s [host](https://dom.spec.whatwg.org/#concept-documentfragment-host).

In *shadow-including tree order*, is [shadow-including preorder, depth-first traversal](https://dom.spec.whatwg.org/#shadow-including-preorder-depth-first-traversal) of a [node tree](https://dom.spec.whatwg.org/#concept-node-tree). *shadow-including preorder, depth-first traversal* of a [node tree](https://dom.spec.whatwg.org/#concept-node-tree) *tree* is preorder, depth-first traversal of *tree*, with for each [shadow host](https://dom.spec.whatwg.org/#element-shadow-host) encountered in *tree*, [shadow-including preorder, depth-first traversal](https://dom.spec.whatwg.org/#shadow-including-preorder-depth-first-traversal) of that [element](https://dom.spec.whatwg.org/#concept-element)’s [shadow root](https://dom.spec.whatwg.org/#concept-element-shadow-root)’s [node tree](https://dom.spec.whatwg.org/#concept-node-tree) just after it is encountered.

The *shadow-including root* of an object is its [root](https://dom.spec.whatwg.org/#concept-tree-root)’s [host](https://dom.spec.whatwg.org/#concept-documentfragment-host)’s [shadow-including root](https://dom.spec.whatwg.org/#concept-shadow-including-root), if the object’s [root](https://dom.spec.whatwg.org/#concept-tree-root) is a [shadow root](https://dom.spec.whatwg.org/#concept-shadow-root), and its [root](https://dom.spec.whatwg.org/#concept-tree-root) otherwise.

An object *A* is a *shadow-including descendant* of an object *B*, if *A* is a [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant) of *B*, or *A*’s [root](https://dom.spec.whatwg.org/#concept-tree-root) is a [shadow root](https://dom.spec.whatwg.org/#concept-shadow-root) and *A*’s [root](https://dom.spec.whatwg.org/#concept-tree-root)’s [host](https://dom.spec.whatwg.org/#concept-documentfragment-host) is a [shadow-including inclusive descendant](https://dom.spec.whatwg.org/#concept-shadow-including-inclusive-descendant) of *B*.

A *shadow-including inclusive descendant* is an object or one of its [shadow-including descendants](https://dom.spec.whatwg.org/#concept-shadow-including-descendant).

An object *A* is a *shadow-including ancestor* of an object *B*, if and only if *B* is a [shadow-including descendant](https://dom.spec.whatwg.org/#concept-shadow-including-descendant) of *A*.

A *shadow-including inclusive ancestor* is an object or one of its [shadow-including ancestors](https://dom.spec.whatwg.org/#concept-shadow-including-ancestor).

For now you can find more information about this object in [Shadow DOM](https://w3c.github.io/webcomponents/spec/shadow/). The DOM Standard will be updated over time to cover more details.

**4.9. Interface** [**Element**](https://dom.spec.whatwg.org/#element)

[Exposed=Window]

interface *Element* : [Node](https://dom.spec.whatwg.org/#node) {

readonly attribute DOMString? [namespaceURI](https://dom.spec.whatwg.org/#dom-element-namespaceuri);

readonly attribute DOMString? [prefix](https://dom.spec.whatwg.org/#dom-element-prefix);

readonly attribute DOMString [localName](https://dom.spec.whatwg.org/#dom-element-localname);

readonly attribute DOMString [tagName](https://dom.spec.whatwg.org/#dom-element-tagname);

[CEReactions] attribute DOMString [id](https://dom.spec.whatwg.org/#dom-element-id);

[CEReactions] attribute DOMString [className](https://dom.spec.whatwg.org/#dom-element-classname);

[CEReactions, SameObject, PutForwards=[value](https://dom.spec.whatwg.org/#dom-domtokenlist-value)] readonly attribute [DOMTokenList](https://dom.spec.whatwg.org/#domtokenlist) [classList](https://dom.spec.whatwg.org/#dom-element-classlist);

[CEReactions] attribute DOMString [slot](https://dom.spec.whatwg.org/#dom-element-slot);

boolean [hasAttributes](https://dom.spec.whatwg.org/#dom-element-hasattributes)();

[SameObject] readonly attribute [NamedNodeMap](https://dom.spec.whatwg.org/#namednodemap) [attributes](https://dom.spec.whatwg.org/#dom-element-attributes);

sequence<DOMString> [getAttributeNames](https://dom.spec.whatwg.org/#dom-element-getattributenames)();

DOMString? [getAttribute](https://dom.spec.whatwg.org/#dom-element-getattribute)(DOMString *qualifiedName*);

DOMString? [getAttributeNS](https://dom.spec.whatwg.org/#dom-element-getattributens)(DOMString? *namespace*, DOMString *localName*);

[CEReactions] void [setAttribute](https://dom.spec.whatwg.org/#dom-element-setattribute)(DOMString *qualifiedName*, DOMString *value*);

[CEReactions] void [setAttributeNS](https://dom.spec.whatwg.org/#dom-element-setattributens)(DOMString? *namespace*, DOMString *qualifiedName*, DOMString *value*);

[CEReactions] void [removeAttribute](https://dom.spec.whatwg.org/#dom-element-removeattribute)(DOMString *qualifiedName*);

[CEReactions] void [removeAttributeNS](https://dom.spec.whatwg.org/#dom-element-removeattributens)(DOMString? *namespace*, DOMString *localName*);

boolean [hasAttribute](https://dom.spec.whatwg.org/#dom-element-hasattribute)(DOMString *qualifiedName*);

boolean [hasAttributeNS](https://dom.spec.whatwg.org/#dom-element-hasattributens)(DOMString? *namespace*, DOMString *localName*);

[Attr](https://dom.spec.whatwg.org/#attr)? [getAttributeNode](https://dom.spec.whatwg.org/#dom-element-getattributenode)(DOMString *qualifiedName*);

[Attr](https://dom.spec.whatwg.org/#attr)? [getAttributeNodeNS](https://dom.spec.whatwg.org/#dom-element-getattributenodens)(DOMString? *namespace*, DOMString *localName*);

[CEReactions] [Attr](https://dom.spec.whatwg.org/#attr)? [setAttributeNode](https://dom.spec.whatwg.org/#dom-element-setattributenode)([Attr](https://dom.spec.whatwg.org/#attr) *attr*);

[CEReactions] [Attr](https://dom.spec.whatwg.org/#attr)? [setAttributeNodeNS](https://dom.spec.whatwg.org/#dom-element-setattributenodens)([Attr](https://dom.spec.whatwg.org/#attr) *attr*);

[CEReactions] [Attr](https://dom.spec.whatwg.org/#attr) [removeAttributeNode](https://dom.spec.whatwg.org/#dom-element-removeattributenode)([Attr](https://dom.spec.whatwg.org/#attr) *attr*);

[ShadowRoot](https://dom.spec.whatwg.org/#shadowroot) [attachShadow](https://dom.spec.whatwg.org/#dom-element-attachshadow)([ShadowRootInit](https://dom.spec.whatwg.org/#dictdef-shadowrootinit) *init*);

readonly attribute [ShadowRoot](https://dom.spec.whatwg.org/#shadowroot)? [shadowRoot](https://dom.spec.whatwg.org/#dom-element-shadowroot);

[Element](https://dom.spec.whatwg.org/#element)? [closest](https://dom.spec.whatwg.org/#dom-element-closest)(DOMString *selectors*);

boolean [matches](https://dom.spec.whatwg.org/#dom-element-matches)(DOMString *selectors*);

boolean [webkitMatchesSelector](https://dom.spec.whatwg.org/#dom-element-webkitmatchesselector)(DOMString *selectors*); // historical alias of .matches

[HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) [getElementsByTagName](https://dom.spec.whatwg.org/#dom-element-getelementsbytagname)(DOMString *qualifiedName*);

[HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) [getElementsByTagNameNS](https://dom.spec.whatwg.org/#dom-element-getelementsbytagnamens)(DOMString? *namespace*, DOMString *localName*);

[HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) [getElementsByClassName](https://dom.spec.whatwg.org/#dom-element-getelementsbyclassname)(DOMString *classNames*);

[CEReactions] [Element](https://dom.spec.whatwg.org/#element)? [insertAdjacentElement](https://dom.spec.whatwg.org/#dom-element-insertadjacentelement)(DOMString *where*, [Element](https://dom.spec.whatwg.org/#element) *element*); // historical

[CEReactions] void [insertAdjacentText](https://dom.spec.whatwg.org/#dom-element-insertadjacenttext)(DOMString *where*, DOMString *data*); // historical

};

dictionary *ShadowRootInit* {

required [ShadowRootMode](https://dom.spec.whatwg.org/#enumdef-shadowrootmode) *mode*;

};

[Element](https://dom.spec.whatwg.org/#element) [nodes](https://dom.spec.whatwg.org/#concept-node) are simply known as *elements*.

[Elements](https://dom.spec.whatwg.org/#concept-element) have an associated *namespace*, *namespace prefix*, *local name*, and *custom element state*. When an [element](https://dom.spec.whatwg.org/#concept-element) is [created](https://dom.spec.whatwg.org/#concept-create-element), all of these values are initialized.

An [element](https://dom.spec.whatwg.org/#concept-element)’s [custom element state](https://dom.spec.whatwg.org/#concept-element-custom-element-state) is one of "undefined", "uncustomized", or "custom". An [element](https://dom.spec.whatwg.org/#concept-element) whose [custom element state](https://dom.spec.whatwg.org/#concept-element-custom-element-state) is "uncustomized" or "custom" is said to be *defined*. An [element](https://dom.spec.whatwg.org/#concept-element) whose [custom element state](https://dom.spec.whatwg.org/#concept-element-custom-element-state) is "custom", is said to be *custom*.

Whether or not an element is [defined](https://dom.spec.whatwg.org/#concept-element-defined) is used to determine the behavior of the [:defined](https://html.spec.whatwg.org/multipage/scripting.html#selector-defined) pseudo-class. Whether or not an element is [custom](https://dom.spec.whatwg.org/#concept-element-custom) is used to determine the behavior of the [mutation algorithms](https://dom.spec.whatwg.org/#mutation-algorithms).

The following code illustrates elements in each of these three states:

<!DOCTYPE html>

<script>

window.customElements.define("sw-rey", class extends HTMLElement {})

window.customElements.define("sw-finn", class extends HTMLElement {}, { extends: "p" })

window.customElements.define("sw-kylo", class extends HTMLElement {

constructor() {

super()

throw new Error("The droid... stole a freighter?")

}

})

</script>

<!-- "undefined" (not defined, not custom) -->

<sw-han></sw-han>

<sw-kylo></sw-kylo>

<p is="sw-luke"></p>

<p is="asdf"></p>

<!-- "uncustomized" (defined, not custom) -->

<p></p>

<asdf></asdf>

<!-- "custom" (defined, custom) -->

<sw-rey></sw-rey>

<p is="sw-finn"></p>

[Elements](https://dom.spec.whatwg.org/#concept-element) also have an associated *shadow root* (null or a [shadow root](https://dom.spec.whatwg.org/#concept-shadow-root)). It is null unless otherwise stated. An [element](https://dom.spec.whatwg.org/#concept-element) is a *shadow host* if its [shadow root](https://dom.spec.whatwg.org/#concept-element-shadow-root) is non-null.

An [element](https://dom.spec.whatwg.org/#concept-element)’s *qualified name* is its [local name](https://dom.spec.whatwg.org/#concept-element-local-name) if its [namespace prefix](https://dom.spec.whatwg.org/#concept-element-namespace-prefix) is null, and its [namespace prefix](https://dom.spec.whatwg.org/#concept-element-namespace-prefix), followed by ":", followed by its [local name](https://dom.spec.whatwg.org/#concept-element-local-name), otherwise.

User agents could have this as an internal slot as an optimization, but are not required to do so. The standard has this concept for readability.

To *create an element*, given a *document*, *localName*, *namespace*, and optional *prefix*, *is*, and *synchronous custom elements flag*, run these steps:

1. If *prefix* was not given, let *prefix* be null.
2. If *is* was not given, let *is* be null.
3. Let *result* be null.
4. Let *definition* be the result of [looking up a custom element definition](https://html.spec.whatwg.org/multipage/scripting.html#look-up-a-custom-element-definition) given *document*, *namespace*, *localName*, and *is*.
5. If *definition* is non-null, and *definition*’s [name](https://html.spec.whatwg.org/multipage/scripting.html#concept-custom-element-definition-name) is not equal to its [local name](https://html.spec.whatwg.org/multipage/scripting.html#concept-custom-element-definition-local-name) (i.e., *definition* represents a [customized built-in element](https://html.spec.whatwg.org/multipage/scripting.html#customized-built-in-element)), then:
   1. Let *interface* be the [element interface](https://dom.spec.whatwg.org/#concept-element-interface) for *localName* and the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace).
   2. Set *result* to a new [element](https://dom.spec.whatwg.org/#concept-element) that implements *interface*, with no attributes, [namespace](https://dom.spec.whatwg.org/#concept-element-namespace) set to the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace), [namespace prefix](https://dom.spec.whatwg.org/#concept-element-namespace-prefix) set to *prefix*, [local name](https://dom.spec.whatwg.org/#concept-element-local-name) set to *localName*, [custom element state](https://dom.spec.whatwg.org/#concept-element-custom-element-state) set to "undefined", and [node document](https://dom.spec.whatwg.org/#concept-node-document) set to *document*.
   3. If the *synchronous custom elements flag* is set, [upgrade](https://html.spec.whatwg.org/multipage/scripting.html#concept-upgrade-an-element) *element* using *definition*.
   4. Otherwise, [enqueue a custom element upgrade reaction](https://html.spec.whatwg.org/multipage/scripting.html#enqueue-a-custom-element-upgrade-reaction) given *result* and *definition*.
6. Otherwise, if *definition* is non-null, then:
   1. If the *synchronous custom elements flag* is set:
      1. Let *C* be *definition*’s [constructor](https://html.spec.whatwg.org/multipage/scripting.html#concept-custom-element-definition-constructor).
      2. Set *result* to [Construct](https://tc39.github.io/ecma262/#sec-construct)(*C*). Rethrow any exceptions.
      3. If *result* does not implement the [HTMLElement](https://html.spec.whatwg.org/multipage/dom.html#htmlelement) interface, [throw](https://heycam.github.io/webidl/#dfn-throw) a TypeError.

This is meant to be a brand check to ensure that the object was allocated by the [HTMLElement](https://html.spec.whatwg.org/multipage/dom.html#htmlelement) constructor. See [webidl #97](https://github.com/heycam/webidl/issues/97) about making this more precise.

* + 1. If *result*’s [attribute list](https://dom.spec.whatwg.org/#concept-element-attribute) is not empty, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotSupportedError](https://heycam.github.io/webidl/#notsupportederror).
    2. If *result* has [children](https://dom.spec.whatwg.org/#concept-tree-child), then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotSupportedError](https://heycam.github.io/webidl/#notsupportederror).
    3. If *result*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is not null, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotSupportedError](https://heycam.github.io/webidl/#notsupportederror).
    4. If *result*’s [node document](https://dom.spec.whatwg.org/#concept-node-document) is not *document*, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotSupportedError](https://heycam.github.io/webidl/#notsupportederror).
    5. If *result*’s [namespace](https://dom.spec.whatwg.org/#concept-element-namespace) is not the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace), then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotSupportedError](https://heycam.github.io/webidl/#notsupportederror).
    6. If *result*’s [local name](https://dom.spec.whatwg.org/#concept-element-local-name) is not equal to *localName*, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotSupportedError](https://heycam.github.io/webidl/#notsupportederror).
    7. Set *result*’s [namespace prefix](https://dom.spec.whatwg.org/#concept-element-namespace-prefix) to *prefix*.
    8. Set *result*’s [custom element state](https://dom.spec.whatwg.org/#concept-element-custom-element-state) to "custom".
  1. Otherwise:
     1. Set *result* to a new [element](https://dom.spec.whatwg.org/#concept-element) that implements the [HTMLElement](https://html.spec.whatwg.org/multipage/dom.html#htmlelement) interface, with no attributes, [namespace](https://dom.spec.whatwg.org/#concept-element-namespace) set to the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace), [namespace prefix](https://dom.spec.whatwg.org/#concept-element-namespace-prefix) set to *prefix*, [local name](https://dom.spec.whatwg.org/#concept-element-local-name) set to *localName*, [custom element state](https://dom.spec.whatwg.org/#concept-element-custom-element-state) set to "undefined", and [node document](https://dom.spec.whatwg.org/#concept-node-document) set to *document*.
     2. [Enqueue a custom element upgrade reaction](https://html.spec.whatwg.org/multipage/scripting.html#enqueue-a-custom-element-upgrade-reaction) given *result* and *definition*.

1. Otherwise:
   1. Let *interface* be the [element interface](https://dom.spec.whatwg.org/#concept-element-interface) for *localName* and *namespace*.
   2. Set *result* to a new [element](https://dom.spec.whatwg.org/#concept-element) that implements *interface*, with no attributes, [namespace](https://dom.spec.whatwg.org/#concept-element-namespace) set to *namespace*, [namespace prefix](https://dom.spec.whatwg.org/#concept-element-namespace-prefix) set to *prefix*, [local name](https://dom.spec.whatwg.org/#concept-element-local-name) set to *localName*, [custom element state](https://dom.spec.whatwg.org/#concept-element-custom-element-state) set to "uncustomized", and [node document](https://dom.spec.whatwg.org/#concept-node-document) set to *document*.
   3. If *document* has a [browsing context](https://html.spec.whatwg.org/multipage/browsers.html#concept-document-bc), and *namespace* is the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace), and either *localName* is a [valid custom element name](https://html.spec.whatwg.org/multipage/scripting.html#valid-custom-element-name) or *is* is is non-null, set *result*’s [custom element state](https://dom.spec.whatwg.org/#concept-element-custom-element-state) to "undefined".
2. Return *result*.

[Elements](https://dom.spec.whatwg.org/#concept-element) also have an ordered *attribute list* exposed through a [NamedNodeMap](https://dom.spec.whatwg.org/#namednodemap). Unless explicitly given when an [element](https://dom.spec.whatwg.org/#concept-element) is created, its [attribute list](https://dom.spec.whatwg.org/#concept-element-attribute) is empty. An [element](https://dom.spec.whatwg.org/#concept-element) *has* an [attribute](https://dom.spec.whatwg.org/#concept-attribute) *A* if *A* is in its [attribute list](https://dom.spec.whatwg.org/#concept-element-attribute).

This and [other specifications](https://dom.spec.whatwg.org/#other-applicable-specifications) may define *attribute change steps* for [elements](https://dom.spec.whatwg.org/#concept-element). The algorithm is passed *element*, *localName*, *oldValue*, *value*, and *namespace*.

To *change* an [attribute](https://dom.spec.whatwg.org/#concept-attribute) *attribute* from an [element](https://dom.spec.whatwg.org/#concept-element) *element* to *value*, run these steps:

1. [Queue a mutation record](https://dom.spec.whatwg.org/#queue-a-mutation-record) of "attributes" for *element* with name *attribute*’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name), namespace *attribute*’s [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace), and oldValue *attribute*’s [value](https://dom.spec.whatwg.org/#concept-attribute-value).
2. If *element* is [custom](https://dom.spec.whatwg.org/#concept-element-custom), then [enqueue a custom element callback reaction](https://html.spec.whatwg.org/multipage/scripting.html#enqueue-a-custom-element-callback-reaction) with *element*, callback name "attributeChangedCallback", and an argument list containing *attribute*’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name), *attribute*’s [value](https://dom.spec.whatwg.org/#concept-attribute-value), *value*, and *attribute*’s [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace).
3. Run the [attribute change steps](https://dom.spec.whatwg.org/#concept-element-attributes-change-ext) with *element*, *attribute*’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name), *attribute*’s [value](https://dom.spec.whatwg.org/#concept-attribute-value), *value*, and *attribute*’s [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace).
4. Set *attribute*’s [value](https://dom.spec.whatwg.org/#concept-attribute-value) to *value*.

To *append* an [attribute](https://dom.spec.whatwg.org/#concept-attribute) *attribute* to an [element](https://dom.spec.whatwg.org/#concept-element) *element*, run these steps:

1. [Queue a mutation record](https://dom.spec.whatwg.org/#queue-a-mutation-record) of "attributes" for *element* with name *attribute*’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name), namespace *attribute*’s [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace), and oldValue null.
2. If *element* is [custom](https://dom.spec.whatwg.org/#concept-element-custom), then [enqueue a custom element callback reaction](https://html.spec.whatwg.org/multipage/scripting.html#enqueue-a-custom-element-callback-reaction) with *element*, callback name "attributeChangedCallback", and an argument list containing *attribute*’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name), null, *attribute*’s [value](https://dom.spec.whatwg.org/#concept-attribute-value), and *attribute*’s [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace).
3. Run the [attribute change steps](https://dom.spec.whatwg.org/#concept-element-attributes-change-ext) with *element*, *attribute*’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name), null, *attribute*’s [value](https://dom.spec.whatwg.org/#concept-attribute-value), and *attribute*’s [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace).
4. Append the *attribute* to the *element*’s [attribute list](https://dom.spec.whatwg.org/#concept-element-attribute).
5. Set *attribute*’s [element](https://dom.spec.whatwg.org/#concept-attribute-element) to *element*.

To *remove* an [attribute](https://dom.spec.whatwg.org/#concept-attribute) *attribute* from an [element](https://dom.spec.whatwg.org/#concept-element) *element*, run these steps:

1. [Queue a mutation record](https://dom.spec.whatwg.org/#queue-a-mutation-record) of "attributes" for *element* with name *attribute*’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name), namespace *attribute*’s [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace), and oldValue *attribute*’s [value](https://dom.spec.whatwg.org/#concept-attribute-value).
2. If *element* is [custom](https://dom.spec.whatwg.org/#concept-element-custom), then [enqueue a custom element callback reaction](https://html.spec.whatwg.org/multipage/scripting.html#enqueue-a-custom-element-callback-reaction) with *element*, callback name "attributeChangedCallback", and an argument list containing *attribute*’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name), *attribute*’s [value](https://dom.spec.whatwg.org/#concept-attribute-value), null, and *attribute*’s [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace).
3. Run the [attribute change steps](https://dom.spec.whatwg.org/#concept-element-attributes-change-ext) with *element*, *attribute*’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name), *attribute*’s [value](https://dom.spec.whatwg.org/#concept-attribute-value), null, and *attribute*’s [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace).
4. Remove *attribute* from the *element*’s [attribute list](https://dom.spec.whatwg.org/#concept-element-attribute).
5. Set *attribute*’s [element](https://dom.spec.whatwg.org/#concept-attribute-element) to null.

To *replace* an [attribute](https://dom.spec.whatwg.org/#concept-attribute) *oldAttr* by an [attribute](https://dom.spec.whatwg.org/#concept-attribute) *newAttr* in an [element](https://dom.spec.whatwg.org/#concept-element) *element*, run these steps:

1. [Queue a mutation record](https://dom.spec.whatwg.org/#queue-a-mutation-record) of "attributes" for *element* with name *oldAttr*’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name), namespace *oldAttr*’s [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace), and oldValue *oldAttr*’s [value](https://dom.spec.whatwg.org/#concept-attribute-value).
2. If *element* is [custom](https://dom.spec.whatwg.org/#concept-element-custom), then [enqueue a custom element callback reaction](https://html.spec.whatwg.org/multipage/scripting.html#enqueue-a-custom-element-callback-reaction) with *element*, callback name "attributeChangedCallback", and an argument list containing *oldAttr*’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name), *oldAttr*’s [value](https://dom.spec.whatwg.org/#concept-attribute-value), *newAttr*’s [value](https://dom.spec.whatwg.org/#concept-attribute-value), and *oldAttr*’s [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace).
3. Run the [attribute change steps](https://dom.spec.whatwg.org/#concept-element-attributes-change-ext) with *element*, *oldAttr*’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name), *oldAttr*’s [value](https://dom.spec.whatwg.org/#concept-attribute-value), *newAttr*’s [value](https://dom.spec.whatwg.org/#concept-attribute-value), and *oldAttr*’s [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace).
4. Replace *oldAttr* by *newAttr* in the *element*’s [attribute list](https://dom.spec.whatwg.org/#concept-element-attribute).
5. Set *oldAttr*’s [element](https://dom.spec.whatwg.org/#concept-attribute-element) to null.
6. Set *newAttr*’s [element](https://dom.spec.whatwg.org/#concept-attribute-element) to *element*.

To *get an attribute by name* given a *qualifiedName* and [element](https://dom.spec.whatwg.org/#concept-element) *element*, run these steps:

1. If *element* is in the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace) and its [node document](https://dom.spec.whatwg.org/#concept-node-document) is an [HTML document](https://dom.spec.whatwg.org/#html-document), let *qualifiedName* be [converted to ASCII lowercase](https://dom.spec.whatwg.org/#converted-to-ascii-lowercase).
2. Return the first [attribute](https://dom.spec.whatwg.org/#concept-attribute) in *element*’s [attribute list](https://dom.spec.whatwg.org/#concept-element-attribute) whose [qualified name](https://dom.spec.whatwg.org/#concept-attribute-qualified-name) is *qualifiedName*, and null otherwise.

To *get an attribute by namespace and local name* given a *namespace*, *localName*, and [element](https://dom.spec.whatwg.org/#concept-element) *element*, run these steps:

1. If *namespace* is the empty string, set it to null.
2. Return the [attribute](https://dom.spec.whatwg.org/#concept-attribute) in *element*’s [attribute list](https://dom.spec.whatwg.org/#concept-element-attribute) whose [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace) is *namespace* and [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name) is *localName*, if any, and null otherwise.

To *get an attribute value* given an [element](https://dom.spec.whatwg.org/#concept-element) *element*, *localName*, and optionally a *namespace* (null unless stated otherwise), run these steps:

1. Let *attr* be the result of [getting an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-get-by-namespace) given *namespace*, *localName*, and *element*.
2. If *attr* is null, then return the empty string.
3. Return *attr*’s [value](https://dom.spec.whatwg.org/#concept-attribute-value).

To *set an attribute* given an *attr* and *element*, run these steps:

1. If *attr*’s [element](https://dom.spec.whatwg.org/#concept-attribute-element) is neither null nor *element*, [throw](https://heycam.github.io/webidl/#dfn-throw) an [InUseAttributeError](https://heycam.github.io/webidl/#inuseattributeerror).
2. Let *oldAttr* be the result of [getting an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-get-by-namespace) given *attr*’s [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace), *attr*’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name), and *element*.
3. If *oldAttr* is *attr*, return *attr*.
4. If *oldAttr* is non-null, [replace](https://dom.spec.whatwg.org/#concept-element-attributes-replace) it by *attr* in *element*.
5. Otherwise, [append](https://dom.spec.whatwg.org/#concept-element-attributes-append) *attr* to *element*.
6. Return *oldAttr*.

To *set an attribute value* for an [element](https://dom.spec.whatwg.org/#concept-element) *element* using a *localName* and *value*, and an optional *prefix*, and *namespace*, run these steps:

1. If *prefix* is not given, set it to null.
2. If *namespace* is not given, set it to null.
3. Let *attribute* be the result of [getting an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-get-by-namespace) given *namespace*, *localName*, and *element*.
4. If *attribute* is null, create an [attribute](https://dom.spec.whatwg.org/#concept-attribute) whose [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace) is *namespace*, [namespace prefix](https://dom.spec.whatwg.org/#concept-attribute-namespace-prefix) is *prefix*, [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name) is *localName*, and [value](https://dom.spec.whatwg.org/#concept-attribute-value) is *value*, and then [append](https://dom.spec.whatwg.org/#concept-element-attributes-append) this [attribute](https://dom.spec.whatwg.org/#concept-attribute) to *element* and terminate these steps.
5. [Change](https://dom.spec.whatwg.org/#concept-element-attributes-change) *attribute* from *element* to *value*.

To *remove an attribute by name* given a *qualifiedName* and [element](https://dom.spec.whatwg.org/#concept-element) *element*, run these steps:

1. Let *attr* be the result of [getting an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-get-by-name) given *qualifiedName* and *element*.
2. If *attr* is non-null, [remove](https://dom.spec.whatwg.org/#concept-element-attributes-remove) it from *element*.
3. Return *attr*.

To *remove an attribute by namespace and local name* given a *namespace*, *localName*, and [element](https://dom.spec.whatwg.org/#concept-element) *element*, run these steps:

1. Let *attr* be the result of [getting an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-get-by-namespace) given *namespace*, *localName*, and *element*.
2. If *attr* is non-null, [remove](https://dom.spec.whatwg.org/#concept-element-attributes-remove) it from *element*.
3. Return *attr*.

An [element](https://dom.spec.whatwg.org/#concept-element) can have an associated *unique identifier (ID)*

Historically [elements](https://dom.spec.whatwg.org/#concept-element) could have multiple identifiers e.g., by using the HTML id [attribute](https://dom.spec.whatwg.org/#concept-attribute) and a DTD. This specification makes [ID](https://dom.spec.whatwg.org/#concept-id) a concept of the DOM and allows for only one per [element](https://dom.spec.whatwg.org/#concept-element), given by an [id attribute](https://dom.spec.whatwg.org/#concept-named-attribute).

Use these [attribute change steps](https://dom.spec.whatwg.org/#concept-element-attributes-change-ext) to update an [element](https://dom.spec.whatwg.org/#concept-element)’s [ID](https://dom.spec.whatwg.org/#concept-id):

1. If *localName* is id, *namespace* is null, and *value* is null or the empty string, then unset *element*’s [ID](https://dom.spec.whatwg.org/#concept-id).
2. Otherwise, if *localName* is id, *namespace* is null, then set *element*’s [ID](https://dom.spec.whatwg.org/#concept-id) to *value*.

While this specification defines requirements for class, id, and slot [attributes](https://dom.spec.whatwg.org/#concept-attribute) on any [element](https://dom.spec.whatwg.org/#concept-element), it makes no claims as to whether using them is conforming or not.

A [node](https://dom.spec.whatwg.org/#concept-node)’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) of type [Element](https://dom.spec.whatwg.org/#element) is known as a *parent element*. If the [node](https://dom.spec.whatwg.org/#concept-node) has a [parent](https://dom.spec.whatwg.org/#concept-tree-parent) of a different type, its [parent element](https://dom.spec.whatwg.org/#parent-element) is null.

*namespace* = *element* . [namespaceURI](https://dom.spec.whatwg.org/#dom-element-namespaceuri)

Returns the [namespace](https://dom.spec.whatwg.org/#concept-element-namespace).

*prefix* = *element* . [prefix](https://dom.spec.whatwg.org/#dom-element-prefix)

Returns the [namespace prefix](https://dom.spec.whatwg.org/#concept-element-namespace-prefix).

*localName* = *element* . [localName](https://dom.spec.whatwg.org/#dom-element-localname)

Returns the [local name](https://dom.spec.whatwg.org/#concept-element-local-name).

*qualifiedName* = *element* . [tagName](https://dom.spec.whatwg.org/#dom-element-tagname)

Returns the [qualified name](https://dom.spec.whatwg.org/#concept-element-qualified-name). (The return value is uppercased in an [HTML document](https://dom.spec.whatwg.org/#html-document).)

The *namespaceURI* attribute’s getter must return the [context object](https://dom.spec.whatwg.org/#context-object)’s [namespace](https://dom.spec.whatwg.org/#concept-element-namespace).

The *prefix* attribute’s getter must return the [context object](https://dom.spec.whatwg.org/#context-object)’s [namespace prefix](https://dom.spec.whatwg.org/#concept-element-namespace-prefix).

The *localName* attribute’s getter must return the [context object](https://dom.spec.whatwg.org/#context-object)’s [local name](https://dom.spec.whatwg.org/#concept-element-local-name).

The *tagName* attribute’s getter must run these steps:

1. Let *qualifiedName* be [context object](https://dom.spec.whatwg.org/#context-object)’s [qualified name](https://dom.spec.whatwg.org/#concept-element-qualified-name).
2. If the [context object](https://dom.spec.whatwg.org/#context-object) is in the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace) and its [node document](https://dom.spec.whatwg.org/#concept-node-document) is an [HTML document](https://dom.spec.whatwg.org/#html-document), let *qualifiedName* be [converted to ASCII uppercase](https://dom.spec.whatwg.org/#converted-to-ascii-uppercase).
3. Return *qualifiedName*.

IDL attributes that are defined to *reflect* a content [attribute](https://dom.spec.whatwg.org/#concept-attribute) of a given *name*, must have a getter and setter that follow these steps:

getter

Return the result of running [get an attribute value](https://dom.spec.whatwg.org/#concept-element-attributes-get-value) given [context object](https://dom.spec.whatwg.org/#context-object) and *name*.

setter

[Set an attribute value](https://dom.spec.whatwg.org/#concept-element-attributes-set-value) for the [context object](https://dom.spec.whatwg.org/#context-object) using *name* and the given value.

The *id* attribute must [reflect](https://dom.spec.whatwg.org/#concept-reflect) the "id" content attribute.

The *className* attribute must [reflect](https://dom.spec.whatwg.org/#concept-reflect) the "class" content attribute.

The *classList* attribute’s getter must return a [DOMTokenList](https://dom.spec.whatwg.org/#domtokenlist) object whose associated [element](https://dom.spec.whatwg.org/#concept-element) is the [context object](https://dom.spec.whatwg.org/#context-object) and whose associated [attribute](https://dom.spec.whatwg.org/#concept-attribute)’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name) is class. The [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens) of this particular [DOMTokenList](https://dom.spec.whatwg.org/#domtokenlist) object are also known as the [element](https://dom.spec.whatwg.org/#concept-element)’s *classes*.

The *slot* attribute must [reflect](https://dom.spec.whatwg.org/#concept-reflect) the "slot" content attribute.

id, class, and slot are effectively superglobal attributes as they can appear on any element, regardless of that element’s namespace.

The *hasAttributes()* method, when invoked, must return false if [context object](https://dom.spec.whatwg.org/#context-object)’s [attribute list](https://dom.spec.whatwg.org/#concept-element-attribute) is empty, and true otherwise.

The *attributes* attribute must return the associated [NamedNodeMap](https://dom.spec.whatwg.org/#namednodemap).

The *getAttributeNames()* method, when invoked, must return the [qualified names](https://dom.spec.whatwg.org/#concept-attribute-qualified-name) of the [attributes](https://dom.spec.whatwg.org/#concept-attribute) in the [context object](https://dom.spec.whatwg.org/#context-object)’s [attribute list](https://dom.spec.whatwg.org/#concept-element-attribute), in order, and the empty sequence otherwise.

These are not guaranteed to be unique.

The *getAttribute(qualifiedName)* method, when invoked, must run these steps:

1. Let *attr* be the result of [getting an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-get-by-name) given *qualifiedName* and the [context object](https://dom.spec.whatwg.org/#context-object).
2. If *attr* is null, return null.
3. Return *attr*’s [value](https://dom.spec.whatwg.org/#concept-attribute-value).

The *getAttributeNS(namespace, localName)* method, when invoked, must these steps:

1. Let *attr* be the result of [getting an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-get-by-namespace) given *namespace*, *localName*, and the [context object](https://dom.spec.whatwg.org/#context-object).
2. If *attr* is null, return null.
3. Return *attr*’s [value](https://dom.spec.whatwg.org/#concept-attribute-value).

The *setAttribute(qualifiedName, value)* method, when invoked, must run these steps:

1. If *qualifiedName* does not match the [Name](https://www.w3.org/TR/xml/#NT-Name) production in XML, then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror).
2. If the [context object](https://dom.spec.whatwg.org/#context-object) is in the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace) and its [node document](https://dom.spec.whatwg.org/#concept-node-document) is an [HTML document](https://dom.spec.whatwg.org/#html-document), let *qualifiedName* be [converted to ASCII lowercase](https://dom.spec.whatwg.org/#converted-to-ascii-lowercase).
3. Let *attribute* be the first [attribute](https://dom.spec.whatwg.org/#concept-attribute) in [context object](https://dom.spec.whatwg.org/#context-object)’s [attribute list](https://dom.spec.whatwg.org/#concept-element-attribute) whose [qualified name](https://dom.spec.whatwg.org/#concept-attribute-qualified-name) is *qualifiedName*, and null otherwise.
4. If *attribute* is null, create an [attribute](https://dom.spec.whatwg.org/#concept-attribute) whose [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name) is *qualifiedName* and [value](https://dom.spec.whatwg.org/#concept-attribute-value) is *value*, [append](https://dom.spec.whatwg.org/#concept-element-attributes-append) this [attribute](https://dom.spec.whatwg.org/#concept-attribute) to the [context object](https://dom.spec.whatwg.org/#context-object)’s [attribute list](https://dom.spec.whatwg.org/#concept-element-attribute), and then terminate these steps.
5. [Change](https://dom.spec.whatwg.org/#concept-element-attributes-change) *attribute* from [context object](https://dom.spec.whatwg.org/#context-object) to *value*.

The *setAttributeNS(namespace, qualifiedName, value)* method, when invoked, must run these steps:

1. Let *namespace*, *prefix*, and *localName* be the result of passing *namespace* and *qualifiedName* to [validate and extract](https://dom.spec.whatwg.org/#validate-and-extract). Rethrow any exceptions.
2. [Set an attribute value](https://dom.spec.whatwg.org/#concept-element-attributes-set-value) for the [context object](https://dom.spec.whatwg.org/#context-object) using *localName*, *value*, and also *prefix* and *namespace*.

The *removeAttribute(qualifiedName)* method, when invoked, must [remove an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-remove-by-name) given *qualifiedName* and the [context object](https://dom.spec.whatwg.org/#context-object), and then return undefined.

The *removeAttributeNS(namespace, localName)* method must [remove an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-remove-by-namespace) given *namespace*, *localName*, and the [context object](https://dom.spec.whatwg.org/#context-object), and then return undefined.

The *hasAttribute(qualifiedName)* method, when invoked, must run these steps:

1. If the [context object](https://dom.spec.whatwg.org/#context-object) is in the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace) and its [node document](https://dom.spec.whatwg.org/#concept-node-document) is an [HTML document](https://dom.spec.whatwg.org/#html-document), let *qualifiedName* be [converted to ASCII lowercase](https://dom.spec.whatwg.org/#converted-to-ascii-lowercase).
2. Return true if the [context object](https://dom.spec.whatwg.org/#context-object) [has](https://dom.spec.whatwg.org/#concept-element-attribute-has) an [attribute](https://dom.spec.whatwg.org/#concept-attribute) whose [qualified name](https://dom.spec.whatwg.org/#concept-attribute-qualified-name) is *qualifiedName*, and false otherwise.

The *hasAttributeNS(namespace, localName)* method, when invoked, must run these steps:

1. If *namespace* is the empty string, set it to null.
2. Return true if the [context object](https://dom.spec.whatwg.org/#context-object) [has](https://dom.spec.whatwg.org/#concept-element-attribute-has) an [attribute](https://dom.spec.whatwg.org/#concept-attribute) whose [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace) is *namespace* and [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name) is *localName*, and false otherwise.

The *getAttributeNode(qualifiedName)* method, when invoked, must return the result of [getting an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-get-by-name) given *qualifiedName* and the [context object](https://dom.spec.whatwg.org/#context-object).

The *getAttributeNodeNS(namespace, localName)* method, when invoked, must return the result of [getting an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-get-by-namespace) given *namespace*, *localName*, and the [context object](https://dom.spec.whatwg.org/#context-object).

The *setAttributeNode(attr)* and *setAttributeNodeNS(attr)* methods, when invoked, must return the result of [setting an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-set) given *attr* and the [context object](https://dom.spec.whatwg.org/#context-object). Rethrow any exceptions.

The *removeAttributeNode(attr)* method, when invoked, must run these steps:

1. If *attr* is not in [context object](https://dom.spec.whatwg.org/#context-object)’s [attribute list](https://dom.spec.whatwg.org/#concept-element-attribute), then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotFoundError](https://heycam.github.io/webidl/#notfounderror).
2. [Remove](https://dom.spec.whatwg.org/#concept-element-attributes-remove) *attr* from [context object](https://dom.spec.whatwg.org/#context-object).
3. Return *attr*.

var shadow = *element* . [attachShadow(init)](https://dom.spec.whatwg.org/#dom-element-attachshadow)

Creates a [shadow root](https://dom.spec.whatwg.org/#concept-shadow-root) for *element* and returns it.

var shadow = *element* . [shadowRoot](https://dom.spec.whatwg.org/#dom-element-shadowroot)

Returns *element*’s [shadow root](https://dom.spec.whatwg.org/#concept-element-shadow-root), if any, and if [shadow root](https://dom.spec.whatwg.org/#concept-shadow-root)’s [mode](https://dom.spec.whatwg.org/#shadowroot-mode) is "open", and null otherwise.

The *attachShadow(init)* method, when invoked, must run these steps:

1. If [context object](https://dom.spec.whatwg.org/#context-object)’s [namespace](https://dom.spec.whatwg.org/#concept-element-namespace) is *not* the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace), then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotSupportedError](https://heycam.github.io/webidl/#notsupportederror).
2. If [context object](https://dom.spec.whatwg.org/#context-object)’s [local name](https://dom.spec.whatwg.org/#concept-element-local-name) is *not* a [valid custom element name](https://html.spec.whatwg.org/multipage/scripting.html#valid-custom-element-name), "article", "aside", "blockquote", "body", "div", "footer", "h1", "h2", "h3", "h4", "h5", "h6", "header", "nav", "p", "section", or "span", then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotSupportedError](https://heycam.github.io/webidl/#notsupportederror).
3. If [context object](https://dom.spec.whatwg.org/#context-object) is a [shadow host](https://dom.spec.whatwg.org/#element-shadow-host), then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidStateError](https://heycam.github.io/webidl/#invalidstateerror).
4. Let *shadow* be a new [shadow root](https://dom.spec.whatwg.org/#concept-shadow-root) whose [node document](https://dom.spec.whatwg.org/#concept-node-document) is [context object](https://dom.spec.whatwg.org/#context-object)’s [node document](https://dom.spec.whatwg.org/#concept-node-document), [host](https://dom.spec.whatwg.org/#concept-documentfragment-host) is [context object](https://dom.spec.whatwg.org/#context-object), and [mode](https://dom.spec.whatwg.org/#shadowroot-mode) is *init*’s [mode](https://dom.spec.whatwg.org/#dom-shadowrootinit-mode).
5. Set [context object](https://dom.spec.whatwg.org/#context-object)’s [shadow root](https://dom.spec.whatwg.org/#concept-element-shadow-root) to *shadow*.
6. Return *shadow*.

The *shadowRoot* attribute’s getter must run these steps:

1. Let *shadow* be [context object](https://dom.spec.whatwg.org/#context-object)’s [shadow root](https://dom.spec.whatwg.org/#concept-element-shadow-root).
2. If *shadow* is null or its [mode](https://dom.spec.whatwg.org/#shadowroot-mode) is "closed", then return null.
3. Return *shadow*.

*element* . [closest(selectors)](https://dom.spec.whatwg.org/#dom-element-closest)

Returns the first (starting at *element*) [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) that matches *selectors*, and null otherwise.

*element* . [matches(selectors)](https://dom.spec.whatwg.org/#dom-element-matches)

Returns true if matching *selectors* against *element*’s [root](https://dom.spec.whatwg.org/#concept-tree-root) yields *element*, and false otherwise.

The *closest(selectors)* method, when invoked, must run these steps:

1. Let *s* be the result of [parse a selector](https://drafts.csswg.org/selectors-4/#parse-a-selector) from *selectors*. [[SELECTORS4]](https://dom.spec.whatwg.org/#biblio-selectors4)
2. If *s* is failure, [throw](https://heycam.github.io/webidl/#dfn-throw) a [SyntaxError](https://heycam.github.io/webidl/#syntaxerror).
3. Let *elements* be [context object](https://dom.spec.whatwg.org/#context-object)’s [inclusive ancestors](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) that are [elements](https://dom.spec.whatwg.org/#concept-element), in reverse [tree order](https://dom.spec.whatwg.org/#concept-tree-order).
4. For each *element* in *elements*, if [match a selector against an element](https://drafts.csswg.org/selectors-4/#match-a-selector-against-an-element), using *s*, *element*, and [:scope element](https://drafts.csswg.org/selectors-4/#scope-element) [context object](https://dom.spec.whatwg.org/#context-object), returns success, return *element*. [[SELECTORS4]](https://dom.spec.whatwg.org/#biblio-selectors4)
5. Return null.

The *matches(selectors)* and *webkitMatchesSelector(selectors)* methods, when invoked, must run these steps:

1. Let *s* be the result of [parse a selector](https://drafts.csswg.org/selectors-4/#parse-a-selector) from *selectors*. [[SELECTORS4]](https://dom.spec.whatwg.org/#biblio-selectors4)
2. If *s* is failure, [throw](https://heycam.github.io/webidl/#dfn-throw) a [SyntaxError](https://heycam.github.io/webidl/#syntaxerror).
3. Return true if the result of [match a selector against an element](https://drafts.csswg.org/selectors-4/#match-a-selector-against-an-element), using *s*, *element*, and [:scope element](https://drafts.csswg.org/selectors-4/#scope-element) [context object](https://dom.spec.whatwg.org/#context-object), returns success, and false otherwise. [[SELECTORS4]](https://dom.spec.whatwg.org/#biblio-selectors4)

The *getElementsByTagName(qualifiedName)* method, when invoked, must return the [list of elements with qualified name *qualifiedName*](https://dom.spec.whatwg.org/#concept-getelementsbytagname) for the [context object](https://dom.spec.whatwg.org/#context-object).

The *getElementsByTagNameNS(namespace, localName)* method must return the [list of elements with namespace *namespace* and local name *localName*](https://dom.spec.whatwg.org/#concept-getelementsbytagnamens) for the [context object](https://dom.spec.whatwg.org/#context-object).

The *getElementsByClassName(classNames)* method must return the [list of elements with class names *classNames*](https://dom.spec.whatwg.org/#concept-getelementsbyclassname) for the [context object](https://dom.spec.whatwg.org/#context-object).

To *insert adjacent*, given an [element](https://dom.spec.whatwg.org/#concept-element) *element*, string *where*, and a [node](https://dom.spec.whatwg.org/#concept-node) *node*, run the steps associated with the first [ASCII case-insensitive](https://dom.spec.whatwg.org/#ascii-case-insensitive) match for *where*:

"beforebegin"

If *element*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is null, return null.

Return the result of [pre-inserting](https://dom.spec.whatwg.org/#concept-node-pre-insert) *node* into *element*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) before *element*. Rethrow any exceptions.

"afterbegin"

Return the result of [pre-inserting](https://dom.spec.whatwg.org/#concept-node-pre-insert) *node* into *element* before *element*’s [first child](https://dom.spec.whatwg.org/#concept-tree-first-child). Rethrow any exceptions.

"beforeend"

Return the result of [pre-inserting](https://dom.spec.whatwg.org/#concept-node-pre-insert) *node* into *element* before null. Rethrow any exceptions.

"afterend"

If *element*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is null, return null.

Return the result of [pre-inserting](https://dom.spec.whatwg.org/#concept-node-pre-insert) *node* into *element*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) before *element*’s [next sibling](https://dom.spec.whatwg.org/#concept-tree-next-sibling). Rethrow any exceptions.

Otherwise

[Throw](https://heycam.github.io/webidl/#dfn-throw) a [SyntaxError](https://heycam.github.io/webidl/#syntaxerror).

The *insertAdjacentElement(where, element)* method, when invoked, must return the result of running [insert adjacent](https://dom.spec.whatwg.org/#insert-adjacent), given [context object](https://dom.spec.whatwg.org/#context-object), *where*, and *element*. Rethrow any exceptions.

The *insertAdjacentText(where, data)* method, when invoked, must run these steps:

1. Let *text* be a new [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node) whose [data](https://dom.spec.whatwg.org/#concept-cd-data) is *data* and [node document](https://dom.spec.whatwg.org/#concept-node-document) is [context object](https://dom.spec.whatwg.org/#context-object)’s [node document](https://dom.spec.whatwg.org/#concept-node-document).
2. Run [insert adjacent](https://dom.spec.whatwg.org/#insert-adjacent), given [context object](https://dom.spec.whatwg.org/#context-object), *where*, and *text*. Rethrow any exceptions.

This method returns nothing because it existed before we had a chance to design it.

**4.9.1. Interface** [**NamedNodeMap**](https://dom.spec.whatwg.org/#namednodemap)

[Exposed=Window, LegacyUnenumerableNamedProperties]

interface *NamedNodeMap* {

readonly attribute unsigned long [length](https://dom.spec.whatwg.org/#dom-namednodemap-length);

getter [Attr](https://dom.spec.whatwg.org/#attr)? [item](https://dom.spec.whatwg.org/#dom-namednodemap-item)(unsigned long *index*);

getter [Attr](https://dom.spec.whatwg.org/#attr)? [getNamedItem](https://dom.spec.whatwg.org/#dom-namednodemap-getnameditem)(DOMString *qualifiedName*);

[Attr](https://dom.spec.whatwg.org/#attr)? [getNamedItemNS](https://dom.spec.whatwg.org/#dom-namednodemap-getnameditemns)(DOMString? *namespace*, DOMString *localName*);

[CEReactions] [Attr](https://dom.spec.whatwg.org/#attr)? [setNamedItem](https://dom.spec.whatwg.org/#dom-namednodemap-setnameditem)([Attr](https://dom.spec.whatwg.org/#attr) *attr*);

[CEReactions] [Attr](https://dom.spec.whatwg.org/#attr)? [setNamedItemNS](https://dom.spec.whatwg.org/#dom-namednodemap-setnameditemns)([Attr](https://dom.spec.whatwg.org/#attr) *attr*);

[CEReactions] [Attr](https://dom.spec.whatwg.org/#attr) [removeNamedItem](https://dom.spec.whatwg.org/#dom-namednodemap-removenameditem)(DOMString *qualifiedName*);

[CEReactions] [Attr](https://dom.spec.whatwg.org/#attr) [removeNamedItemNS](https://dom.spec.whatwg.org/#dom-namednodemap-removenameditemns)(DOMString? *namespace*, DOMString *localName*);

};

A [NamedNodeMap](https://dom.spec.whatwg.org/#namednodemap) has an associated *element* (an [element](https://dom.spec.whatwg.org/#concept-element)).

A [NamedNodeMap](https://dom.spec.whatwg.org/#namednodemap) object’s *attribute list* is its [element](https://dom.spec.whatwg.org/#concept-namednodemap-element)’s [attribute list](https://dom.spec.whatwg.org/#concept-element-attribute).

A [NamedNodeMap](https://dom.spec.whatwg.org/#namednodemap) object’s [supported property indices](https://heycam.github.io/webidl/#dfn-supported-property-indices) are the numbers in the range zero to the number of [attributes](https://dom.spec.whatwg.org/#concept-attribute) in its [attribute list](https://dom.spec.whatwg.org/#concept-namednodemap-attribute) map minus one, unless the [attribute list](https://dom.spec.whatwg.org/#concept-namednodemap-attribute) is empty, in which case there are no [supported property indices](https://heycam.github.io/webidl/#dfn-supported-property-indices).

The *length* attribute’s getter must return the number of [attributes](https://dom.spec.whatwg.org/#concept-attribute) in the [attribute list](https://dom.spec.whatwg.org/#concept-namednodemap-attribute).

The *item(index)* method, when invoked, must run these steps:

1. If *index* is equal to or greater than the number of [attributes](https://dom.spec.whatwg.org/#concept-attribute) in the [attribute list](https://dom.spec.whatwg.org/#concept-namednodemap-attribute), return null.
2. Otherwise, return the *index*th [attribute](https://dom.spec.whatwg.org/#concept-attribute) in the [attribute list](https://dom.spec.whatwg.org/#concept-namednodemap-attribute).

A [NamedNodeMap](https://dom.spec.whatwg.org/#namednodemap) object’s [supported property names](https://heycam.github.io/webidl/#dfn-supported-property-names) are the return value of running these steps:

1. Let *names* be the [qualified names](https://dom.spec.whatwg.org/#concept-attribute-qualified-name) of the [attributes](https://dom.spec.whatwg.org/#concept-attribute) in this [NamedNodeMap](https://dom.spec.whatwg.org/#namednodemap) object’s [attribute list](https://dom.spec.whatwg.org/#concept-namednodemap-attribute), with duplicates omitted, in order.
2. If this [NamedNodeMap](https://dom.spec.whatwg.org/#namednodemap) object’s [element](https://dom.spec.whatwg.org/#concept-namednodemap-element) is in the [HTML namespace](https://dom.spec.whatwg.org/#html-namespace) and its [node document](https://dom.spec.whatwg.org/#concept-node-document) is an [HTML document](https://dom.spec.whatwg.org/#html-document), then for each *name* in *names*, run these substeps:
   1. Let *lowercaseName* be *name*, [converted to ASCII lowercase](https://dom.spec.whatwg.org/#converted-to-ascii-lowercase).
   2. If *lowercaseName* is not equal to *name*, remove *name* from *names*.
3. Return *names*.

The *getNamedItem(qualifiedName)* method, when invoked, must return the result of [getting an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-get-by-name) given *qualifiedName* and [element](https://dom.spec.whatwg.org/#concept-namednodemap-element).

The *getNamedItemNS(namespace, localName)* method, when invoked, must return the result of [getting an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-get-by-namespace) given *namespace*, *localName*, and [element](https://dom.spec.whatwg.org/#concept-namednodemap-element).

The *setNamedItem(attr)* and *setNamedItemNS(attr)* methods, when invoked, must return the result of [setting an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-set) given *attr* and [element](https://dom.spec.whatwg.org/#concept-namednodemap-element). Rethrow any exceptions.

The *removeNamedItem(qualifiedName)* method, when invoked, must run these steps:

1. Let *attr* be the result of [removing an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-remove-by-name) given *qualifiedName* and [element](https://dom.spec.whatwg.org/#concept-namednodemap-element).
2. If *attr* is null, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotFoundError](https://heycam.github.io/webidl/#notfounderror).
3. Return *attr*.

The *removeNamedItemNS(namespace, localName)* method, when invoked, must run these steps:

1. Let *attr* be the result of [removing an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-remove-by-namespace) given *namespace*, *localName*, and [element](https://dom.spec.whatwg.org/#concept-namednodemap-element).
2. If *attr* is null, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotFoundError](https://heycam.github.io/webidl/#notfounderror).
3. Return *attr*.

**4.9.2. Interface** [**Attr**](https://dom.spec.whatwg.org/#attr)

[Exposed=Window]

interface *Attr* {

readonly attribute DOMString? [namespaceURI](https://dom.spec.whatwg.org/#dom-attr-namespaceuri);

readonly attribute DOMString? [prefix](https://dom.spec.whatwg.org/#dom-attr-prefix);

readonly attribute DOMString [localName](https://dom.spec.whatwg.org/#dom-attr-localname);

readonly attribute DOMString [name](https://dom.spec.whatwg.org/#dom-attr-name);

readonly attribute DOMString [nodeName](https://dom.spec.whatwg.org/#dom-attr-nodename); // historical alias of .name

[CEReactions] attribute DOMString [value](https://dom.spec.whatwg.org/#dom-attr-value);

[CEReactions, TreatNullAs=EmptyString] attribute DOMString [nodeValue](https://dom.spec.whatwg.org/#dom-attr-nodevalue); // historical alias of .value

[CEReactions, TreatNullAs=EmptyString] attribute DOMString [textContent](https://dom.spec.whatwg.org/#dom-attr-textcontent); // historical alias of .value

readonly attribute [Element](https://dom.spec.whatwg.org/#element)? [ownerElement](https://dom.spec.whatwg.org/#dom-attr-ownerelement);

readonly attribute boolean [specified](https://dom.spec.whatwg.org/#dom-attr-specified); // useless; always returns true

};

[Attr](https://dom.spec.whatwg.org/#attr) objects are simply known as *attributes*. They are sometimes referred to as *content attributes* to avoid confusion with IDL attributes.

[Attributes](https://dom.spec.whatwg.org/#concept-attribute) have a *namespace* (null or a non-empty string), *namespace prefix* (null or a non-empty string), *local name* (a non-empty string), *value* (a string), and *element* (null or an [element](https://dom.spec.whatwg.org/#concept-element)).

If designed today they would just have a name and value. ☹

An [attribute](https://dom.spec.whatwg.org/#concept-attribute)’s *qualified name* is its [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name) if its [namespace prefix](https://dom.spec.whatwg.org/#concept-attribute-namespace-prefix) is null, and its [namespace prefix](https://dom.spec.whatwg.org/#concept-attribute-namespace-prefix), followed by ":", followed by its [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name), otherwise.

User agents could have this as an internal slot as an optimization, but are not required to do so. The standard has this concept for readability.

When an [attribute](https://dom.spec.whatwg.org/#concept-attribute) is created, its [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name) is given. Unless explicitly given when an [attribute](https://dom.spec.whatwg.org/#concept-attribute) is created, its [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace), [namespace prefix](https://dom.spec.whatwg.org/#concept-attribute-namespace-prefix), and [element](https://dom.spec.whatwg.org/#concept-attribute-element) are set to null, and its [value](https://dom.spec.whatwg.org/#concept-attribute-value) is set to the empty string.

An *A attribute* is an [attribute](https://dom.spec.whatwg.org/#concept-attribute) whose [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name) is *A* and whose [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace) and [namespace prefix](https://dom.spec.whatwg.org/#concept-attribute-namespace-prefix) are null.

The *namespaceURI* attribute’s getter must return the [namespace](https://dom.spec.whatwg.org/#concept-attribute-namespace).

The *prefix* attribute’s getter must return the [namespace prefix](https://dom.spec.whatwg.org/#concept-attribute-namespace-prefix).

The *localName* attribute’s getter must return the [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name).

The *name* attribute’s getter, and *nodeName* attribute’s getter, must return the [qualified name](https://dom.spec.whatwg.org/#concept-attribute-qualified-name).

The *value* attribute’s getter, *nodeValue* attribute’s getter, and *textContent* attribute’s getter, must return the [value](https://dom.spec.whatwg.org/#concept-attribute-value).

The [value](https://dom.spec.whatwg.org/#dom-attr-value) attribute’s setter, [nodeValue](https://dom.spec.whatwg.org/#dom-attr-nodevalue) attribute’s setter, and [textContent](https://dom.spec.whatwg.org/#dom-attr-textcontent) attribute’s setter, must run these steps:

1. If [context object](https://dom.spec.whatwg.org/#context-object)’s [element](https://dom.spec.whatwg.org/#concept-attribute-element) is null, set [context object](https://dom.spec.whatwg.org/#context-object)’s [value](https://dom.spec.whatwg.org/#concept-attribute-value) to the given value.
2. Otherwise, [change](https://dom.spec.whatwg.org/#concept-element-attributes-change) the [context object](https://dom.spec.whatwg.org/#context-object) from [context object](https://dom.spec.whatwg.org/#context-object)’s [element](https://dom.spec.whatwg.org/#concept-attribute-element) to the given value.

Unlike [node](https://dom.spec.whatwg.org/#concept-node)’s [textContent](https://dom.spec.whatwg.org/#dom-node-textcontent), no special null handling is required.

The *ownerElement* attribute’s getter must return [context object](https://dom.spec.whatwg.org/#context-object)’s [element](https://dom.spec.whatwg.org/#concept-attribute-element).

The *specified* attribute’s getter must return true.

**4.10. Interface** [**CharacterData**](https://dom.spec.whatwg.org/#characterdata)

[Exposed=Window]

interface *CharacterData* : [Node](https://dom.spec.whatwg.org/#node) {

[TreatNullAs=EmptyString] attribute DOMString [data](https://dom.spec.whatwg.org/#dom-characterdata-data);

readonly attribute unsigned long [length](https://dom.spec.whatwg.org/#dom-characterdata-length);

DOMString [substringData](https://dom.spec.whatwg.org/#dom-characterdata-substringdata)(unsigned long *offset*, unsigned long *count*);

void [appendData](https://dom.spec.whatwg.org/#dom-characterdata-appenddata)(DOMString *data*);

void [insertData](https://dom.spec.whatwg.org/#dom-characterdata-insertdata)(unsigned long *offset*, DOMString *data*);

void [deleteData](https://dom.spec.whatwg.org/#dom-characterdata-deletedata)(unsigned long *offset*, unsigned long *count*);

void [replaceData](https://dom.spec.whatwg.org/#dom-characterdata-replacedata)(unsigned long *offset*, unsigned long *count*, DOMString *data*);

};

[CharacterData](https://dom.spec.whatwg.org/#characterdata) is an abstract interface and does not exist as [node](https://dom.spec.whatwg.org/#concept-node). It is used by [Text](https://dom.spec.whatwg.org/#text), [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction), and [Comment](https://dom.spec.whatwg.org/#comment) [nodes](https://dom.spec.whatwg.org/#concept-node).

Each [node](https://dom.spec.whatwg.org/#concept-node) inheriting from the [CharacterData](https://dom.spec.whatwg.org/#characterdata) interface has an associated mutable string called *data*.

To *replace data* of node *node* with offset *offset*, count *count*, and data *data*, run these steps:

1. Let *length* be *node*’s [length](https://dom.spec.whatwg.org/#dom-characterdata-length) attribute value.
2. If *offset* is greater than *length*, then [throw](https://heycam.github.io/webidl/#dfn-throw) an [IndexSizeError](https://heycam.github.io/webidl/#indexsizeerror).
3. If *offset* plus *count* is greater than *length* let *count* be *length* minus *offset*.
4. [Queue a mutation record](https://dom.spec.whatwg.org/#queue-a-mutation-record) of "characterData" for *node* with oldValue *node*’s [data](https://dom.spec.whatwg.org/#concept-cd-data).
5. Insert *data* into *node*’s [data](https://dom.spec.whatwg.org/#concept-cd-data) after *offset* [code units](https://heycam.github.io/webidl/#dfn-code-unit).
6. Let *delete offset* be *offset* plus the number of [code units](https://heycam.github.io/webidl/#dfn-code-unit) in *data*.
7. Starting from *delete offset* [code units](https://heycam.github.io/webidl/#dfn-code-unit), remove *count* [code units](https://heycam.github.io/webidl/#dfn-code-unit) from *node*’s [data](https://dom.spec.whatwg.org/#concept-cd-data).
8. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [start node](https://dom.spec.whatwg.org/#concept-range-start-node) is *node* and [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) is greater than *offset* but less than or equal to *offset* plus *count*, set its [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) to *offset*.
9. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [end node](https://dom.spec.whatwg.org/#concept-range-end-node) is *node* and [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset) is greater than *offset* but less than or equal to *offset* plus *count*, set its [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset) to *offset*.
10. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [start node](https://dom.spec.whatwg.org/#concept-range-start-node) is *node* and [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) is greater than *offset* plus *count*, increase its [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) by the number of [code units](https://heycam.github.io/webidl/#dfn-code-unit) in *data*, then decrease it by *count*.
11. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [end node](https://dom.spec.whatwg.org/#concept-range-end-node) is *node* and [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset) is greater than *offset* plus *count*, increase its [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset) by the number of [code units](https://heycam.github.io/webidl/#dfn-code-unit) in *data*, then decrease it by *count*.

To *substring data* with node *node*, offset *offset*, and count *count*, run these steps:

1. Let *length* be *node*’s [length](https://dom.spec.whatwg.org/#dom-characterdata-length) attribute value.
2. If *offset* is greater than *length*, then [throw](https://heycam.github.io/webidl/#dfn-throw) an [IndexSizeError](https://heycam.github.io/webidl/#indexsizeerror).
3. If *offset* plus *count* is greater than *length*, return a string whose value is the [code units](https://heycam.github.io/webidl/#dfn-code-unit) from the *offset*th [code unit](https://heycam.github.io/webidl/#dfn-code-unit) to the end of *node*’s [data](https://dom.spec.whatwg.org/#concept-cd-data), and then terminate these steps.
4. Return a string whose value is the [code units](https://heycam.github.io/webidl/#dfn-code-unit) from the *offset*th [code unit](https://heycam.github.io/webidl/#dfn-code-unit) to the *offset*+*count*th [code unit](https://heycam.github.io/webidl/#dfn-code-unit) in *node*’s [data](https://dom.spec.whatwg.org/#concept-cd-data).

The *data* attribute’s getter must return [context object](https://dom.spec.whatwg.org/#context-object)’s [data](https://dom.spec.whatwg.org/#concept-cd-data). Its setter must [replace data](https://dom.spec.whatwg.org/#concept-cd-replace) with node [context object](https://dom.spec.whatwg.org/#context-object) offset 0, count [length](https://dom.spec.whatwg.org/#dom-characterdata-length) attribute value, and data new value.

The *length* attribute’s getter must return the number of [code units](https://heycam.github.io/webidl/#dfn-code-unit) in [context object](https://dom.spec.whatwg.org/#context-object)’s [data](https://dom.spec.whatwg.org/#concept-cd-data).

The *substringData(offset, count)* method, when invoked, must return the result of running [substring data](https://dom.spec.whatwg.org/#concept-cd-substring) with node [context object](https://dom.spec.whatwg.org/#context-object), offset *offset*, and count *count*.

The *appendData(data)* method, when invoked, must [replace data](https://dom.spec.whatwg.org/#concept-cd-replace) with node [context object](https://dom.spec.whatwg.org/#context-object), offset [length](https://dom.spec.whatwg.org/#dom-characterdata-length) attribute value, count 0, and data *data*.

The *insertData(offset, data)* method, when invoked, must [replace data](https://dom.spec.whatwg.org/#concept-cd-replace) with node [context object](https://dom.spec.whatwg.org/#context-object), offset *offset*, count 0, and data *data*.

The *deleteData(offset, count)* method, when invoked, must [replace data](https://dom.spec.whatwg.org/#concept-cd-replace) with node [context object](https://dom.spec.whatwg.org/#context-object), offset *offset*, count *count*, and data the empty string.

The *replaceData(offset, count, data)* method, when invoked, must [replace data](https://dom.spec.whatwg.org/#concept-cd-replace) with node [context object](https://dom.spec.whatwg.org/#context-object), offset *offset*, count *count*, and data *data*.

**4.11. Interface** [**Text**](https://dom.spec.whatwg.org/#text)

[[Constructor](https://dom.spec.whatwg.org/#dom-text-text)(optional DOMString *data* = ""),

Exposed=Window]

interface *Text* : [CharacterData](https://dom.spec.whatwg.org/#characterdata) {

[NewObject] [Text](https://dom.spec.whatwg.org/#text) [splitText](https://dom.spec.whatwg.org/#dom-text-splittext)(unsigned long *offset*);

readonly attribute DOMString [wholeText](https://dom.spec.whatwg.org/#dom-text-wholetext);

};

*text* = new [Text([*data* = ""])](https://dom.spec.whatwg.org/#dom-text-text)

Returns a new [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node) whose [data](https://dom.spec.whatwg.org/#concept-cd-data) is *data*.

*text* . [splitText(offset)](https://dom.spec.whatwg.org/#dom-text-splittext)

Splits [data](https://dom.spec.whatwg.org/#concept-cd-data) at the given *offset* and returns the remainder as [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node).

*text* . [wholeText](https://dom.spec.whatwg.org/#dom-text-wholetext)

Returns the combined [data](https://dom.spec.whatwg.org/#concept-cd-data) of all direct [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node) [siblings](https://dom.spec.whatwg.org/#concept-tree-sibling).

The *Text(data)* constructor must return a new [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node) whose [data](https://dom.spec.whatwg.org/#concept-cd-data) is *data* and [node document](https://dom.spec.whatwg.org/#concept-node-document) is the global object’s associated [document](https://dom.spec.whatwg.org/#concept-document).

To *split* a [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node) *node* with offset *offset*, run these steps:

1. Let *length* be *node*’s [length](https://dom.spec.whatwg.org/#dom-characterdata-length) attribute value.
2. If *offset* is greater than *length*, then [throw](https://heycam.github.io/webidl/#dfn-throw) an [IndexSizeError](https://heycam.github.io/webidl/#indexsizeerror).
3. Let *count* be *length* minus *offset*.
4. Let *new data* be the result of [substringing data](https://dom.spec.whatwg.org/#concept-cd-substring) with node *node*, offset *offset*, and count *count*.
5. Let *new node* be a new [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node), with the same [node document](https://dom.spec.whatwg.org/#concept-node-document) as *node*. Set *new node*’s [data](https://dom.spec.whatwg.org/#concept-cd-data) to *new data*.
6. Let *parent* be *node*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
7. If *parent* is not null, run these substeps:
   1. [Insert](https://dom.spec.whatwg.org/#concept-node-insert) *new node* into *parent* before *node*’s [next sibling](https://dom.spec.whatwg.org/#concept-tree-next-sibling).
   2. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [start node](https://dom.spec.whatwg.org/#concept-range-start-node) is *node* and [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) is greater than *offset*, set its [start node](https://dom.spec.whatwg.org/#concept-range-start-node) to *new node* and decrease its [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) by *offset*.
   3. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [end node](https://dom.spec.whatwg.org/#concept-range-end-node) is *node* and [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset) is greater than *offset*, set its [end node](https://dom.spec.whatwg.org/#concept-range-end-node) to *new node* and decrease its [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset) by *offset*.
   4. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [start node](https://dom.spec.whatwg.org/#concept-range-start-node) is *parent* and [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) is equal to the [index](https://dom.spec.whatwg.org/#concept-tree-index) of *node* + 1, increase its [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) by one.
   5. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [end node](https://dom.spec.whatwg.org/#concept-range-end-node) is *parent* and [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset) is equal to the [index](https://dom.spec.whatwg.org/#concept-tree-index) of *node* + 1, increase its [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset) by one.
8. [Replace data](https://dom.spec.whatwg.org/#concept-cd-replace) with node *node*, offset *offset*, count *count*, and data the empty string.
9. If *parent* is null, run these substeps:
   1. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [start node](https://dom.spec.whatwg.org/#concept-range-start-node) is *node* and [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) is greater than *offset*, set its [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) to *offset*.
   2. For each [range](https://dom.spec.whatwg.org/#concept-range) whose [end node](https://dom.spec.whatwg.org/#concept-range-end-node) is *node* and [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset) is greater than *offset*, set its [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset) to *offset*.
10. Return *new node*.

The *splitText(offset)* method must [split](https://dom.spec.whatwg.org/#concept-text-split) the [context object](https://dom.spec.whatwg.org/#context-object) with offset *offset*.

The *contiguous* [*Text*](https://dom.spec.whatwg.org/#text) *nodes* of a node are the node itself, the [previous sibling](https://dom.spec.whatwg.org/#concept-tree-previous-sibling) [Text](https://dom.spec.whatwg.org/#text) node (if any) and its [contiguous Text nodes](https://dom.spec.whatwg.org/#contiguous-text-nodes), and the [next sibling](https://dom.spec.whatwg.org/#concept-tree-next-sibling) [Text](https://dom.spec.whatwg.org/#text) node (if any) and its [contiguous Text nodes](https://dom.spec.whatwg.org/#contiguous-text-nodes), avoiding any duplicates.

The *wholeText* attribute must return a concatenation of the [data](https://dom.spec.whatwg.org/#concept-cd-data) of the [contiguous Text nodes](https://dom.spec.whatwg.org/#contiguous-text-nodes) of the [context object](https://dom.spec.whatwg.org/#context-object), in [tree order](https://dom.spec.whatwg.org/#concept-tree-order).

**4.12. Interface** [**ProcessingInstruction**](https://dom.spec.whatwg.org/#processinginstruction)

[Exposed=Window]

interface *ProcessingInstruction* : [CharacterData](https://dom.spec.whatwg.org/#characterdata) {

readonly attribute DOMString [target](https://dom.spec.whatwg.org/#dom-processinginstruction-target);

};

[ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction) [nodes](https://dom.spec.whatwg.org/#concept-node) have an associated *target*.

The *target* attribute must return the [target](https://dom.spec.whatwg.org/#concept-pi-target).

**4.13. Interface** [**Comment**](https://dom.spec.whatwg.org/#comment)

[[Constructor](https://dom.spec.whatwg.org/#dom-comment-comment)(optional DOMString *data* = ""),

Exposed=Window]

interface *Comment* : [CharacterData](https://dom.spec.whatwg.org/#characterdata) {

};

*comment* = new [Comment([*data* = ""])](https://dom.spec.whatwg.org/#dom-comment-comment)

Returns a new [Comment](https://dom.spec.whatwg.org/#comment) [node](https://dom.spec.whatwg.org/#concept-node) whose [data](https://dom.spec.whatwg.org/#concept-cd-data) is *data*.

The *Comment(data)* constructor must return a new [Comment](https://dom.spec.whatwg.org/#comment) [node](https://dom.spec.whatwg.org/#concept-node) whose [data](https://dom.spec.whatwg.org/#concept-cd-data) is *data* and [node document](https://dom.spec.whatwg.org/#concept-node-document) is the global object’s associated [document](https://dom.spec.whatwg.org/#concept-document).

**5. Ranges**

**5.1. Introduction to "DOM Ranges"**

A [Range](https://dom.spec.whatwg.org/#range) object ([range](https://dom.spec.whatwg.org/#concept-range)) represents a sequence of content within a [node tree](https://dom.spec.whatwg.org/#concept-node-tree). Each [range](https://dom.spec.whatwg.org/#concept-range) has a [start](https://dom.spec.whatwg.org/#concept-range-start) and an [end](https://dom.spec.whatwg.org/#concept-range-end) which are [boundary points](https://dom.spec.whatwg.org/#concept-range-bp). A [boundary point](https://dom.spec.whatwg.org/#concept-range-bp) is a tuple consisting of a [node](https://dom.spec.whatwg.org/#concept-node) and a non-negative numeric [offset](https://dom.spec.whatwg.org/#concept-range-bp-offset). So in other words, a [range](https://dom.spec.whatwg.org/#concept-range) represents a piece of content within a [node tree](https://dom.spec.whatwg.org/#concept-node-tree) between two [boundary points](https://dom.spec.whatwg.org/#concept-range-bp).

[Ranges](https://dom.spec.whatwg.org/#concept-range) are frequently used in editing for selecting and copying content.

* [Element](https://dom.spec.whatwg.org/#element): p
  + [Element](https://dom.spec.whatwg.org/#element): <img src="insanity-wolf" alt="Little-ending BOM; decode as big-endian!">
  + [Text](https://dom.spec.whatwg.org/#text):  CSS 2.1 syndata is
  + [Element](https://dom.spec.whatwg.org/#element): <em>
    - [Text](https://dom.spec.whatwg.org/#text): awesome
  + [Text](https://dom.spec.whatwg.org/#text): !

In the [node tree](https://dom.spec.whatwg.org/#concept-node-tree) above, a [range](https://dom.spec.whatwg.org/#concept-range) can be used to represent the sequence “syndata is awes”. Assuming *p* is assigned to the p [element](https://dom.spec.whatwg.org/#concept-element), and *em* to the em [element](https://dom.spec.whatwg.org/#concept-element), this would be done as follows:

var range = new Range(),

firstText = p.childNodes[1],

secondText = em.firstChild

range.setStart(firstText, 9) // do not forget the leading space

range.setEnd(secondText, 4)

// range now stringifies to the aforementioned quote

[Attributes](https://dom.spec.whatwg.org/#concept-attribute) such as src and alt in the [node tree](https://dom.spec.whatwg.org/#concept-node-tree) above cannot be represented by a [range](https://dom.spec.whatwg.org/#concept-range). The [ranges](https://dom.spec.whatwg.org/#concept-range) concept is only useful for [nodes](https://dom.spec.whatwg.org/#concept-node).

[Ranges](https://dom.spec.whatwg.org/#concept-range) are affected by mutations to the [node tree](https://dom.spec.whatwg.org/#concept-node-tree). Such mutations will not invalidate a [range](https://dom.spec.whatwg.org/#concept-range) and will try to ensure that the [range](https://dom.spec.whatwg.org/#concept-range) still represents the same piece of content. Necessarily, a [range](https://dom.spec.whatwg.org/#concept-range) might itself be modified as part of the mutation to the [node tree](https://dom.spec.whatwg.org/#concept-node-tree) when e.g. part of the content it represents is mutated.

See the [insert](https://dom.spec.whatwg.org/#concept-node-insert) and [remove](https://dom.spec.whatwg.org/#concept-node-remove) algorithms, the [normalize()](https://dom.spec.whatwg.org/#dom-node-normalize) method, and the [replace data](https://dom.spec.whatwg.org/#concept-cd-replace) and [split](https://dom.spec.whatwg.org/#concept-text-split) algorithms for the hairy details.

**5.2. Interface** [**Range**](https://dom.spec.whatwg.org/#range)

[[Constructor](https://dom.spec.whatwg.org/#dom-range-range),

Exposed=Window]

interface *Range* {

readonly attribute [Node](https://dom.spec.whatwg.org/#node) [startContainer](https://dom.spec.whatwg.org/#dom-range-startcontainer);

readonly attribute unsigned long [startOffset](https://dom.spec.whatwg.org/#dom-range-startoffset);

readonly attribute [Node](https://dom.spec.whatwg.org/#node) [endContainer](https://dom.spec.whatwg.org/#dom-range-endcontainer);

readonly attribute unsigned long [endOffset](https://dom.spec.whatwg.org/#dom-range-endoffset);

readonly attribute boolean [collapsed](https://dom.spec.whatwg.org/#dom-range-collapsed);

readonly attribute [Node](https://dom.spec.whatwg.org/#node) [commonAncestorContainer](https://dom.spec.whatwg.org/#dom-range-commonancestorcontainer);

void [setStart](https://dom.spec.whatwg.org/#dom-range-setstart)([Node](https://dom.spec.whatwg.org/#node) *node*, unsigned long *offset*);

void [setEnd](https://dom.spec.whatwg.org/#dom-range-setend)([Node](https://dom.spec.whatwg.org/#node) *node*, unsigned long *offset*);

void [setStartBefore](https://dom.spec.whatwg.org/#dom-range-setstartbefore)([Node](https://dom.spec.whatwg.org/#node) *node*);

void [setStartAfter](https://dom.spec.whatwg.org/#dom-range-setstartafter)([Node](https://dom.spec.whatwg.org/#node) *node*);

void [setEndBefore](https://dom.spec.whatwg.org/#dom-range-setendbefore)([Node](https://dom.spec.whatwg.org/#node) *node*);

void [setEndAfter](https://dom.spec.whatwg.org/#dom-range-setendafter)([Node](https://dom.spec.whatwg.org/#node) *node*);

void [collapse](https://dom.spec.whatwg.org/#dom-range-collapse)(optional boolean *toStart* = false);

void [selectNode](https://dom.spec.whatwg.org/#dom-range-selectnode)([Node](https://dom.spec.whatwg.org/#node) *node*);

void [selectNodeContents](https://dom.spec.whatwg.org/#dom-range-selectnodecontents)([Node](https://dom.spec.whatwg.org/#node) *node*);

const unsigned short *START\_TO\_START* = 0;

const unsigned short *START\_TO\_END* = 1;

const unsigned short *END\_TO\_END* = 2;

const unsigned short *END\_TO\_START* = 3;

short [compareBoundaryPoints](https://dom.spec.whatwg.org/#dom-range-compareboundarypoints)(unsigned short *how*, [Range](https://dom.spec.whatwg.org/#range) *sourceRange*);

[CEReactions] void [deleteContents](https://dom.spec.whatwg.org/#dom-range-deletecontents)();

[CEReactions, NewObject] [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [extractContents](https://dom.spec.whatwg.org/#dom-range-extractcontents)();

[CEReactions, NewObject] [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [cloneContents](https://dom.spec.whatwg.org/#dom-range-clonecontents)();

[CEReactions] void [insertNode](https://dom.spec.whatwg.org/#dom-range-insertnode)([Node](https://dom.spec.whatwg.org/#node) *node*);

[CEReactions] void [surroundContents](https://dom.spec.whatwg.org/#dom-range-surroundcontents)([Node](https://dom.spec.whatwg.org/#node) *newParent*);

[NewObject] [Range](https://dom.spec.whatwg.org/#range) [cloneRange](https://dom.spec.whatwg.org/#dom-range-clonerange)();

void [detach](https://dom.spec.whatwg.org/#dom-range-detach)();

boolean [isPointInRange](https://dom.spec.whatwg.org/#dom-range-ispointinrange)([Node](https://dom.spec.whatwg.org/#node) *node*, unsigned long *offset*);

short [comparePoint](https://dom.spec.whatwg.org/#dom-range-comparepoint)([Node](https://dom.spec.whatwg.org/#node) *node*, unsigned long *offset*);

boolean [intersectsNode](https://dom.spec.whatwg.org/#dom-range-intersectsnode)([Node](https://dom.spec.whatwg.org/#node) *node*);

[stringifier](https://dom.spec.whatwg.org/#dom-range-stringifier);

};

[Range](https://dom.spec.whatwg.org/#range) objects are simply known as *ranges*.

A *boundary point* is a ([node](https://dom.spec.whatwg.org/#concept-node), *offset*) tuple, where [offset](https://dom.spec.whatwg.org/#concept-range-bp-offset) is a non-negative integer.

Generally speaking, a [boundary point](https://dom.spec.whatwg.org/#concept-range-bp)’s [offset](https://dom.spec.whatwg.org/#concept-range-bp-offset) will be between zero and the [boundary point](https://dom.spec.whatwg.org/#concept-range-bp)’s [node](https://dom.spec.whatwg.org/#concept-node) [length](https://dom.spec.whatwg.org/#concept-node-length), inclusive. Algorithms that modify a [tree](https://dom.spec.whatwg.org/#concept-tree) (in particular the [insert](https://dom.spec.whatwg.org/#concept-node-insert), [remove](https://dom.spec.whatwg.org/#concept-node-remove), [replace data](https://dom.spec.whatwg.org/#concept-cd-replace), and [split](https://dom.spec.whatwg.org/#concept-text-split) algorithms) also modify [ranges](https://dom.spec.whatwg.org/#concept-range) associated with that [tree](https://dom.spec.whatwg.org/#concept-tree).

If the two [nodes](https://dom.spec.whatwg.org/#concept-node) of [boundary points](https://dom.spec.whatwg.org/#concept-range-bp) (*node A*, *offset A*) and (*node B*, *offset B*) have the same [root](https://dom.spec.whatwg.org/#concept-tree-root), the *position* of the first relative to the second is either *before*, *equal*, or *after*, as returned by the following algorithm:

1. If *node A* is the same as *node B*, return [equal](https://dom.spec.whatwg.org/#concept-range-bp-equal) if *offset A* is the same as *offset B*, [before](https://dom.spec.whatwg.org/#concept-range-bp-before) if *offset A* is less than *offset B*, and [after](https://dom.spec.whatwg.org/#concept-range-bp-after) if *offset A* is greater than *offset B*.
2. If *node A* is [following](https://dom.spec.whatwg.org/#concept-tree-following) *node B*, compute the [position](https://dom.spec.whatwg.org/#concept-range-bp-position) of (*node B*, *offset B*) relative to (*node A*, *offset A*). If it is [before](https://dom.spec.whatwg.org/#concept-range-bp-before), return [after](https://dom.spec.whatwg.org/#concept-range-bp-after). If it is [after](https://dom.spec.whatwg.org/#concept-range-bp-after), return [before](https://dom.spec.whatwg.org/#concept-range-bp-before).
3. If *node A* is an [ancestor](https://dom.spec.whatwg.org/#concept-tree-ancestor) of *node B*:
   1. Let *child* equal *node B*.
   2. While *child* is not a [child](https://dom.spec.whatwg.org/#concept-tree-child) of *node A*, set *child* to its [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
   3. If the [index](https://dom.spec.whatwg.org/#concept-tree-index) of *child* is less than *offset A*, return [after](https://dom.spec.whatwg.org/#concept-range-bp-after).
4. Return [before](https://dom.spec.whatwg.org/#concept-range-bp-before).

Each [range](https://dom.spec.whatwg.org/#concept-range) has two associated [boundary points](https://dom.spec.whatwg.org/#concept-range-bp) — a *start* and *end*.

For convenience, *start node* is [start](https://dom.spec.whatwg.org/#concept-range-start)’s [node](https://dom.spec.whatwg.org/#concept-node), *start offset* is [start](https://dom.spec.whatwg.org/#concept-range-start)’s [offset](https://dom.spec.whatwg.org/#concept-range-bp-offset), *end node* is [end](https://dom.spec.whatwg.org/#concept-range-end)’s [node](https://dom.spec.whatwg.org/#concept-node), and *end offset* is [end](https://dom.spec.whatwg.org/#concept-range-end)’s [offset](https://dom.spec.whatwg.org/#concept-range-bp-offset).

The *root* of a [range](https://dom.spec.whatwg.org/#concept-range) is the [root](https://dom.spec.whatwg.org/#concept-tree-root) of its [start node](https://dom.spec.whatwg.org/#concept-range-start-node).

A [node](https://dom.spec.whatwg.org/#concept-node) *node* is *contained* in a [range](https://dom.spec.whatwg.org/#concept-range) *range* if *node*’s [root](https://dom.spec.whatwg.org/#concept-tree-root) is the same as *range*’s [root](https://dom.spec.whatwg.org/#concept-range-root), and (*node*, 0) is [after](https://dom.spec.whatwg.org/#concept-range-bp-after) *range*’s [start](https://dom.spec.whatwg.org/#concept-range-start), and (*node*, [length](https://dom.spec.whatwg.org/#concept-node-length) of *node*) is [before](https://dom.spec.whatwg.org/#concept-range-bp-before) *range*’s [end](https://dom.spec.whatwg.org/#concept-range-end).

A [node](https://dom.spec.whatwg.org/#concept-node) is *partially contained* in a [range](https://dom.spec.whatwg.org/#concept-range) if it is an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of the [range](https://dom.spec.whatwg.org/#concept-range)’s [start node](https://dom.spec.whatwg.org/#concept-range-start-node) but not its [end node](https://dom.spec.whatwg.org/#concept-range-end-node), or vice versa.

Some facts to better understand these definitions:

* The content that one would think of as being within the [range](https://dom.spec.whatwg.org/#concept-range) consists of all [contained](https://dom.spec.whatwg.org/#contained) [nodes](https://dom.spec.whatwg.org/#concept-node), plus possibly some of the contents of the [start node](https://dom.spec.whatwg.org/#concept-range-start-node) and [end node](https://dom.spec.whatwg.org/#concept-range-end-node) if those are [Text](https://dom.spec.whatwg.org/#text), [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction), or [Comment](https://dom.spec.whatwg.org/#comment) [nodes](https://dom.spec.whatwg.org/#concept-node).
* The [nodes](https://dom.spec.whatwg.org/#concept-node) that are contained in a [range](https://dom.spec.whatwg.org/#concept-range) will generally not be contiguous, because the [parent](https://dom.spec.whatwg.org/#concept-tree-parent) of a [contained](https://dom.spec.whatwg.org/#contained) [node](https://dom.spec.whatwg.org/#concept-node) will not always be [contained](https://dom.spec.whatwg.org/#contained).
* However, the [descendants](https://dom.spec.whatwg.org/#concept-tree-descendant) of a [contained](https://dom.spec.whatwg.org/#contained) [node](https://dom.spec.whatwg.org/#concept-node) are [contained](https://dom.spec.whatwg.org/#contained), and if two [siblings](https://dom.spec.whatwg.org/#concept-tree-sibling) are [contained](https://dom.spec.whatwg.org/#contained), so are any [siblings](https://dom.spec.whatwg.org/#concept-tree-sibling) that lie between them.
* The [start node](https://dom.spec.whatwg.org/#concept-range-start-node) and [end node](https://dom.spec.whatwg.org/#concept-range-end-node) of a [range](https://dom.spec.whatwg.org/#concept-range) are never [contained](https://dom.spec.whatwg.org/#contained) within it.
* The first [contained](https://dom.spec.whatwg.org/#contained) [node](https://dom.spec.whatwg.org/#concept-node) (if there are any) will always be after the [start node](https://dom.spec.whatwg.org/#concept-range-start-node), and the last [contained](https://dom.spec.whatwg.org/#contained) [node](https://dom.spec.whatwg.org/#concept-node) will always be equal to or before the [end node](https://dom.spec.whatwg.org/#concept-range-end-node)’s last [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant).
* There exists a partially contained [node](https://dom.spec.whatwg.org/#concept-node) if and only if the [start node](https://dom.spec.whatwg.org/#concept-range-start-node) and [end node](https://dom.spec.whatwg.org/#concept-range-end-node) are different.
* The [commonAncestorContainer](https://dom.spec.whatwg.org/#dom-range-commonancestorcontainer) attribute value is neither [contained](https://dom.spec.whatwg.org/#contained) nor [partially contained](https://dom.spec.whatwg.org/#partially-contained).
* If the [start node](https://dom.spec.whatwg.org/#concept-range-start-node) is an [ancestor](https://dom.spec.whatwg.org/#concept-tree-ancestor) of the [end node](https://dom.spec.whatwg.org/#concept-range-end-node), the common [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) will be the [start node](https://dom.spec.whatwg.org/#concept-range-start-node). Exactly one of its [children](https://dom.spec.whatwg.org/#concept-tree-child) will be [partially contained](https://dom.spec.whatwg.org/#partially-contained), and a [child](https://dom.spec.whatwg.org/#concept-tree-child) will be [contained](https://dom.spec.whatwg.org/#contained) if and only if it [precedes](https://dom.spec.whatwg.org/#concept-tree-preceding) the [partially contained](https://dom.spec.whatwg.org/#partially-contained) [child](https://dom.spec.whatwg.org/#concept-tree-child). If the [end node](https://dom.spec.whatwg.org/#concept-range-end-node) is an [ancestor](https://dom.spec.whatwg.org/#concept-tree-ancestor) of the [start node](https://dom.spec.whatwg.org/#concept-range-start-node), the opposite holds.
* If the [start node](https://dom.spec.whatwg.org/#concept-range-start-node) is not an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of the [end node](https://dom.spec.whatwg.org/#concept-range-end-node), nor vice versa, the common [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) will be distinct from both of them. Exactly two of its [children](https://dom.spec.whatwg.org/#concept-tree-child) will be [partially contained](https://dom.spec.whatwg.org/#partially-contained), and a [child](https://dom.spec.whatwg.org/#concept-tree-child) will be contained if and only if it lies between those two.

*range* = new [Range()](https://dom.spec.whatwg.org/#dom-range-range)

Returns a new [range](https://dom.spec.whatwg.org/#concept-range).

The *Range()* constructor must return a new [range](https://dom.spec.whatwg.org/#concept-range) with (global object’s associated [document](https://dom.spec.whatwg.org/#concept-document), 0) as its [start](https://dom.spec.whatwg.org/#concept-range-start) and [end](https://dom.spec.whatwg.org/#concept-range-end).

*node* = *range* . [startContainer](https://dom.spec.whatwg.org/#dom-range-startcontainer)

Returns *range*’s [start node](https://dom.spec.whatwg.org/#concept-range-start-node).

*offset* = *range* . [startOffset](https://dom.spec.whatwg.org/#dom-range-startoffset)

Returns *range*’s [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset).

*node* = *range* . [endContainer](https://dom.spec.whatwg.org/#dom-range-endcontainer)

Returns *range*’s [end node](https://dom.spec.whatwg.org/#concept-range-end-node).

*offset* = *range* . [endOffset](https://dom.spec.whatwg.org/#dom-range-endoffset)

Returns *range*’s [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset).

*collapsed* = *range* . [collapsed](https://dom.spec.whatwg.org/#dom-range-collapsed)

Returns true if *range*’s [start](https://dom.spec.whatwg.org/#concept-range-start) and [end](https://dom.spec.whatwg.org/#concept-range-end) are the same, and false otherwise.

*container* = *range* . [commonAncestorContainer](https://dom.spec.whatwg.org/#dom-range-commonancestorcontainer)

Returns the [node](https://dom.spec.whatwg.org/#concept-node), furthest away from the [document](https://dom.spec.whatwg.org/#concept-document), that is an [ancestor](https://dom.spec.whatwg.org/#concept-tree-ancestor) of both *range*’s [start node](https://dom.spec.whatwg.org/#concept-range-start-node) and [end node](https://dom.spec.whatwg.org/#concept-range-end-node).

The *startContainer* attribute must return the [start node](https://dom.spec.whatwg.org/#concept-range-start-node).

The *startOffset* attribute must return the [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset).

The *endContainer* attribute must return the [end node](https://dom.spec.whatwg.org/#concept-range-end-node).

The *endOffset* attribute must return the [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset).

The *collapsed* attribute must return true if [start](https://dom.spec.whatwg.org/#concept-range-start) is the same as [end](https://dom.spec.whatwg.org/#concept-range-end), and false otherwise.

The *commonAncestorContainer* attribute must run these steps:

1. Let *container* be [start node](https://dom.spec.whatwg.org/#concept-range-start-node).
2. While *container* is not an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of [end node](https://dom.spec.whatwg.org/#concept-range-end-node), let *container* be *container*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
3. Return *container*.

To *set the start or end* of a *range* to a [boundary point](https://dom.spec.whatwg.org/#concept-range-bp) (*node*, *offset*), run these steps:

1. If *node* is a [doctype](https://dom.spec.whatwg.org/#concept-doctype), then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidNodeTypeError](https://heycam.github.io/webidl/#invalidnodetypeerror).
2. If *offset* is greater than *node*’s [length](https://dom.spec.whatwg.org/#concept-node-length), then [throw](https://heycam.github.io/webidl/#dfn-throw) an [IndexSizeError](https://heycam.github.io/webidl/#indexsizeerror).
3. Let *bp* be the [boundary point](https://dom.spec.whatwg.org/#concept-range-bp) (*node*, *offset*).
4. If these steps were invoked as "set the start"
   1. If *bp* is [after](https://dom.spec.whatwg.org/#concept-range-bp-after) the *range*’s [end](https://dom.spec.whatwg.org/#concept-range-end), or if *range*’s [root](https://dom.spec.whatwg.org/#concept-range-root) is not equal to *node*’s [root](https://dom.spec.whatwg.org/#concept-tree-root), set *range*’s [end](https://dom.spec.whatwg.org/#concept-range-end) to *bp*.
   2. Set *range*’s [start](https://dom.spec.whatwg.org/#concept-range-start) to *bp*.

If these steps were invoked as "set the end"

* 1. If *bp* is [before](https://dom.spec.whatwg.org/#concept-range-bp-before) the *range*’s [start](https://dom.spec.whatwg.org/#concept-range-start), or if *range*’s [root](https://dom.spec.whatwg.org/#concept-range-root) is not equal to *node*’s [root](https://dom.spec.whatwg.org/#concept-tree-root), set *range*’s [start](https://dom.spec.whatwg.org/#concept-range-start) to *bp*.
  2. Set *range*’s [end](https://dom.spec.whatwg.org/#concept-range-end) to *bp*.

The *setStart(node, offset)* method must [set the start](https://dom.spec.whatwg.org/#concept-range-bp-set) of the [context object](https://dom.spec.whatwg.org/#context-object) to [boundary point](https://dom.spec.whatwg.org/#concept-range-bp) (*node*, *offset*).

The *setEnd(node, offset)* method must [set the end](https://dom.spec.whatwg.org/#concept-range-bp-set) of the [context object](https://dom.spec.whatwg.org/#context-object) to [boundary point](https://dom.spec.whatwg.org/#concept-range-bp) (*node*, *offset*).

The *setStartBefore(node)* method must run these steps:

1. Let *parent* be *node*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
2. If *parent* is null, then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidNodeTypeError](https://heycam.github.io/webidl/#invalidnodetypeerror).
3. [Set the start](https://dom.spec.whatwg.org/#concept-range-bp-set) of the [context object](https://dom.spec.whatwg.org/#context-object) to [boundary point](https://dom.spec.whatwg.org/#concept-range-bp) (*parent*, *node*’s [index](https://dom.spec.whatwg.org/#concept-tree-index)).

The *setStartAfter(node)* method must run these steps:

1. Let *parent* be *node*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
2. If *parent* is null, then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidNodeTypeError](https://heycam.github.io/webidl/#invalidnodetypeerror).
3. [Set the start](https://dom.spec.whatwg.org/#concept-range-bp-set) of the [context object](https://dom.spec.whatwg.org/#context-object) to [boundary point](https://dom.spec.whatwg.org/#concept-range-bp) (*parent*, *node*’s [index](https://dom.spec.whatwg.org/#concept-tree-index) plus one).

The *setEndBefore(node)* method must run these steps:

1. Let *parent* be *node*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
2. If *parent* is null, then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidNodeTypeError](https://heycam.github.io/webidl/#invalidnodetypeerror).
3. [Set the end](https://dom.spec.whatwg.org/#concept-range-bp-set) of the [context object](https://dom.spec.whatwg.org/#context-object) to [boundary point](https://dom.spec.whatwg.org/#concept-range-bp) (*parent*, *node*’s [index](https://dom.spec.whatwg.org/#concept-tree-index)).

The *setEndAfter(node)* method must run these steps:

1. Let *parent* be *node*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
2. If *parent* is null, then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidNodeTypeError](https://heycam.github.io/webidl/#invalidnodetypeerror).
3. [Set the end](https://dom.spec.whatwg.org/#concept-range-bp-set) of the [context object](https://dom.spec.whatwg.org/#context-object) to [boundary point](https://dom.spec.whatwg.org/#concept-range-bp) (*parent*, *node*’s [index](https://dom.spec.whatwg.org/#concept-tree-index) plus one).

The *collapse(toStart)* method, when invoked, must if *toStart* is true, set [end](https://dom.spec.whatwg.org/#concept-range-end) to [start](https://dom.spec.whatwg.org/#concept-range-start), and set [start](https://dom.spec.whatwg.org/#concept-range-start) to [end](https://dom.spec.whatwg.org/#concept-range-end) otherwise.

To *select* a [node](https://dom.spec.whatwg.org/#concept-node) *node* within a [range](https://dom.spec.whatwg.org/#concept-range) *range*, run these steps:

1. Let *parent* be *node*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
2. If *parent* is null, [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidNodeTypeError](https://heycam.github.io/webidl/#invalidnodetypeerror).
3. Let *index* be *node*’s [index](https://dom.spec.whatwg.org/#concept-tree-index).
4. Set *range*’s [start](https://dom.spec.whatwg.org/#concept-range-start) to [boundary point](https://dom.spec.whatwg.org/#concept-range-bp) (*parent*, *index*).
5. Set *range*’s [end](https://dom.spec.whatwg.org/#concept-range-end) to [boundary point](https://dom.spec.whatwg.org/#concept-range-bp) (*parent*, *index* plus one).

The *selectNode(node)* method must [select](https://dom.spec.whatwg.org/#concept-range-select) *node* within [context object](https://dom.spec.whatwg.org/#context-object).

The *selectNodeContents(node)* method must run these steps:

1. If *node* is a [doctype](https://dom.spec.whatwg.org/#concept-doctype), [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidNodeTypeError](https://heycam.github.io/webidl/#invalidnodetypeerror).
2. Let *length* be the [length](https://dom.spec.whatwg.org/#concept-node-length) of *node*.
3. Set [start](https://dom.spec.whatwg.org/#concept-range-start) to the [boundary point](https://dom.spec.whatwg.org/#concept-range-bp) (*node*, 0).
4. Set [end](https://dom.spec.whatwg.org/#concept-range-end) to the [boundary point](https://dom.spec.whatwg.org/#concept-range-bp) (*node*, *length*).

The *compareBoundaryPoints(how, sourceRange)* method must run these steps:

1. If *how* is not one of
   * [START\_TO\_START](https://dom.spec.whatwg.org/#dom-range-start_to_start),
   * [START\_TO\_END](https://dom.spec.whatwg.org/#dom-range-start_to_end),
   * [END\_TO\_END](https://dom.spec.whatwg.org/#dom-range-end_to_end), and
   * [END\_TO\_START](https://dom.spec.whatwg.org/#dom-range-end_to_start),

then [throw](https://heycam.github.io/webidl/#dfn-throw) a [NotSupportedError](https://heycam.github.io/webidl/#notsupportederror).

1. If [context object](https://dom.spec.whatwg.org/#context-object)’s [root](https://dom.spec.whatwg.org/#concept-range-root) is not the same as *sourceRange*’s [root](https://dom.spec.whatwg.org/#concept-range-root), then [throw](https://heycam.github.io/webidl/#dfn-throw) a [WrongDocumentError](https://heycam.github.io/webidl/#wrongdocumenterror).
2. If *how* is:

[START\_TO\_START](https://dom.spec.whatwg.org/#dom-range-start_to_start):

Let *this point* be the [context object](https://dom.spec.whatwg.org/#context-object)’s [start](https://dom.spec.whatwg.org/#concept-range-start). Let *other point* be *sourceRange*’s [start](https://dom.spec.whatwg.org/#concept-range-start).

[START\_TO\_END](https://dom.spec.whatwg.org/#dom-range-start_to_end):

Let *this point* be the [context object](https://dom.spec.whatwg.org/#context-object)’s [end](https://dom.spec.whatwg.org/#concept-range-end). Let *other point* be *sourceRange*’s [start](https://dom.spec.whatwg.org/#concept-range-start).

[END\_TO\_END](https://dom.spec.whatwg.org/#dom-range-end_to_end):

Let *this point* be the [context object](https://dom.spec.whatwg.org/#context-object)’s [end](https://dom.spec.whatwg.org/#concept-range-end). Let *other point* be *sourceRange*’s [end](https://dom.spec.whatwg.org/#concept-range-end).

[END\_TO\_START](https://dom.spec.whatwg.org/#dom-range-end_to_start):

Let *this point* be the [context object](https://dom.spec.whatwg.org/#context-object)’s [start](https://dom.spec.whatwg.org/#concept-range-start). Let *other point* be *sourceRange*’s [end](https://dom.spec.whatwg.org/#concept-range-end).

1. If the [position](https://dom.spec.whatwg.org/#concept-range-bp-position) of *this point* relative to *other point* is

[before](https://dom.spec.whatwg.org/#concept-range-bp-before)

Return −1.

[equal](https://dom.spec.whatwg.org/#concept-range-bp-equal)

Return 0.

[after](https://dom.spec.whatwg.org/#concept-range-bp-after)

Return 1.

The *deleteContents()* method, when invoked, must run these steps:

1. If [start](https://dom.spec.whatwg.org/#concept-range-start) is [end](https://dom.spec.whatwg.org/#concept-range-end), terminate these steps.
2. Let *original start node*, *original start offset*, *original end node*, and *original end offset* be the [context object](https://dom.spec.whatwg.org/#context-object)’s [start node](https://dom.spec.whatwg.org/#concept-range-start-node), [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset), [end node](https://dom.spec.whatwg.org/#concept-range-end-node), and [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset), respectively.
3. If *original start node* and *original end node* are the same, and they are a [Text](https://dom.spec.whatwg.org/#text), [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction), or [Comment](https://dom.spec.whatwg.org/#comment) [node](https://dom.spec.whatwg.org/#concept-node), [replace data](https://dom.spec.whatwg.org/#concept-cd-replace) with node *original start node*, offset *original start offset*, count *original end offset* minus *original start offset*, and data the empty string, and then terminate these steps.
4. Let *nodes to remove* be a list of all the [nodes](https://dom.spec.whatwg.org/#concept-node) that are [contained](https://dom.spec.whatwg.org/#contained) in the [context object](https://dom.spec.whatwg.org/#context-object), in [tree order](https://dom.spec.whatwg.org/#concept-tree-order), omitting any [node](https://dom.spec.whatwg.org/#concept-node) whose [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is also [contained](https://dom.spec.whatwg.org/#contained) in the [context object](https://dom.spec.whatwg.org/#context-object).
5. If *original start node* is an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of *original end node*, set *new node* to *original start node* and *new offset* to *original start offset*.
6. Otherwise:
   1. Let *reference node* equal *original start node*.
   2. While *reference node*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is not null and is not an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of *original end node*, set *reference node* to its [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
   3. Set *new node* to the [parent](https://dom.spec.whatwg.org/#concept-tree-parent) of *reference node*, and *new offset* to one plus the [index](https://dom.spec.whatwg.org/#concept-tree-index) of *reference node*.

If *reference node*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) were null, it would be the [root](https://dom.spec.whatwg.org/#concept-range-root) of the [context object](https://dom.spec.whatwg.org/#context-object), so would be an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of *original end node*, and we could not reach this point.

1. If *original start node* is a [Text](https://dom.spec.whatwg.org/#text), [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction), or [Comment](https://dom.spec.whatwg.org/#comment) [node](https://dom.spec.whatwg.org/#concept-node), [replace data](https://dom.spec.whatwg.org/#concept-cd-replace) with node *original start node*, offset *original start offset*, count *original start node*’s [length](https://dom.spec.whatwg.org/#concept-node-length) minus *original start offset*, data the empty string.
2. For each *node* in *nodes to remove*, in [tree order](https://dom.spec.whatwg.org/#concept-tree-order), [remove](https://dom.spec.whatwg.org/#concept-node-remove) *node* from its [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
3. If *original end node* is a [Text](https://dom.spec.whatwg.org/#text), [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction), or [Comment](https://dom.spec.whatwg.org/#comment) [node](https://dom.spec.whatwg.org/#concept-node), [replace data](https://dom.spec.whatwg.org/#concept-cd-replace) with node *original end node*, offset 0, count *original end offset* and data the empty string.
4. Set [start](https://dom.spec.whatwg.org/#concept-range-start) and [end](https://dom.spec.whatwg.org/#concept-range-end) to (*new node*, *new offset*).

To *extract* a [range](https://dom.spec.whatwg.org/#concept-range) *range*, run these steps:

1. Let *fragment* be a new [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [node](https://dom.spec.whatwg.org/#concept-node) whose [node document](https://dom.spec.whatwg.org/#concept-node-document) is *range*’s [start node](https://dom.spec.whatwg.org/#concept-range-start-node)’s [node document](https://dom.spec.whatwg.org/#concept-node-document).
2. If *range*’s [start](https://dom.spec.whatwg.org/#concept-range-start) is its [end](https://dom.spec.whatwg.org/#concept-range-end), return *fragment*.
3. Let *original start node*, *original start offset*, *original end node*, and *original end offset* be *range*’s [start node](https://dom.spec.whatwg.org/#concept-range-start-node), [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset), [end node](https://dom.spec.whatwg.org/#concept-range-end-node), and [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset), respectively.
4. If *original start node* is *original end node*, and they are a [Text](https://dom.spec.whatwg.org/#text), [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction), or [Comment](https://dom.spec.whatwg.org/#comment) [node](https://dom.spec.whatwg.org/#concept-node):
   1. Let *clone* be a [clone](https://dom.spec.whatwg.org/#concept-node-clone) of *original start node*.
   2. Set the [data](https://dom.spec.whatwg.org/#concept-cd-data) of *clone* to the result of [substringing data](https://dom.spec.whatwg.org/#concept-cd-substring) with node *original start node*, offset *original start offset*, and count *original end offset* minus *original start offset*.
   3. [Append](https://dom.spec.whatwg.org/#concept-node-append) *clone* to *fragment*.
   4. [Replace data](https://dom.spec.whatwg.org/#concept-cd-replace) with node *original start node*, offset *original start offset*, count *original end offset* minus *original start offset*, and data the empty string.
   5. Return *fragment*.
5. Let *common ancestor* be *original start node*.
6. While *common ancestor* is not an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of *original end node*, set *common ancestor* to its own [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
7. Let *first partially contained child* be null.
8. If *original start node* is *not* an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of *original end node*, set *first partially contained child* to the first [child](https://dom.spec.whatwg.org/#concept-tree-child) of *common ancestor* that is [partially contained](https://dom.spec.whatwg.org/#partially-contained) in *range*.
9. Let *last partially contained child* be null.
10. If *original end node* is *not* an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of *original start node*, set *last partially contained child* to the last [child](https://dom.spec.whatwg.org/#concept-tree-child) of *common ancestor* that is [partially contained](https://dom.spec.whatwg.org/#partially-contained) in *range*.

These variable assignments do actually always make sense. For instance, if *original start node* is not an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of *original end node*, *original start node* is itself [partially contained](https://dom.spec.whatwg.org/#partially-contained) in *range*, and so are all its [ancestors](https://dom.spec.whatwg.org/#concept-tree-ancestor) up until a [child](https://dom.spec.whatwg.org/#concept-tree-child) of *common ancestor*. *common ancestor* cannot be *original start node*, because it has to be an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of *original end node*. The other case is similar. Also, notice that the two [children](https://dom.spec.whatwg.org/#concept-tree-child) will never be equal if both are defined.

1. Let *contained children* be a list of all [children](https://dom.spec.whatwg.org/#concept-tree-child) of *common ancestor* that are [contained](https://dom.spec.whatwg.org/#contained) in *range*, in [tree order](https://dom.spec.whatwg.org/#concept-tree-order).
2. If any member of *contained children* is a [doctype](https://dom.spec.whatwg.org/#concept-doctype), then [throw](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror).

We do not have to worry about the first or last partially contained node, because a [doctype](https://dom.spec.whatwg.org/#concept-doctype) can never be partially contained. It cannot be a boundary point of a range, and it cannot be the ancestor of anything.

1. If *original start node* is an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of *original end node*, set *new node* to *original start node* and *new offset* to *original start offset*.
2. Otherwise:
   1. Let *reference node* equal *original start node*.
   2. While *reference node*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is not null and is not an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of *original end node*, set *reference node* to its [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
   3. Set *new node* to the [parent](https://dom.spec.whatwg.org/#concept-tree-parent) of *reference node*, and *new offset* to one plus *reference node*’s [index](https://dom.spec.whatwg.org/#concept-tree-index).

If *reference node*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is null, it would be the [root](https://dom.spec.whatwg.org/#concept-range-root) of *range*, so would be an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of *original end node*, and we could not reach this point.

1. If *first partially contained child* is a [Text](https://dom.spec.whatwg.org/#text), [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction), or [Comment](https://dom.spec.whatwg.org/#comment) [node](https://dom.spec.whatwg.org/#concept-node):

In this case, *first partially contained child* is *original start node*.

* 1. Let *clone* be a [clone](https://dom.spec.whatwg.org/#concept-node-clone) of *original start node*.
  2. Set the [data](https://dom.spec.whatwg.org/#concept-cd-data) of *clone* to the result of [substringing data](https://dom.spec.whatwg.org/#concept-cd-substring) with node *original start node*, offset *original start offset*, and count *original start node*’s [length](https://dom.spec.whatwg.org/#concept-node-length) minus *original start offset*.
  3. [Append](https://dom.spec.whatwg.org/#concept-node-append) *clone* to *fragment*.
  4. [Replace data](https://dom.spec.whatwg.org/#concept-cd-replace) with node *original start node*, offset *original start offset*, count *original start node*’s [length](https://dom.spec.whatwg.org/#concept-node-length) minus *original start offset*, and data the empty string.

1. Otherwise, if *first partially contained child* is not null:
   1. Let *clone* be a [clone](https://dom.spec.whatwg.org/#concept-node-clone) of *first partially contained child*.
   2. [Append](https://dom.spec.whatwg.org/#concept-node-append) *clone* to *fragment*.
   3. Let *subrange* be a new [range](https://dom.spec.whatwg.org/#concept-range) whose [start](https://dom.spec.whatwg.org/#concept-range-start) is (*original start node*, *original start offset*) and whose [end](https://dom.spec.whatwg.org/#concept-range-end) is (*first partially contained child*, *first partially contained child*’s [length](https://dom.spec.whatwg.org/#concept-node-length)).
   4. Let *subfragment* be the result of [extracting](https://dom.spec.whatwg.org/#concept-range-extract) *subrange*.
   5. [Append](https://dom.spec.whatwg.org/#concept-node-append) *subfragment* to *clone*.
2. For each *contained child* in *contained children*, [append](https://dom.spec.whatwg.org/#concept-node-append) *contained child* to *fragment*.
3. If *last partially contained child* is a [Text](https://dom.spec.whatwg.org/#text), [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction), or [Comment](https://dom.spec.whatwg.org/#comment) [node](https://dom.spec.whatwg.org/#concept-node):

In this case, *last partially contained child* is *original end node*.

* 1. Let *clone* be a [clone](https://dom.spec.whatwg.org/#concept-node-clone) of *original end node*.
  2. Set the [data](https://dom.spec.whatwg.org/#concept-cd-data) of *clone* to the result of [substringing data](https://dom.spec.whatwg.org/#concept-cd-substring) with node *original end node*, offset 0, and count *original end offset*.
  3. [Append](https://dom.spec.whatwg.org/#concept-node-append) *clone* to *fragment*.
  4. [Replace data](https://dom.spec.whatwg.org/#concept-cd-replace) with node *original end node*, offset 0, count *original end offset*, and data the empty string.

1. Otherwise, if *last partially contained child* is not null:
   1. Let *clone* be a [clone](https://dom.spec.whatwg.org/#concept-node-clone) of *last partially contained child*.
   2. [Append](https://dom.spec.whatwg.org/#concept-node-append) *clone* to *fragment*.
   3. Let *subrange* be a new [range](https://dom.spec.whatwg.org/#concept-range) whose [start](https://dom.spec.whatwg.org/#concept-range-start) is (*last partially contained child*, 0) and whose [end](https://dom.spec.whatwg.org/#concept-range-end) is (*original end node*, *original end offset*).
   4. Let *subfragment* be the result of [extracting](https://dom.spec.whatwg.org/#concept-range-extract) *subrange*.
   5. [Append](https://dom.spec.whatwg.org/#concept-node-append) *subfragment* to *clone*.
2. Set *range*’s [start](https://dom.spec.whatwg.org/#concept-range-start) and [end](https://dom.spec.whatwg.org/#concept-range-end) to (*new node*, *new offset*).
3. Return *fragment*.

The *extractContents()* method must return the result of [extracting](https://dom.spec.whatwg.org/#concept-range-extract) [context object](https://dom.spec.whatwg.org/#context-object).

To *clone the contents* of a [range](https://dom.spec.whatwg.org/#concept-range) *range*, run these steps:

1. Let *fragment* be a new [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [node](https://dom.spec.whatwg.org/#concept-node) whose [node document](https://dom.spec.whatwg.org/#concept-node-document) is *range*’s [start node](https://dom.spec.whatwg.org/#concept-range-start-node)’s [node document](https://dom.spec.whatwg.org/#concept-node-document).
2. If *range*’s [start](https://dom.spec.whatwg.org/#concept-range-start) is its [end](https://dom.spec.whatwg.org/#concept-range-end), return *fragment*.
3. Let *original start node*, *original start offset*, *original end node*, and *original end offset* be *range*’s [start node](https://dom.spec.whatwg.org/#concept-range-start-node), [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset), [end node](https://dom.spec.whatwg.org/#concept-range-end-node), and [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset), respectively.
4. If *original start node* is *original end node*, and they are a [Text](https://dom.spec.whatwg.org/#text), [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction), or [Comment](https://dom.spec.whatwg.org/#comment) [node](https://dom.spec.whatwg.org/#concept-node):
   1. Let *clone* be a [clone](https://dom.spec.whatwg.org/#concept-node-clone) of *original start node*.
   2. Set the [data](https://dom.spec.whatwg.org/#concept-cd-data) of *clone* to the result of [substringing data](https://dom.spec.whatwg.org/#concept-cd-substring) with node *original start node*, offset *original start offset*, and count *original end offset* minus *original start offset*.
   3. [Append](https://dom.spec.whatwg.org/#concept-node-append) *clone* to *fragment*.
   4. Return *fragment*.
5. Let *common ancestor* be *original start node*.
6. While *common ancestor* is not an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of *original end node*, set *common ancestor* to its own [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
7. Let *first partially contained child* be null.
8. If *original start node* is *not* an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of *original end node*, set *first partially contained child* to the first [child](https://dom.spec.whatwg.org/#concept-tree-child) of *common ancestor* that is [partially contained](https://dom.spec.whatwg.org/#partially-contained) in *range*.
9. Let *last partially contained child* be null.
10. If *original end node* is *not* an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of *original start node*, set *last partially contained child* to the last [child](https://dom.spec.whatwg.org/#concept-tree-child) of *common ancestor* that is [partially contained](https://dom.spec.whatwg.org/#partially-contained) in *range*.

These variable assignments do actually always make sense. For instance, if *original start node* is not an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of *original end node*, *original start node* is itself [partially contained](https://dom.spec.whatwg.org/#partially-contained) in *range*, and so are all its [ancestors](https://dom.spec.whatwg.org/#concept-tree-ancestor) up until a [child](https://dom.spec.whatwg.org/#concept-tree-child) of *common ancestor*. *common ancestor* cannot be *original start node*, because it has to be an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of *original end node*. The other case is similar. Also, notice that the two [children](https://dom.spec.whatwg.org/#concept-tree-child) will never be equal if both are defined.

1. Let *contained children* be a list of all [children](https://dom.spec.whatwg.org/#concept-tree-child) of *common ancestor* that are [contained](https://dom.spec.whatwg.org/#contained) in *range*, in [tree order](https://dom.spec.whatwg.org/#concept-tree-order).
2. If any member of *contained children* is a [doctype](https://dom.spec.whatwg.org/#concept-doctype), then [throw](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror).

We do not have to worry about the first or last partially contained node, because a [doctype](https://dom.spec.whatwg.org/#concept-doctype) can never be partially contained. It cannot be a boundary point of a range, and it cannot be the ancestor of anything.

1. If *first partially contained child* is a [Text](https://dom.spec.whatwg.org/#text), [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction), or [Comment](https://dom.spec.whatwg.org/#comment) [node](https://dom.spec.whatwg.org/#concept-node):

In this case, *first partially contained child* is *original start node*.

* 1. Let *clone* be a [clone](https://dom.spec.whatwg.org/#concept-node-clone) of *original start node*.
  2. Set the [data](https://dom.spec.whatwg.org/#concept-cd-data) of *clone* to the result of [substringing data](https://dom.spec.whatwg.org/#concept-cd-substring) with node *original start node*, offset *original start offset*, and count *original start node*’s [length](https://dom.spec.whatwg.org/#concept-node-length) minus *original start offset*.
  3. [Append](https://dom.spec.whatwg.org/#concept-node-append) *clone* to *fragment*.

1. Otherwise, if *first partially contained child* is not null:
   1. Let *clone* be a [clone](https://dom.spec.whatwg.org/#concept-node-clone) of *first partially contained child*.
   2. [Append](https://dom.spec.whatwg.org/#concept-node-append) *clone* to *fragment*.
   3. Let *subrange* be a new [range](https://dom.spec.whatwg.org/#concept-range) whose [start](https://dom.spec.whatwg.org/#concept-range-start) is (*original start node*, *original start offset*) and whose [end](https://dom.spec.whatwg.org/#concept-range-end) is (*first partially contained child*, *first partially contained child*’s [length](https://dom.spec.whatwg.org/#concept-node-length)).
   4. Let *subfragment* be the result of [cloning the contents](https://dom.spec.whatwg.org/#concept-range-clone) of *subrange*.
   5. [Append](https://dom.spec.whatwg.org/#concept-node-append) *subfragment* to *clone*.
2. For each *contained child* in *contained children*:
   1. Let *clone* be a [clone](https://dom.spec.whatwg.org/#concept-node-clone) of *contained child* with the *clone children flag* set.
   2. [Append](https://dom.spec.whatwg.org/#concept-node-append) *clone* to *fragment*.
3. If *last partially contained child* is a [Text](https://dom.spec.whatwg.org/#text), [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction), or [Comment](https://dom.spec.whatwg.org/#comment) [node](https://dom.spec.whatwg.org/#concept-node):

In this case, *last partially contained child* is *original end node*.

* 1. Let *clone* be a [clone](https://dom.spec.whatwg.org/#concept-node-clone) of *original end node*.
  2. Set the [data](https://dom.spec.whatwg.org/#concept-cd-data) of *clone* to the result of [substringing data](https://dom.spec.whatwg.org/#concept-cd-substring) with node *original end node*, offset 0, and count *original end offset*.
  3. [Append](https://dom.spec.whatwg.org/#concept-node-append) *clone* to *fragment*.

1. Otherwise, if *last partially contained child* is not null:
   1. Let *clone* be a [clone](https://dom.spec.whatwg.org/#concept-node-clone) of *last partially contained child*.
   2. [Append](https://dom.spec.whatwg.org/#concept-node-append) *clone* to *fragment*.
   3. Let *subrange* be a new [range](https://dom.spec.whatwg.org/#concept-range) whose [start](https://dom.spec.whatwg.org/#concept-range-start) is (*last partially contained child*, 0) and whose [end](https://dom.spec.whatwg.org/#concept-range-end) is (*original end node*, *original end offset*).
   4. Let *subfragment* be the result of [cloning the contents](https://dom.spec.whatwg.org/#concept-range-clone) of *subrange*.
   5. [Append](https://dom.spec.whatwg.org/#concept-node-append) *subfragment* to *clone*.
2. Return *fragment*.

The *cloneContents()* method must return the result of [cloning the contents](https://dom.spec.whatwg.org/#concept-range-clone) of [context object](https://dom.spec.whatwg.org/#context-object).

To *insert* a [node](https://dom.spec.whatwg.org/#concept-node) *node* into a [range](https://dom.spec.whatwg.org/#concept-range) *range*, run these steps:

1. If *range*’s [start node](https://dom.spec.whatwg.org/#concept-range-start-node) is a [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction) or [Comment](https://dom.spec.whatwg.org/#comment) [node](https://dom.spec.whatwg.org/#concept-node), is a [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node) whose [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is null, or is *node*, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror).
2. Let *referenceNode* be null.
3. If *range*’s [start node](https://dom.spec.whatwg.org/#concept-range-start-node) is a [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node), set *referenceNode* to that [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node).
4. Otherwise, set *referenceNode* to the [child](https://dom.spec.whatwg.org/#concept-tree-child) of [start node](https://dom.spec.whatwg.org/#concept-range-start-node) whose [index](https://dom.spec.whatwg.org/#concept-tree-index) is [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset), and null if there is no such [child](https://dom.spec.whatwg.org/#concept-tree-child).
5. Let *parent* be *range*’s [start node](https://dom.spec.whatwg.org/#concept-range-start-node) if *referenceNode* is null, and *referenceNode*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) otherwise.
6. [Ensure pre-insertion validity](https://dom.spec.whatwg.org/#concept-node-ensure-pre-insertion-validity) of *node* into *parent* before *referenceNode*.
7. If *range*’s [start node](https://dom.spec.whatwg.org/#concept-range-start-node) is a [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node), set *referenceNode* to the result of [splitting](https://dom.spec.whatwg.org/#concept-text-split) it with offset *range*’s [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset).
8. If *node* is *referenceNode*, set *referenceNode* to its [next sibling](https://dom.spec.whatwg.org/#concept-tree-next-sibling).
9. If *node*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is not null, [remove](https://dom.spec.whatwg.org/#concept-node-remove) *node* from its [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
10. Let *newOffset* be *parent*’s [length](https://dom.spec.whatwg.org/#concept-node-length) if *referenceNode* is null, and *referenceNode*’s [index](https://dom.spec.whatwg.org/#concept-tree-index) otherwise.
11. Increase *newOffset* by *node*’s [length](https://dom.spec.whatwg.org/#concept-node-length) if *node* is a [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [node](https://dom.spec.whatwg.org/#concept-node), and one otherwise.
12. [Pre-insert](https://dom.spec.whatwg.org/#concept-node-pre-insert) *node* into *parent* before *referenceNode*.
13. If *range*’s [start](https://dom.spec.whatwg.org/#concept-range-start) and [end](https://dom.spec.whatwg.org/#concept-range-end) are the same, set *range*’s [end](https://dom.spec.whatwg.org/#concept-range-end) to (*parent*, *newOffset*).

The *insertNode(node)* method must [insert](https://dom.spec.whatwg.org/#concept-range-insert) *node* into [context object](https://dom.spec.whatwg.org/#context-object).

The *surroundContents(newParent)* method must run these steps:

1. If a non-[Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node) is [partially contained](https://dom.spec.whatwg.org/#partially-contained) in the [context object](https://dom.spec.whatwg.org/#context-object), then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidStateError](https://heycam.github.io/webidl/#invalidstateerror).
2. If *newParent* is a [Document](https://dom.spec.whatwg.org/#document), [DocumentType](https://dom.spec.whatwg.org/#documenttype), or [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [node](https://dom.spec.whatwg.org/#concept-node), then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidNodeTypeError](https://heycam.github.io/webidl/#invalidnodetypeerror).
3. Let *fragment* be the result of [extracting](https://dom.spec.whatwg.org/#concept-range-extract) [context object](https://dom.spec.whatwg.org/#context-object).
4. If *newParent* has [children](https://dom.spec.whatwg.org/#concept-tree-child), [replace all](https://dom.spec.whatwg.org/#concept-node-replace-all) with null within *newParent*.
5. [Insert](https://dom.spec.whatwg.org/#concept-range-insert) *newParent* into [context object](https://dom.spec.whatwg.org/#context-object).
6. [Append](https://dom.spec.whatwg.org/#concept-node-append) *fragment* to *newParent*.
7. [Select](https://dom.spec.whatwg.org/#concept-range-select) *newParent* within [context object](https://dom.spec.whatwg.org/#context-object).

The *cloneRange()* method must return a new [range](https://dom.spec.whatwg.org/#concept-range) with the same [start](https://dom.spec.whatwg.org/#concept-range-start) and [end](https://dom.spec.whatwg.org/#concept-range-end) as the [context object](https://dom.spec.whatwg.org/#context-object).

The *detach()* method must do nothing. Its functionality (disabling a [Range](https://dom.spec.whatwg.org/#range) object) was removed, but the method itself is preserved for compatibility.

*position* = *range* . [comparePoint(node, offset)](https://dom.spec.whatwg.org/#dom-range-comparepoint)

Returns −1 if the point is before the range, 0 if the point is in the range, and 1 if the point is after the range.

*intersects* = *range* . [intersectsNode(node)](https://dom.spec.whatwg.org/#dom-range-intersectsnode)

Returns whether *range* intersects *node*.

The *isPointInRange(node, offset)* must run these steps:

1. If *node*’s [root](https://dom.spec.whatwg.org/#concept-tree-root) is different from the [context object](https://dom.spec.whatwg.org/#context-object)’s [root](https://dom.spec.whatwg.org/#concept-range-root), return false.
2. If *node* is a [doctype](https://dom.spec.whatwg.org/#concept-doctype), then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidNodeTypeError](https://heycam.github.io/webidl/#invalidnodetypeerror).
3. If *offset* is greater than *node*’s [length](https://dom.spec.whatwg.org/#concept-node-length), then [throw](https://heycam.github.io/webidl/#dfn-throw) an [IndexSizeError](https://heycam.github.io/webidl/#indexsizeerror).
4. If (*node*, *offset*) is [before](https://dom.spec.whatwg.org/#concept-range-bp-before) [start](https://dom.spec.whatwg.org/#concept-range-start) or [after](https://dom.spec.whatwg.org/#concept-range-bp-after) [end](https://dom.spec.whatwg.org/#concept-range-end), return false.
5. Return true.

The *comparePoint(node, offset)* method must run these steps:

1. If *node*’s [root](https://dom.spec.whatwg.org/#concept-tree-root) is different from the [context object](https://dom.spec.whatwg.org/#context-object)’s [root](https://dom.spec.whatwg.org/#concept-range-root), then [throw](https://heycam.github.io/webidl/#dfn-throw) a [WrongDocumentError](https://heycam.github.io/webidl/#wrongdocumenterror).
2. If *node* is a [doctype](https://dom.spec.whatwg.org/#concept-doctype), then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidNodeTypeError](https://heycam.github.io/webidl/#invalidnodetypeerror).
3. If *offset* is greater than *node*’s [length](https://dom.spec.whatwg.org/#concept-node-length), then [throw](https://heycam.github.io/webidl/#dfn-throw) an [IndexSizeError](https://heycam.github.io/webidl/#indexsizeerror).
4. If (*node*, *offset*) is [before](https://dom.spec.whatwg.org/#concept-range-bp-before) [start](https://dom.spec.whatwg.org/#concept-range-start), return −1.
5. If (*node*, *offset*) is [after](https://dom.spec.whatwg.org/#concept-range-bp-after) [end](https://dom.spec.whatwg.org/#concept-range-end), return 1.
6. Return 0.

The *intersectsNode(node)* method must run these steps:

1. If *node*’s [root](https://dom.spec.whatwg.org/#concept-tree-root) is different from the [context object](https://dom.spec.whatwg.org/#context-object)’s [root](https://dom.spec.whatwg.org/#concept-range-root), return false.
2. Let *parent* be *node*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
3. If *parent* is null, return true.
4. Let *offset* be *node*’s [index](https://dom.spec.whatwg.org/#concept-tree-index).
5. If (*parent*, *offset*) is [before](https://dom.spec.whatwg.org/#concept-range-bp-before) [end](https://dom.spec.whatwg.org/#concept-range-end) and (*parent*, *offset* + 1) is [after](https://dom.spec.whatwg.org/#concept-range-bp-after) [start](https://dom.spec.whatwg.org/#concept-range-start), return true.
6. Return false.

The *stringification behavior* must run these steps:

1. Let *s* be the empty string.
2. If [start node](https://dom.spec.whatwg.org/#concept-range-start-node) is [end node](https://dom.spec.whatwg.org/#concept-range-end-node), and it is a [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node), return the substring of that [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node)’s [data](https://dom.spec.whatwg.org/#concept-cd-data) beginning at [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) and ending at [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset).
3. If [start node](https://dom.spec.whatwg.org/#concept-range-start-node) is a [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node), append to *s* the substring of that [node](https://dom.spec.whatwg.org/#concept-node)’s [data](https://dom.spec.whatwg.org/#concept-cd-data) from the [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset) until the end.
4. Append to *s* the concatenation, in [tree order](https://dom.spec.whatwg.org/#concept-tree-order), of the [data](https://dom.spec.whatwg.org/#concept-cd-data) of all [Text](https://dom.spec.whatwg.org/#text) [nodes](https://dom.spec.whatwg.org/#concept-node) that are [contained](https://dom.spec.whatwg.org/#contained) in the [context object](https://dom.spec.whatwg.org/#context-object).
5. If [end node](https://dom.spec.whatwg.org/#concept-range-end-node) is a [Text](https://dom.spec.whatwg.org/#text) [node](https://dom.spec.whatwg.org/#concept-node), append to *s* the substring of that [node](https://dom.spec.whatwg.org/#concept-node)’s [data](https://dom.spec.whatwg.org/#concept-cd-data) from its start until the [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset).
6. Return *s*.

The [createContextualFragment()](https://w3c.github.io/DOM-Parsing/#widl-Range-createContextualFragment-DocumentFragment-DOMString-fragment), [getClientRects()](https://drafts.csswg.org/cssom-view-1/#dom-range-getclientrects), and [getBoundingClientRect()](https://drafts.csswg.org/cssom-view-1/#dom-range-getboundingclientrect) methods are defined in other specifications. [[DOM-Parsing]](https://dom.spec.whatwg.org/#biblio-dom-parsing) [[CSSOM-VIEW]](https://dom.spec.whatwg.org/#biblio-cssom-view)

**6. Traversal**

[NodeIterator](https://dom.spec.whatwg.org/#nodeiterator) and [TreeWalker](https://dom.spec.whatwg.org/#treewalker) objects can be used to filter and traverse [node](https://dom.spec.whatwg.org/#concept-node) [trees](https://dom.spec.whatwg.org/#concept-tree).

Each [NodeIterator](https://dom.spec.whatwg.org/#nodeiterator) and [TreeWalker](https://dom.spec.whatwg.org/#treewalker) object also has an associated *root* [node](https://dom.spec.whatwg.org/#concept-node), *whatToShow* bitmask, and *filter* callback.

To *filter* *node* run these steps:

1. Let *n* be *node*’s [nodeType](https://dom.spec.whatwg.org/#dom-node-nodetype) attribute value minus 1.
2. If the *n*th bit (where 0 is the least significant bit) of [whatToShow](https://dom.spec.whatwg.org/#concept-traversal-whattoshow) is not set, return [FILTER\_SKIP](https://dom.spec.whatwg.org/#dom-nodefilter-filter_skip).
3. If [filter](https://dom.spec.whatwg.org/#concept-traversal-filter) is null, return [FILTER\_ACCEPT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_accept).
4. Let *result* be the return value of calling [filter](https://dom.spec.whatwg.org/#concept-traversal-filter)’s [acceptNode()](https://dom.spec.whatwg.org/#dom-nodefilter-acceptnode) with *node* as argument. Rethrow any exceptions.
5. Return *result*.

**6.1. Interface** [**NodeIterator**](https://dom.spec.whatwg.org/#nodeiterator)

[Exposed=Window]

interface *NodeIterator* {

[SameObject] readonly attribute [Node](https://dom.spec.whatwg.org/#node) [root](https://dom.spec.whatwg.org/#dom-nodeiterator-root);

readonly attribute [Node](https://dom.spec.whatwg.org/#node) [referenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-referencenode);

readonly attribute boolean [pointerBeforeReferenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-pointerbeforereferencenode);

readonly attribute unsigned long [whatToShow](https://dom.spec.whatwg.org/#dom-nodeiterator-whattoshow);

readonly attribute [NodeFilter](https://dom.spec.whatwg.org/#callbackdef-nodefilter)? [filter](https://dom.spec.whatwg.org/#dom-nodeiterator-filter);

[Node](https://dom.spec.whatwg.org/#node)? [nextNode](https://dom.spec.whatwg.org/#dom-nodeiterator-nextnode)();

[Node](https://dom.spec.whatwg.org/#node)? [previousNode](https://dom.spec.whatwg.org/#dom-nodeiterator-previousnode)();

void [detach](https://dom.spec.whatwg.org/#dom-nodeiterator-detach)();

};

[NodeIterator](https://dom.spec.whatwg.org/#nodeiterator) objects can be created using the [createNodeIterator()](https://dom.spec.whatwg.org/#dom-document-createnodeiterator) method.

Each [NodeIterator](https://dom.spec.whatwg.org/#nodeiterator) object has an associated *iterator collection*, which is a [collection](https://dom.spec.whatwg.org/#concept-collection) rooted at [root](https://dom.spec.whatwg.org/#concept-traversal-root), whose filter matches any [node](https://dom.spec.whatwg.org/#concept-node).

As mentioned earlier, [NodeIterator](https://dom.spec.whatwg.org/#nodeiterator) objects have an associated [root](https://dom.spec.whatwg.org/#concept-traversal-root) [node](https://dom.spec.whatwg.org/#concept-node), [whatToShow](https://dom.spec.whatwg.org/#concept-traversal-whattoshow) bitmask, and [filter](https://dom.spec.whatwg.org/#concept-traversal-filter) callback as well.

The *NodeIterator pre-removing steps* given a *nodeIterator* and *toBeRemovedNode*, are as follows:

1. If *toBeRemovedNode* is not an [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor) of the [referenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-referencenode) attribute value, terminate these steps.
2. If the [pointerBeforeReferenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-pointerbeforereferencenode) attribute value is true, run these substeps:
   1. Let *next* be *toBeRemovedNode*’s first [following](https://dom.spec.whatwg.org/#concept-tree-following) [node](https://dom.spec.whatwg.org/#concept-node) that is an [inclusive descendant](https://dom.spec.whatwg.org/#concept-tree-inclusive-descendant) of *nodeIterator*’s [root](https://dom.spec.whatwg.org/#concept-traversal-root) and is not an [inclusive descendant](https://dom.spec.whatwg.org/#concept-tree-inclusive-descendant) of *toBeRemovedNode*, and null if there is no such [node](https://dom.spec.whatwg.org/#concept-node).
   2. If *next* is non-null, set *nodeIterator*’s [referenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-referencenode) attribute to *next* and terminate these steps.
   3. Otherwise, set *nodeIterator*’s [pointerBeforeReferenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-pointerbeforereferencenode) attribute to false.

Steps are not terminated here.

1. Set *nodeIterator*’s [referenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-referencenode) attribute to *toBeRemovedNode*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent), if *toBeRemovedNode*’s [previous sibling](https://dom.spec.whatwg.org/#concept-tree-previous-sibling) is null, and to the [inclusive descendant](https://dom.spec.whatwg.org/#concept-tree-inclusive-descendant) of *toBeRemovedNode*’s [previous sibling](https://dom.spec.whatwg.org/#concept-tree-previous-sibling) that appears last in [tree order](https://dom.spec.whatwg.org/#concept-tree-order) otherwise.

The *root* attribute must return [root](https://dom.spec.whatwg.org/#concept-traversal-root).

The *referenceNode* and *pointerBeforeReferenceNode* attributes must return what they were initialized to.

The *whatToShow* attribute must return [whatToShow](https://dom.spec.whatwg.org/#concept-traversal-whattoshow).

The *filter* attribute must return [filter](https://dom.spec.whatwg.org/#concept-traversal-filter).

To *traverse* in direction *direction* run these steps:

1. Let *node* be the value of the [referenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-referencenode) attribute.
2. Let *before node* be the value of the [pointerBeforeReferenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-pointerbeforereferencenode) attribute.
3. Run these substeps:
   1. If *direction* is next

If *before node* is false, let *node* be the first [node](https://dom.spec.whatwg.org/#concept-node) [following](https://dom.spec.whatwg.org/#concept-tree-following) *node* in the [iterator collection](https://dom.spec.whatwg.org/#iterator-collection). If there is no such [node](https://dom.spec.whatwg.org/#concept-node) return null. If *before node* is true, set it to false.

If *direction* is previous

If *before node* is true, let *node* be the first [node](https://dom.spec.whatwg.org/#concept-node) [preceding](https://dom.spec.whatwg.org/#concept-tree-preceding) *node* in the [iterator collection](https://dom.spec.whatwg.org/#iterator-collection). If there is no such [node](https://dom.spec.whatwg.org/#concept-node) return null. If *before node* is false, set it to true.

* 1. [Filter](https://dom.spec.whatwg.org/#concept-node-filter) *node* and let *result* be the return value.
  2. If *result* is [FILTER\_ACCEPT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_accept), go to the next step in the overall set of steps. Otherwise, run these substeps again.

1. Set the [referenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-referencenode) attribute to *node*, set the [pointerBeforeReferenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-pointerbeforereferencenode) attribute to *before node*, and return *node*.

The *nextNode()* method must [traverse](https://dom.spec.whatwg.org/#concept-nodeiterator-traverse) in direction next.

The *previousNode()* method must [traverse](https://dom.spec.whatwg.org/#concept-nodeiterator-traverse) in direction previous.

The *detach()* method must do nothing. Its functionality (disabling a [NodeIterator](https://dom.spec.whatwg.org/#nodeiterator) object) was removed, but the method itself is preserved for compatibility.

**6.2. Interface** [**TreeWalker**](https://dom.spec.whatwg.org/#treewalker)

[Exposed=Window]

interface *TreeWalker* {

[SameObject] readonly attribute [Node](https://dom.spec.whatwg.org/#node) [root](https://dom.spec.whatwg.org/#dom-treewalker-root);

readonly attribute unsigned long [whatToShow](https://dom.spec.whatwg.org/#dom-treewalker-whattoshow);

readonly attribute [NodeFilter](https://dom.spec.whatwg.org/#callbackdef-nodefilter)? [filter](https://dom.spec.whatwg.org/#dom-treewalker-filter);

attribute [Node](https://dom.spec.whatwg.org/#node) [currentNode](https://dom.spec.whatwg.org/#dom-treewalker-currentnode);

[Node](https://dom.spec.whatwg.org/#node)? [parentNode](https://dom.spec.whatwg.org/#dom-treewalker-parentnode)();

[Node](https://dom.spec.whatwg.org/#node)? [firstChild](https://dom.spec.whatwg.org/#dom-treewalker-firstchild)();

[Node](https://dom.spec.whatwg.org/#node)? [lastChild](https://dom.spec.whatwg.org/#dom-treewalker-lastchild)();

[Node](https://dom.spec.whatwg.org/#node)? [previousSibling](https://dom.spec.whatwg.org/#dom-treewalker-previoussibling)();

[Node](https://dom.spec.whatwg.org/#node)? [nextSibling](https://dom.spec.whatwg.org/#dom-treewalker-nextsibling)();

[Node](https://dom.spec.whatwg.org/#node)? [previousNode](https://dom.spec.whatwg.org/#dom-treewalker-previousnode)();

[Node](https://dom.spec.whatwg.org/#node)? [nextNode](https://dom.spec.whatwg.org/#dom-treewalker-nextnode)();

};

[TreeWalker](https://dom.spec.whatwg.org/#treewalker) objects can be created using the [createTreeWalker()](https://dom.spec.whatwg.org/#dom-document-createtreewalker) method.

As mentioned earlier [TreeWalker](https://dom.spec.whatwg.org/#treewalker) objects have an associated [root](https://dom.spec.whatwg.org/#concept-traversal-root) [node](https://dom.spec.whatwg.org/#concept-node), [whatToShow](https://dom.spec.whatwg.org/#concept-traversal-whattoshow) bitmask, and [filter](https://dom.spec.whatwg.org/#concept-traversal-filter) callback.

The *root* attribute must return [root](https://dom.spec.whatwg.org/#concept-traversal-root).

The *whatToShow* attribute must return [whatToShow](https://dom.spec.whatwg.org/#concept-traversal-whattoshow).

The *filter* attribute must return [filter](https://dom.spec.whatwg.org/#concept-traversal-filter).

The *currentNode* attribute must return what it was initialized to.

Setting the [currentNode](https://dom.spec.whatwg.org/#dom-treewalker-currentnode) attribute must set it to the new value.

The *parentNode()* method must run these steps:

1. Let *node* be the value of the [currentNode](https://dom.spec.whatwg.org/#dom-treewalker-currentnode) attribute.
2. While *node* is not null and is not [root](https://dom.spec.whatwg.org/#concept-traversal-root), run these substeps:
   1. Let *node* be *node*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
   2. If *node* is not null and [filtering](https://dom.spec.whatwg.org/#concept-node-filter) *node* returns [FILTER\_ACCEPT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_accept), then set the [currentNode](https://dom.spec.whatwg.org/#dom-treewalker-currentnode) attribute to *node*, return *node*.
3. Return null.

To *traverse children* of type *type*, run these steps:

1. Let *node* be the value of the [currentNode](https://dom.spec.whatwg.org/#dom-treewalker-currentnode) attribute.
2. Set *node* to *node*’s [first child](https://dom.spec.whatwg.org/#concept-tree-first-child) if *type* is first, and *node*’s [last child](https://dom.spec.whatwg.org/#concept-tree-last-child) if *type* is last.
3. If *node* is null, return null.
4. *Main*: Repeat these substeps:
   1. [Filter](https://dom.spec.whatwg.org/#concept-node-filter) *node* and let *result* be the return value.
   2. If *result* is [FILTER\_ACCEPT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_accept), then set the [currentNode](https://dom.spec.whatwg.org/#dom-treewalker-currentnode) attribute to *node* and return *node*.
   3. If *result* is [FILTER\_SKIP](https://dom.spec.whatwg.org/#dom-nodefilter-filter_skip), run these subsubsteps:
      1. Let *child* be *node*’s [first child](https://dom.spec.whatwg.org/#concept-tree-first-child) if *type* is first, and *node*’s [last child](https://dom.spec.whatwg.org/#concept-tree-last-child) if *type* is last.
      2. If *child* is not null, set *node* to *child* and goto [Main](https://dom.spec.whatwg.org/#concept-traverse-children-main).
   4. Repeat these subsubsteps:
      1. Let *sibling* be *node*’s [next sibling](https://dom.spec.whatwg.org/#concept-tree-next-sibling) if *type* is first, and *node*’s [previous sibling](https://dom.spec.whatwg.org/#concept-tree-previous-sibling) if *type* is last.
      2. If *sibling* is not null, set *node* to *sibling* and goto [Main](https://dom.spec.whatwg.org/#concept-traverse-children-main).
      3. Let *parent* be *node*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
      4. If *parent* is null, *parent* is [root](https://dom.spec.whatwg.org/#concept-traversal-root), or *parent* is [currentNode](https://dom.spec.whatwg.org/#dom-treewalker-currentnode) attribute’s value, return null.
      5. Otherwise, set *node* to *parent*.

The *firstChild()* method must [traverse children](https://dom.spec.whatwg.org/#concept-traverse-children) of type first.

The *lastChild()* method must [traverse children](https://dom.spec.whatwg.org/#concept-traverse-children) of type last.

To *traverse siblings* of type *type* run these steps:

1. Let *node* be the value of the [currentNode](https://dom.spec.whatwg.org/#dom-treewalker-currentnode) attribute.
2. If *node* is [root](https://dom.spec.whatwg.org/#concept-traversal-root), return null.
3. Run these substeps:
   1. Let *sibling* be *node*’s [next sibling](https://dom.spec.whatwg.org/#concept-tree-next-sibling) if *type* is next, and *node*’s [previous sibling](https://dom.spec.whatwg.org/#concept-tree-previous-sibling) if *type* is previous.
   2. While *sibling* is not null, run these subsubsteps:
      1. Set *node* to *sibling*.
      2. [Filter](https://dom.spec.whatwg.org/#concept-node-filter) *node* and let *result* be the return value.
      3. If *result* is [FILTER\_ACCEPT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_accept), then set the [currentNode](https://dom.spec.whatwg.org/#dom-treewalker-currentnode) attribute to *node* and return *node*.
      4. Set *sibling* to *node*’s [first child](https://dom.spec.whatwg.org/#concept-tree-first-child) if *type* is next, and *node*’s [last child](https://dom.spec.whatwg.org/#concept-tree-last-child) if *type* is previous.
      5. If *result* is [FILTER\_REJECT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_reject) or *sibling* is null, then set *sibling* to *node*’s [next sibling](https://dom.spec.whatwg.org/#concept-tree-next-sibling) if *type* is next, and *node*’s [previous sibling](https://dom.spec.whatwg.org/#concept-tree-previous-sibling) if *type* is previous.
   3. Set *node* to its [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
   4. If *node* is null or is [root](https://dom.spec.whatwg.org/#concept-traversal-root), return null.
   5. [Filter](https://dom.spec.whatwg.org/#concept-node-filter) *node* and if the return value is [FILTER\_ACCEPT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_accept), then return null.
   6. Run these substeps again.

The *nextSibling()* method must [traverse siblings](https://dom.spec.whatwg.org/#concept-traverse-siblings) of type next.

The *previousSibling()* method must [traverse siblings](https://dom.spec.whatwg.org/#concept-traverse-siblings) of type previous.

The *previousNode()* method must run these steps:

1. Let *node* be the value of the [currentNode](https://dom.spec.whatwg.org/#dom-treewalker-currentnode) attribute.
2. While *node* is not [root](https://dom.spec.whatwg.org/#concept-traversal-root), run these substeps:
   1. Let *sibling* be the [previous sibling](https://dom.spec.whatwg.org/#concept-tree-previous-sibling) of *node*.
   2. While *sibling* is not null, run these subsubsteps:
      1. Set *node* to *sibling*.
      2. [Filter](https://dom.spec.whatwg.org/#concept-node-filter) *node* and let *result* be the return value.
      3. While *result* is not [FILTER\_REJECT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_reject) and *node* has a [child](https://dom.spec.whatwg.org/#concept-tree-child), set *node* to its [last child](https://dom.spec.whatwg.org/#concept-tree-last-child) and then [filter](https://dom.spec.whatwg.org/#concept-node-filter) *node* and set *result* to the return value.
      4. If *result* is [FILTER\_ACCEPT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_accept), then set the [currentNode](https://dom.spec.whatwg.org/#dom-treewalker-currentnode) attribute to *node* and return *node*.
      5. Set *sibling* to the [previous sibling](https://dom.spec.whatwg.org/#concept-tree-previous-sibling) of *node*.
   3. If *node* is [root](https://dom.spec.whatwg.org/#concept-traversal-root) or *node*’s [parent](https://dom.spec.whatwg.org/#concept-tree-parent) is null, return null.
   4. Set *node* to its [parent](https://dom.spec.whatwg.org/#concept-tree-parent).
   5. [Filter](https://dom.spec.whatwg.org/#concept-node-filter) *node* and if the return value is [FILTER\_ACCEPT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_accept), then set the [currentNode](https://dom.spec.whatwg.org/#dom-treewalker-currentnode) attribute to *node* and return *node*.
3. Return null.

The *nextNode()* method must run these steps:

1. Let *node* be the value of the [currentNode](https://dom.spec.whatwg.org/#dom-treewalker-currentnode) attribute.
2. Let *result* be [FILTER\_ACCEPT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_accept).
3. Run these substeps:
   1. While *result* is not [FILTER\_REJECT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_reject) and *node* has a [child](https://dom.spec.whatwg.org/#concept-tree-child), run these subsubsteps:
      1. Set *node* to its [first child](https://dom.spec.whatwg.org/#concept-tree-first-child).
      2. [Filter](https://dom.spec.whatwg.org/#concept-node-filter) *node* and set *result* to the return value.
      3. If *result* is [FILTER\_ACCEPT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_accept), then set the [currentNode](https://dom.spec.whatwg.org/#dom-treewalker-currentnode) attribute to *node* and return *node*.
   2. If a [node](https://dom.spec.whatwg.org/#concept-node) is [following](https://dom.spec.whatwg.org/#concept-tree-following) *node* and is not [following](https://dom.spec.whatwg.org/#concept-tree-following) [root](https://dom.spec.whatwg.org/#concept-traversal-root), set *node* to the first such [node](https://dom.spec.whatwg.org/#concept-node). Otherwise, return null.
   3. [Filter](https://dom.spec.whatwg.org/#concept-node-filter) *node* and set *result* to the return value.
   4. If *result* is [FILTER\_ACCEPT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_accept), then set the [currentNode](https://dom.spec.whatwg.org/#dom-treewalker-currentnode) attribute to *node* and return *node*.
   5. Run these substeps again.

**6.3. Interface** [**NodeFilter**](https://dom.spec.whatwg.org/#callbackdef-nodefilter)

[Exposed=Window]

callback interface *NodeFilter* {

// Constants for acceptNode()

const unsigned short [FILTER\_ACCEPT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_accept) = 1;

const unsigned short [FILTER\_REJECT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_reject) = 2;

const unsigned short [FILTER\_SKIP](https://dom.spec.whatwg.org/#dom-nodefilter-filter_skip) = 3;

// Constants for whatToShow

const unsigned long [SHOW\_ALL](https://dom.spec.whatwg.org/#dom-nodefilter-show_all) = 0xFFFFFFFF;

const unsigned long [SHOW\_ELEMENT](https://dom.spec.whatwg.org/#dom-nodefilter-show_element) = 0x1;

const unsigned long *SHOW\_ATTRIBUTE* = 0x2; // historical

const unsigned long [SHOW\_TEXT](https://dom.spec.whatwg.org/#dom-nodefilter-show_text) = 0x4;

const unsigned long *SHOW\_CDATA\_SECTION* = 0x8; // historical

const unsigned long *SHOW\_ENTITY\_REFERENCE* = 0x10; // historical

const unsigned long *SHOW\_ENTITY* = 0x20; // historical

const unsigned long [SHOW\_PROCESSING\_INSTRUCTION](https://dom.spec.whatwg.org/#dom-nodefilter-show_processing_instruction) = 0x40;

const unsigned long [SHOW\_COMMENT](https://dom.spec.whatwg.org/#dom-nodefilter-show_comment) = 0x80;

const unsigned long [SHOW\_DOCUMENT](https://dom.spec.whatwg.org/#dom-nodefilter-show_document) = 0x100;

const unsigned long [SHOW\_DOCUMENT\_TYPE](https://dom.spec.whatwg.org/#dom-nodefilter-show_document_type) = 0x200;

const unsigned long [SHOW\_DOCUMENT\_FRAGMENT](https://dom.spec.whatwg.org/#dom-nodefilter-show_document_fragment) = 0x400;

const unsigned long *SHOW\_NOTATION* = 0x800; // historical

unsigned short *acceptNode*([Node](https://dom.spec.whatwg.org/#node) *node*);

};

[NodeFilter](https://dom.spec.whatwg.org/#callbackdef-nodefilter) objects can be used as [filter](https://dom.spec.whatwg.org/#concept-traversal-filter) callback and provide constants for the [whatToShow](https://dom.spec.whatwg.org/#concept-traversal-whattoshow) bitmask.

It is typically implemented as a JavaScript function.

These constants can be used as callback return value:

* *FILTER\_ACCEPT* (1);
* *FILTER\_REJECT* (2);
* *FILTER\_SKIP* (3).

These constants can be used for the [whatToShow](https://dom.spec.whatwg.org/#concept-traversal-whattoshow) bitmask:

* *SHOW\_ALL* (4294967295, FFFFFFFF in hexadecimal);
* *SHOW\_ELEMENT* (1);
* *SHOW\_TEXT* (4);
* *SHOW\_PROCESSING\_INSTRUCTION* (64, 40 in hexadecimal);
* *SHOW\_COMMENT* (128, 80 in hexadecimal);
* *SHOW\_DOCUMENT* (256, 100 in hexadecimal);
* *SHOW\_DOCUMENT\_TYPE* (512, 200 in hexadecimal);
* *SHOW\_DOCUMENT\_FRAGMENT* (1024, 400 in hexadecimal).

**7. Sets**

Yes, the name [DOMTokenList](https://dom.spec.whatwg.org/#domtokenlist) is an unfortunate legacy mishap.

**7.1. Interface** [**DOMTokenList**](https://dom.spec.whatwg.org/#domtokenlist)

interface *DOMTokenList* {

readonly attribute unsigned long [length](https://dom.spec.whatwg.org/#dom-domtokenlist-length);

getter DOMString? [item](https://dom.spec.whatwg.org/#dom-domtokenlist-item)(unsigned long *index*);

boolean [contains](https://dom.spec.whatwg.org/#dom-domtokenlist-contains)(DOMString *token*);

[CEReactions] void [add](https://dom.spec.whatwg.org/#dom-domtokenlist-add)(DOMString... *tokens*);

[CEReactions] void [remove](https://dom.spec.whatwg.org/#dom-domtokenlist-remove)(DOMString... *tokens*);

[CEReactions] boolean [toggle](https://dom.spec.whatwg.org/#dom-domtokenlist-toggle)(DOMString *token*, optional boolean *force*);

[CEReactions] void [replace](https://dom.spec.whatwg.org/#dom-domtokenlist-replace)(DOMString *token*, DOMString *newToken*);

boolean [supports](https://dom.spec.whatwg.org/#dom-domtokenlist-supports)(DOMString *token*);

[CEReactions] attribute DOMString [value](https://dom.spec.whatwg.org/#dom-domtokenlist-value);

[stringifier](https://dom.spec.whatwg.org/#dom-domtokenlist-stringifier);

iterable<DOMString>;

};

A [DOMTokenList](https://dom.spec.whatwg.org/#domtokenlist) object has an associated ordered set of *tokens*, which is initially empty.

A [DOMTokenList](https://dom.spec.whatwg.org/#domtokenlist) object also has an associated [element](https://dom.spec.whatwg.org/#concept-element) and an [attribute](https://dom.spec.whatwg.org/#concept-attribute)’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name).

[Specifications](https://dom.spec.whatwg.org/#other-applicable-specifications) may define *supported tokens* for a [DOMTokenList](https://dom.spec.whatwg.org/#domtokenlist)'s associated [attribute](https://dom.spec.whatwg.org/#concept-attribute)’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name).

A [DOMTokenList](https://dom.spec.whatwg.org/#domtokenlist) object’s *validation steps* for a given *token* are:

1. If the associated [attribute](https://dom.spec.whatwg.org/#concept-attribute)’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name) does not define [supported tokens](https://dom.spec.whatwg.org/#concept-supported-tokens), [throw](https://heycam.github.io/webidl/#dfn-throw) a TypeError.
2. Let *lowercase token* be a copy of *token*, [converted to ASCII lowercase](https://dom.spec.whatwg.org/#converted-to-ascii-lowercase).
3. If *lowercase token* is present in [supported tokens](https://dom.spec.whatwg.org/#concept-supported-tokens), return true.
4. Return false.

A [DOMTokenList](https://dom.spec.whatwg.org/#domtokenlist) object’s *update steps* are to [set an attribute value](https://dom.spec.whatwg.org/#concept-element-attributes-set-value) for the associated [element](https://dom.spec.whatwg.org/#concept-element) using associated [attribute](https://dom.spec.whatwg.org/#concept-attribute)’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name) and the result of running the [ordered set serializer](https://dom.spec.whatwg.org/#concept-ordered-set-serializer) for [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens).

A [DOMTokenList](https://dom.spec.whatwg.org/#domtokenlist) object’s *serialize steps* are to return the result of running [get an attribute value](https://dom.spec.whatwg.org/#concept-element-attributes-get-value) given the associated [element](https://dom.spec.whatwg.org/#concept-element) and the associated [attribute](https://dom.spec.whatwg.org/#concept-attribute)’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name).

A [DOMTokenList](https://dom.spec.whatwg.org/#domtokenlist) object has these [attribute change steps](https://dom.spec.whatwg.org/#concept-element-attributes-change-ext) for its associated [element](https://dom.spec.whatwg.org/#concept-element):

1. If *localName* is associated attribute’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name), *namespace* is null, and *value* is null, then set [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens) to the empty set.
2. Otherwise, *localName* is associated attribute’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name), *namespace* is null, then set [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens) to *value*, [parsed](https://dom.spec.whatwg.org/#concept-ordered-set-parser).

When a [DOMTokenList](https://dom.spec.whatwg.org/#domtokenlist) object is created, run these substeps:

1. Let *element* be associated [element](https://dom.spec.whatwg.org/#concept-element).
2. Let *localName* be associated attribute’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name).
3. Let *value* be the result of [getting an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-get-by-namespace) given null, *localName*, and *element*.
4. Run the [attribute change steps](https://dom.spec.whatwg.org/#concept-element-attributes-change-ext) for *element*, *localName*, *value*, *value*, and null.

*tokenlist* . [length](https://dom.spec.whatwg.org/#dom-domtokenlist-length)

Returns the number of tokens.

*tokenlist* . [item(index)](https://dom.spec.whatwg.org/#dom-domtokenlist-item)

*tokenlist*[*index*]

Returns the token with index *index*.

*tokenlist* . [contains(token)](https://dom.spec.whatwg.org/#dom-domtokenlist-contains)

Returns true if *token* is present, and false otherwise.

*tokenlist* . [add(*tokens*…)](https://dom.spec.whatwg.org/#dom-domtokenlist-add)

Adds all arguments passed, except those already present.

Throws a [SyntaxError](https://heycam.github.io/webidl/#syntaxerror) if one of the arguments is the empty string.

Throws an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror) if one of the arguments contains any [ASCII whitespace](https://encoding.spec.whatwg.org/#ascii-whitespace).

*tokenlist* . [remove(*tokens*…)](https://dom.spec.whatwg.org/#dom-domtokenlist-remove)

Removes arguments passed, if they are present.

Throws a [SyntaxError](https://heycam.github.io/webidl/#syntaxerror) if one of the arguments is the empty string.

Throws an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror) if one of the arguments contains any [ASCII whitespace](https://encoding.spec.whatwg.org/#ascii-whitespace).

*tokenlist* . [toggle(*token* [, *force*])](https://dom.spec.whatwg.org/#dom-domtokenlist-toggle)

If *force* is not given, "toggles" *token*, removing it if it’s present and adding it if it’s not present. If *force* is true, adds *token* (same as [add()](https://dom.spec.whatwg.org/#dom-domtokenlist-add)). If *force* is false, removes *token* (same as [remove()](https://dom.spec.whatwg.org/#dom-domtokenlist-remove)).

Returns true if *token* is now present, and false otherwise.

Throws a [SyntaxError](https://heycam.github.io/webidl/#syntaxerror) if *token* is empty.

Throws an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror) if *token* contains any spaces.

*tokenlist* . [replace(*token*, *newToken*)](https://dom.spec.whatwg.org/#dom-domtokenlist-replace)

Replaces *token* with *newToken*.

Throws a [SyntaxError](https://heycam.github.io/webidl/#syntaxerror) if one of the arguments is the empty string.

Throws an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror) if one of the arguments contains any [ASCII whitespace](https://encoding.spec.whatwg.org/#ascii-whitespace).

*tokenlist* . [supports(*token*)](https://dom.spec.whatwg.org/#dom-domtokenlist-supports)

Returns true if *token* is in the associated attribute’s supported tokens. Returns false otherwise.

Throws a TypeError if the associated attribute has no supported tokens defined.

*tokenlist* . [value](https://dom.spec.whatwg.org/#dom-domtokenlist-value)

Returns the associated set as string.

Can be set, to change the associated attribute.

The *length* attribute' getter must return the number of tokens in the [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens).

The object’s [supported property indices](https://heycam.github.io/webidl/#dfn-supported-property-indices) are the numbers in the range zero to the number of tokens in [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens) minus one, unless [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens) is empty, in which case there are no [supported property indices](https://heycam.github.io/webidl/#dfn-supported-property-indices).

The *item(index)* method, when invoked, must run these steps:

1. If *index* is equal to or greater than the number of tokens in [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens), return null.
2. Return the *index*th token in [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens).

The *contains(token)* method, when invoked, must return true if *token* is in [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens), and false otherwise.

The *add(tokens…)* method, when invoked, must run these steps:

1. For each *token* in *tokens*, run these substeps:
   1. If *token* is the empty string, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [SyntaxError](https://heycam.github.io/webidl/#syntaxerror).
   2. If *token* contains any [ASCII whitespace](https://encoding.spec.whatwg.org/#ascii-whitespace), then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror).
2. For each *token* in *tokens*, in given order, that is not in [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens), append *token* to [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens).
3. Run the [update steps](https://dom.spec.whatwg.org/#concept-dtl-update).

The *remove(tokens…)* method, when invoked, must run these steps:

1. For each *token* in *tokens*, run these substeps:
   1. If *token* is the empty string, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [SyntaxError](https://heycam.github.io/webidl/#syntaxerror).
   2. If *token* contains any [ASCII whitespace](https://encoding.spec.whatwg.org/#ascii-whitespace), then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror).
2. For each *token* in *tokens*, remove *token* from [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens).
3. Run the [update steps](https://dom.spec.whatwg.org/#concept-dtl-update).

The *toggle(token, force)* method, when invoked, must run these steps:

1. If *token* is the empty string, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [SyntaxError](https://heycam.github.io/webidl/#syntaxerror).
2. If *token* contains any [ASCII whitespace](https://encoding.spec.whatwg.org/#ascii-whitespace), then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror).
3. If *token* is in [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens), run these substeps:
   1. If *force* is either not passed or is false, then remove *token* from [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens), run the [update steps](https://dom.spec.whatwg.org/#concept-dtl-update), and return false.
   2. Otherwise, return true.
4. Otherwise, run these substeps:
   1. If *force* is passed and is false, return false.
   2. Otherwise, append *token* to [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens), run the [update steps](https://dom.spec.whatwg.org/#concept-dtl-update), and return true.

The *replace(token, newToken)* method, when invoked, must run these steps:

1. If either *token* or *newToken* is the empty string, then [throw](https://heycam.github.io/webidl/#dfn-throw) a [SyntaxError](https://heycam.github.io/webidl/#syntaxerror).
2. If either *token* or *newToken* contains any [ASCII whitespace](https://encoding.spec.whatwg.org/#ascii-whitespace), then [throw](https://heycam.github.io/webidl/#dfn-throw) an [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror).
3. If *token* is not in [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens), terminate these steps.
4. Replace *token* in [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens) with *newToken*.
5. Run the [update steps](https://dom.spec.whatwg.org/#concept-dtl-update).

The *supports(token)* method, when invoked, must run these steps:

1. Let *result* be the return value of [validation steps](https://dom.spec.whatwg.org/#concept-domtokenlist-validation) called with *token*. Rethrow any exceptions.
2. Return *result*.

The *value* attribute must return the result of running [context object](https://dom.spec.whatwg.org/#context-object)’s [serialize steps](https://dom.spec.whatwg.org/#concept-dtl-serialize).

Setting the [value](https://dom.spec.whatwg.org/#dom-domtokenlist-value) attribute must [set an attribute value](https://dom.spec.whatwg.org/#concept-element-attributes-set-value) for the associated [element](https://dom.spec.whatwg.org/#concept-element) using associated [attribute](https://dom.spec.whatwg.org/#concept-attribute)’s [local name](https://dom.spec.whatwg.org/#concept-attribute-local-name) and the given value.

The *stringification behavior* must return the result of running [context object](https://dom.spec.whatwg.org/#context-object)’s [serialize steps](https://dom.spec.whatwg.org/#concept-dtl-serialize).

**8. Historical**

As explained in [goals](https://dom.spec.whatwg.org/#goals) this specification is a significant revision of various DOM specifications. This section attempts to enumerate the changes.

**8.1. DOM Events**

These are the changes made to the features described in the "DOM Event Architecture", "Basic Event Interfaces", "Mutation Events", and "Mutation Name Event Types" chapters of *DOM Level 3 Events*. The other chapters are defined by the *UI Events* specification. [[UIEVENTS]](https://dom.spec.whatwg.org/#biblio-uievents)

* Events have constructors now.
* Removes *MutationEvent*, and *MutationNameEvent*.
* Fire is no longer synonymous with dispatch, but includes initializing an event.
* The propagation and canceled flags are unset when invoking [initEvent()](https://dom.spec.whatwg.org/#dom-event-initevent) rather than after dispatch.

**8.2. DOM Core**

These are the changes made to the features described in *DOM Level 3 Core*.

[DOMString](https://heycam.github.io/webidl/#idl-DOMString), [DOMException](https://heycam.github.io/webidl/#dfn-DOMException), and [DOMTimeStamp](https://heycam.github.io/webidl/#common-domtimestamp) are now defined in Web IDL.

[Node](https://dom.spec.whatwg.org/#node) now inherits from [EventTarget](https://dom.spec.whatwg.org/#eventtarget).

[Nodes](https://dom.spec.whatwg.org/#concept-node) are implicitly [adopted](https://dom.spec.whatwg.org/#concept-node-adopt) across [document](https://dom.spec.whatwg.org/#concept-document) boundaries.

[Doctypes](https://dom.spec.whatwg.org/#concept-doctype) now always have a [node document](https://dom.spec.whatwg.org/#concept-node-document) and can be moved across [document](https://dom.spec.whatwg.org/#concept-document) boundaries.

[ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction) now inherits from [CharacterData](https://dom.spec.whatwg.org/#characterdata).

*hasAttributes()* and *attributes* moved from [Node](https://dom.spec.whatwg.org/#node) to [Element](https://dom.spec.whatwg.org/#element).

*namespaceURI*, *prefix*, and *localName* moved from [Node](https://dom.spec.whatwg.org/#node) to [Element](https://dom.spec.whatwg.org/#element) and [Attr](https://dom.spec.whatwg.org/#attr).

The remainder of interfaces and interface members listed in this section were removed to simplify the DOM platform. Implementations conforming to this specification will not support them.

It is not yet clear if it would be web-compatible to remove all the following features. The editors welcome any data showing that some of these features should be reintroduced.

Interfaces:

* *CDATASection*
* *DOMConfiguration*
* *DOMError*
* *DOMErrorHandler*
* *DOMImplementationList*
* *DOMImplementationSource*
* *DOMLocator*
* *DOMObject*
* *DOMStringList*
* *DOMUserData*
* *Entity*
* *EntityReference*
* *NameList*
* *Notation*
* *TypeInfo*
* *UserDataHandler*

Interface members:

[Node](https://dom.spec.whatwg.org/#node)

* *isSupported*
* *getFeature()*
* *getUserData()*
* *setUserData()*

[Document](https://dom.spec.whatwg.org/#document)

* *createCDATASection()*
* *createEntityReference()*
* *xmlEncoding*
* *xmlStandalone*
* *xmlVersion*
* *strictErrorChecking*
* *domConfig*
* *normalizeDocument()*
* *renameNode()*

[DOMImplementation](https://dom.spec.whatwg.org/#domimplementation)

* *getFeature()*

[Attr](https://dom.spec.whatwg.org/#attr)

No longer inherits from [Node](https://dom.spec.whatwg.org/#node) and therefore completely changed.

[Element](https://dom.spec.whatwg.org/#element)

* *schemaTypeInfo*
* *setIdAttribute()*
* *setIdAttributeNS()*
* *setIdAttributeNode()*

[DocumentType](https://dom.spec.whatwg.org/#documenttype)

* *entities*
* *notations*
* *internalSubset*

[Text](https://dom.spec.whatwg.org/#text)

* *isElementContentWhitespace*
* *replaceWholeText()*

**8.3. DOM Ranges**

These are the changes made to the features described in the "Document Object Model Range" chapter of *DOM Level 2 Traversal and Range*.

* *RangeException* has been removed.
* [Range](https://dom.spec.whatwg.org/#range) objects can now be moved between [documents](https://dom.spec.whatwg.org/#concept-document) and used on [nodes](https://dom.spec.whatwg.org/#concept-node) that are not [in a document](https://dom.spec.whatwg.org/#in-a-document).
* A wild [Range()](https://dom.spec.whatwg.org/#dom-range-range) constructor appeared.
* New methods [comparePoint()](https://dom.spec.whatwg.org/#dom-range-comparepoint), [intersectsNode()](https://dom.spec.whatwg.org/#dom-range-intersectsnode), and [isPointInRange()](https://dom.spec.whatwg.org/#dom-range-ispointinrange) have been added.
* [detach()](https://dom.spec.whatwg.org/#dom-range-detach) is now a no-op.
* [toString](https://dom.spec.whatwg.org/#dom-range-stringifier) is now defined through IDL.

**8.4. DOM Traversal**

These are the changes made to the features described in the "Document Object Model Traversal" chapter of *DOM Level 2 Traversal and Range*.

* [createNodeIterator()](https://dom.spec.whatwg.org/#dom-document-createnodeiterator) and [createTreeWalker()](https://dom.spec.whatwg.org/#dom-document-createtreewalker) now have optional arguments and lack a fourth argument which is no longer relevant given entity references never made it into the DOM.
* The *expandEntityReferences* attribute has been removed from the [NodeIterator](https://dom.spec.whatwg.org/#nodeiterator) and [TreeWalker](https://dom.spec.whatwg.org/#treewalker) interfaces for the aforementioned reason.
* The [referenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-referencenode) and [pointerBeforeReferenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-pointerbeforereferencenode) attributes have been added to [NodeIterator](https://dom.spec.whatwg.org/#nodeiterator) objects to align with proprietary extensions of implementations.
* [nextNode()](https://dom.spec.whatwg.org/#dom-nodeiterator-nextnode) and [previousNode()](https://dom.spec.whatwg.org/#dom-nodeiterator-previousnode) now throw when invoked from a [NodeFilter](https://dom.spec.whatwg.org/#callbackdef-nodefilter) to align with user agents.
* [detach()](https://dom.spec.whatwg.org/#dom-nodeiterator-detach) is now a no-op.

**Acknowledgments**

There have been a lot of people that have helped make DOM more interoperable over the years and thereby furthered the goals of this standard. Likewise many people have helped making this standard what it is today.

With that, many thanks to Adam Klein, Adrian Bateman, Aleksey Shvayka, Alex Komoroske, Alex Russell, Anthony Ramine, Arkadiusz Michalski, Arnaud Le Hors, Arun Ranganathan, Björn Höhrmann, Boris Zbarsky, Brandon Payton, Brandon Slade, Brandon Wallace, Brian Kardell, Cameron McCormack, Chris Dumez, Chris Paris, Chris Rebert, Daniel Glazman, Darin Fisher, David Bruant, David Flanagan, David Håsäther, David Hyatt, Deepak Sherveghar, Dethe Elza, Dimitri Glazkov, Domenic Denicola, Dominic Cooney, Dominique Hazaël-Massieux, Don Jordan, Doug Schepers, Edward O’Connor, Elisée Maurer Elliott Sprehn, Eric Bidelman, Erik Arvidsson, Gavin Nicol, Geoffrey Sneddon, Giorgio Liscio, Glen Huang, Glenn Adams, Glenn Maynard, Hajime Morrita, Harald Alvestrand, Hayato Ito, Henri Sivonen, Hunan Rostomyan, Ian Hickson, Igor Bukanov, Jacob Rossi, Jake Archibald, Jake Verbaten, James Graham, James Greene, James Robinson, Jeffrey Yasskin, Jens Lindström, Jesse McCarthy, João Eiras, Joe Kesselman, John Atkins, Jonas Sicking, Jonathan Robie, Joris van der Wel, Joshua Bell, Jungkee Song, Justin Summerlin, 呂康豪 (Kang-Hao Lu), Kevin Sweeney, Koji Ishii, Lachlan Hunt, Lauren Wood, Malte Ubl, Manish Goregaokar, Manish Tripathi, Marcos Caceres, Mark Miller, Mats Palmgren, Mounir Lamouri, Michael™ Smith, Mike Champion, Mike Taylor, Ojan Vafai, Oliver Nightingale, Olli Pettay, Ondřej Žára, Peter Sharpe, Philip Jägenstedt, Philippe Le Hégaret, Rafael Weinstein, Richard Bradshaw, Rick Byers, Rick Waldron, Robbert Broersma, Robin Berjon, Roland Steiner, Rune F. Halvorsen, Ruud Steltenpool, Ryosuke Niwa, Sam Dutton, Samuel Giles, Sebastian Mayr, Seo Sanghyeon, Sergey G. Grekhov, Shiki Okasaka, Shinya Kawanaka, Simon Pieters, Steve Byrne, Stig Halvorsen, Tab Atkins, Takashi Sakamoto, Takayoshi Kochi, *timeless*, Timo Tijhof, Tobie Langel, Tom Pixley, Travis Leithead, *triple-underscore*, Veli Şenol, Vidur Apparao, Warren He, Yehuda Katz, Yoav Weiss, Yoichi Osato, Yoshinori Sano, and Zack Weinberg for being awesome!

This standard is written by [Anne van Kesteren](https://annevankesteren.nl/) ([Mozilla](https://www.mozilla.org/), [annevk@annevk.nl](mailto:annevk@annevk.nl)) with substantial contributions from Aryeh Gregor ([Mozilla](https://www.mozilla.org/), [ayg@aryeh.name](mailto:ayg@aryeh.name)) and Ms2ger ([Mozilla](https://www.mozilla.org/), [ms2ger@gmail.com](mailto:ms2ger@gmail.com)).

Part of the revision history of the integration points related to [custom](https://dom.spec.whatwg.org/#concept-element-custom) elements can be found in [the w3c/webcomponents repository](https://github.com/w3c/webcomponents), which is available under the [W3C Permissive Document License](https://www.w3.org/Consortium/Legal/2015/copyright-software-and-document).

Per [CC0](https://creativecommons.org/publicdomain/zero/1.0/), to the extent possible under law, the editors have waived all copyright and related or neighboring rights to this work.

[File an issue about the selected text](https://github.com/whatwg/dom/issues/new?title=%22%0D%20%0D%20WHATWG%0D%20DOM%0D%20Living%20Standard%20%E2%80%94%20Last%20Updated...%22&body=https%3A%2F%2Fdom.spec.whatwg.org%2F%0A%0A%3E%20%0A%3E%20%0A%3E%20WHATWG%0A%3E%20DOM%0A%3E%20Living%20Standard%20%E2%80%94%20Last%20Updated%2026%20April%202016%0A%3E%20%0A%3E%20Participate%3A%E2%80%8B%0A%3E%20%20%20%20%20GitHub%20whatwg%2Fdom%20%5C%28new%20issue%2C%20open%20issues%2C%20legacy%20open%20bugs%5C%29%20%0A%3E%20%20%20%20%20IRC%3A%E2%80%8B%20%5C%23whatwg%20on%20Freenode%20%0A%3E%20Commits%3A%E2%80%8B%0A%3E%20%20%20%20%20GitHub%20whatwg%2Fdom%2Fcommits%20%0A%3E%20%20%20%20%20%40%E2%80%8Bthedomstandard%20%0A%3E%20%0A%3E%20Abstract%0A%3E%20%0A%3E%20DOM%20defines%20a%20platform%5C-neutral%20model%20for%20events%20and%20node%20trees%5C.%0A%3E%20Table%20of%20Contents%0A%3E%20%0A%3E%20%20%20%20%20Goals%0A%3E%20%20%20%20%201%20Conformance%0A%3E%20%20%20%20%20%20%20%20%201%5C.1%20Dependencies%0A%3E%20%20%20%20%20%20%20%20%201%5C.2%20Extensibility%20%0A%3E%20%20%20%20%202%20Terminology%0A%3E%20%20%20%20%20%20%20%20%202%5C.1%20Trees%0A%3E%20%20%20%20%20%20%20%20%202%5C.2%20Strings%0A%3E%20%20%20%20%20%20%20%20%202%5C.3%20Ordered%20sets%0A%3E%20%20%20%20%20%20%20%20%202%5C.4%20Selectors%0A%3E%20%20%20%20%20%20%20%20%202%5C.5%20Namespaces%20%0A%3E%20%20%20%20%203%20Events%0A%3E%20%20%20%20%20%20%20%20%203%5C.1%20Introduction%20to%20%22DOM%20Events%22%0A%3E%20%20%20%20%20%20%20%20%203%5C.2%20Interface%20Event%0A%3E%20%20%20%20%20%20%20%20%203%5C.3%20Interface%20CustomEvent%0A%3E%20%20%20%20%20%20%20%20%203%5C.4%20Constructing%20events%0A%3E%20%20%20%20%20%20%20%20%203%5C.5%20Defining%20event%20interfaces%0A%3E%20%20%20%20%20%20%20%20%203%5C.6%20Interface%20EventTarget%0A%3E%20%20%20%20%20%20%20%20%203%5C.7%20Observing%20event%20listeners%0A%3E%20%20%20%20%20%20%20%20%203%5C.8%20Dispatching%20events%0A%3E%20%20%20%20%20%20%20%20%203%5C.9%20Firing%20events%0A%3E%20%20%20%20%20%20%20%20%203%5C.10%20Action%20versus%20occurrence%20%0A%3E%20%20%20%20%204%20Nodes%0A%3E%20%20%20%20%20%20%20%20%204%5C.1%20Introduction%20t)

**Index**

**Terms defined by this specification**

* [acceptNode(node)](https://dom.spec.whatwg.org/#dom-nodefilter-acceptnode), in §6.3
* [add()](https://dom.spec.whatwg.org/#dom-domtokenlist-add), in §7.1
* [addedNodes](https://dom.spec.whatwg.org/#dom-mutationrecord-addednodes), in §4.3.3
* [AddEventListenerOptions](https://dom.spec.whatwg.org/#dictdef-addeventlisteneroptions), in §3.6
* [addEventListener(type, callback)](https://dom.spec.whatwg.org/#dom-eventtarget-addeventlistener), in §3.6
* [addEventListener(type, callback, options)](https://dom.spec.whatwg.org/#dom-eventtarget-addeventlistener), in §3.6
* [add(tokens)](https://dom.spec.whatwg.org/#dom-domtokenlist-add), in §7.1
* [adopt](https://dom.spec.whatwg.org/#concept-node-adopt), in §4.5
* [adopting steps](https://dom.spec.whatwg.org/#concept-node-adopt-ext), in §4.5
* [adoptNode(node)](https://dom.spec.whatwg.org/#dom-document-adoptnode), in §4.5
* [after](https://dom.spec.whatwg.org/#concept-range-bp-after), in §5.2
* [after(nodes)](https://dom.spec.whatwg.org/#dom-childnode-after), in §4.2.8
* [after(nodes...)](https://dom.spec.whatwg.org/#dom-childnode-after), in §4.2.8
* [ancestor](https://dom.spec.whatwg.org/#concept-tree-ancestor), in §2.1
* [append](https://dom.spec.whatwg.org/#concept-node-append), in §4.2.3
* [append an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-append), in §4.9
* [appendChild(node)](https://dom.spec.whatwg.org/#dom-node-appendchild), in §4.4
* [appendData(data)](https://dom.spec.whatwg.org/#dom-characterdata-appenddata), in §4.10
* [append(nodes)](https://dom.spec.whatwg.org/#dom-parentnode-append), in §4.2.6
* [append(nodes...)](https://dom.spec.whatwg.org/#dom-parentnode-append), in §4.2.6
* [ASCII case-insensitive](https://dom.spec.whatwg.org/#ascii-case-insensitive), in §2.2
* [ASCII case-insensitively](https://dom.spec.whatwg.org/#ascii-case-insensitive), in §2.2
* [assign a slot](https://dom.spec.whatwg.org/#assign-a-slot), in §4.2.2.4
* [assigned](https://dom.spec.whatwg.org/#slotable-assigned), in §4.2.2.2
* [assigned nodes](https://dom.spec.whatwg.org/#slot-assigned-nodes), in §4.2.2.1
* [assigned slot](https://dom.spec.whatwg.org/#slotable-assigned-slot), in §4.2.2.2
* [assignedSlot](https://dom.spec.whatwg.org/#dom-slotable-assignedslot), in §4.2.9
* [assign slotables](https://dom.spec.whatwg.org/#assign-slotables), in §4.2.2.4
* [assign slotables for a tree](https://dom.spec.whatwg.org/#assign-slotables-for-a-tree), in §4.2.2.4
* [attachShadow(init)](https://dom.spec.whatwg.org/#dom-element-attachshadow), in §4.9
* [AT\_TARGET](https://dom.spec.whatwg.org/#dom-event-at_target), in §3.2
* [Attr](https://dom.spec.whatwg.org/#attr), in §4.9.2
* [attribute](https://dom.spec.whatwg.org/#concept-attribute), in §4.9.2
* [attribute change steps](https://dom.spec.whatwg.org/#concept-element-attributes-change-ext), in §4.9
* [attributeFilter](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributefilter), in §4.3.1
* attribute list
  + [dfn for Element](https://dom.spec.whatwg.org/#concept-element-attribute), in §4.9
  + [dfn for NamedNodeMap](https://dom.spec.whatwg.org/#concept-namednodemap-attribute), in §4.9.1
* [attributeName](https://dom.spec.whatwg.org/#dom-mutationrecord-attributename), in §4.3.3
* [attributeNamespace](https://dom.spec.whatwg.org/#dom-mutationrecord-attributenamespace), in §4.3.3
* [ATTRIBUTE\_NODE](https://dom.spec.whatwg.org/#dom-node-attribute_node), in §4.4
* [attributeOldValue](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributeoldvalue), in §4.3.1
* attributes
  + [dict-member for MutationObserverInit](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributes), in §4.3.1
  + [attribute for Element](https://dom.spec.whatwg.org/#dom-element-attributes), in §4.9
  + [attribute for Node](https://dom.spec.whatwg.org/#dom-node-attributes), in §8.2
* [baseURI](https://dom.spec.whatwg.org/#dom-node-baseuri), in §4.4
* [base URL](https://dom.spec.whatwg.org/#concept-document-base-url), in §4.5
* [before](https://dom.spec.whatwg.org/#concept-range-bp-before), in §5.2
* [before(nodes)](https://dom.spec.whatwg.org/#dom-childnode-before), in §4.2.8
* [before(nodes...)](https://dom.spec.whatwg.org/#dom-childnode-before), in §4.2.8
* [boundary point](https://dom.spec.whatwg.org/#concept-range-bp), in §5.2
* bubbles
  + [dict-member for EventInit](https://dom.spec.whatwg.org/#dom-eventinit-bubbles), in §3.2
  + [attribute for Event](https://dom.spec.whatwg.org/#dom-event-bubbles), in §3.2
* [BUBBLING\_PHASE](https://dom.spec.whatwg.org/#dom-event-bubbling_phase), in §3.2
* [callback](https://dom.spec.whatwg.org/#concept-mo-callback), in §4.3.1
* cancelable
  + [dict-member for EventInit](https://dom.spec.whatwg.org/#dom-eventinit-cancelable), in §3.2
  + [attribute for Event](https://dom.spec.whatwg.org/#dom-event-cancelable), in §3.2
* [canceled flag](https://dom.spec.whatwg.org/#canceled-flag), in §3.2
* [capture](https://dom.spec.whatwg.org/#dom-eventlisteneroptions-capture), in §3.6
* [CAPTURING\_PHASE](https://dom.spec.whatwg.org/#dom-event-capturing_phase), in §3.2
* [case-sensitive](https://dom.spec.whatwg.org/#case-sensitive), in §2.2
* [case-sensitively](https://dom.spec.whatwg.org/#case-sensitive), in §2.2
* [CDATASection](https://dom.spec.whatwg.org/#cdatasection), in §8.2
* [CDATA\_SECTION\_NODE](https://dom.spec.whatwg.org/#dom-node-cdata_section_node), in §4.4
* [change an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-change), in §4.9
* [characterData](https://dom.spec.whatwg.org/#dom-mutationobserverinit-characterdata), in §4.3.1
* [CharacterData](https://dom.spec.whatwg.org/#characterdata), in §4.10
* [characterDataOldValue](https://dom.spec.whatwg.org/#dom-mutationobserverinit-characterdataoldvalue), in §4.3.1
* [characterSet](https://dom.spec.whatwg.org/#dom-document-characterset), in §4.5
* [charset](https://dom.spec.whatwg.org/#dom-document-charset), in §4.5
* [child](https://dom.spec.whatwg.org/#concept-tree-child), in §2.1
* [childElementCount](https://dom.spec.whatwg.org/#dom-parentnode-childelementcount), in §4.2.6
* [childList](https://dom.spec.whatwg.org/#dom-mutationobserverinit-childlist), in §4.3.1
* [ChildNode](https://dom.spec.whatwg.org/#childnode), in §4.2.8
* [childNodes](https://dom.spec.whatwg.org/#dom-node-childnodes), in §4.4
* children
  + [dfn for tree](https://dom.spec.whatwg.org/#concept-tree-child), in §2.1
  + [attribute for ParentNode](https://dom.spec.whatwg.org/#dom-parentnode-children), in §4.2.6
* [class](https://dom.spec.whatwg.org/#concept-class), in §4.9
* [classList](https://dom.spec.whatwg.org/#dom-element-classlist), in §4.9
* [className](https://dom.spec.whatwg.org/#dom-element-classname), in §4.9
* [clone](https://dom.spec.whatwg.org/#concept-node-clone), in §4.4
* [clone a node](https://dom.spec.whatwg.org/#concept-node-clone), in §4.4
* [cloneContents()](https://dom.spec.whatwg.org/#dom-range-clonecontents), in §5.2
* [cloneNode()](https://dom.spec.whatwg.org/#dom-node-clonenode), in §4.4
* [cloneNode(deep)](https://dom.spec.whatwg.org/#dom-node-clonenode), in §4.4
* [cloneRange()](https://dom.spec.whatwg.org/#dom-range-clonerange), in §5.2
* [clone the contents of a range](https://dom.spec.whatwg.org/#concept-range-clone), in §5.2
* [cloning steps](https://dom.spec.whatwg.org/#concept-node-clone-ext), in §4.4
* ["closed"](https://dom.spec.whatwg.org/#dom-shadowrootmode-closed), in §4.8
* [closest(selectors)](https://dom.spec.whatwg.org/#dom-element-closest), in §4.9
* [collapse()](https://dom.spec.whatwg.org/#dom-range-collapse), in §5.2
* [collapsed](https://dom.spec.whatwg.org/#dom-range-collapsed), in §5.2
* [collapse(toStart)](https://dom.spec.whatwg.org/#dom-range-collapse), in §5.2
* [collect a code point sequence](https://dom.spec.whatwg.org/#collect-a-code-point-sequence), in §2.3
* [collection](https://dom.spec.whatwg.org/#concept-collection), in §4.2.10
* [Comment](https://dom.spec.whatwg.org/#comment), in §4.13
* [Comment(data)](https://dom.spec.whatwg.org/#dom-comment-comment), in §4.13
* [COMMENT\_NODE](https://dom.spec.whatwg.org/#dom-node-comment_node), in §4.4
* [commonAncestorContainer](https://dom.spec.whatwg.org/#dom-range-commonancestorcontainer), in §5.2
* [compareBoundaryPoints(how, sourceRange)](https://dom.spec.whatwg.org/#dom-range-compareboundarypoints), in §5.2
* [compareDocumentPosition(other)](https://dom.spec.whatwg.org/#dom-node-comparedocumentposition), in §4.4
* [comparePoint(node, offset)](https://dom.spec.whatwg.org/#dom-range-comparepoint), in §5.2
* [compatMode](https://dom.spec.whatwg.org/#dom-document-compatmode), in §4.5
* [constructor](https://dom.spec.whatwg.org/#concept-event-constructor), in §3.4
* [contained](https://dom.spec.whatwg.org/#contained), in §5.2
* [contains(other)](https://dom.spec.whatwg.org/#dom-node-contains), in §4.4
* [contains(token)](https://dom.spec.whatwg.org/#dom-domtokenlist-contains), in §7.1
* [contentType](https://dom.spec.whatwg.org/#dom-document-contenttype), in §4.5
* [content type](https://dom.spec.whatwg.org/#concept-document-content-type), in §4.5
* [context object](https://dom.spec.whatwg.org/#context-object), in §2
* [contiguous Text nodes](https://dom.spec.whatwg.org/#contiguous-text-nodes), in §4.11
* [converted to ASCII lowercase](https://dom.spec.whatwg.org/#converted-to-ascii-lowercase), in §2.2
* [converted to ASCII uppercase](https://dom.spec.whatwg.org/#converted-to-ascii-uppercase), in §2.2
* [converting nodes into a node](https://dom.spec.whatwg.org/#converting-nodes-into-a-node), in §4.2.6
* [create an element](https://dom.spec.whatwg.org/#concept-create-element), in §4.9
* [createAttribute(localName)](https://dom.spec.whatwg.org/#dom-document-createattribute), in §4.5
* [createAttributeNS(namespace, qualifiedName)](https://dom.spec.whatwg.org/#dom-document-createattributens), in §4.5
* [createCDATASection()](https://dom.spec.whatwg.org/#dom-document-createcdatasection), in §8.2
* [createComment(data)](https://dom.spec.whatwg.org/#dom-document-createcomment), in §4.5
* [createDocumentFragment()](https://dom.spec.whatwg.org/#dom-document-createdocumentfragment), in §4.5
* [createDocument(namespace, qualifiedName)](https://dom.spec.whatwg.org/#dom-domimplementation-createdocument), in §4.5.1
* [createDocument(namespace, qualifiedName, doctype)](https://dom.spec.whatwg.org/#dom-domimplementation-createdocument), in §4.5.1
* [createDocumentType(qualifiedName, publicId, systemId)](https://dom.spec.whatwg.org/#dom-domimplementation-createdocumenttype), in §4.5.1
* [createElement(localName)](https://dom.spec.whatwg.org/#dom-document-createelement), in §4.5
* [createElement(localName, options)](https://dom.spec.whatwg.org/#dom-document-createelement), in §4.5
* [createElementNS(namespace, qualifiedName)](https://dom.spec.whatwg.org/#dom-document-createelementns), in §4.5
* [createElementNS(namespace, qualifiedName, options)](https://dom.spec.whatwg.org/#dom-document-createelementns), in §4.5
* [createEntityReference()](https://dom.spec.whatwg.org/#dom-document-createentityreference), in §8.2
* [createEvent(interface)](https://dom.spec.whatwg.org/#dom-document-createevent), in §4.5
* [createHTMLDocument()](https://dom.spec.whatwg.org/#dom-domimplementation-createhtmldocument), in §4.5.1
* [createHTMLDocument(title)](https://dom.spec.whatwg.org/#dom-domimplementation-createhtmldocument), in §4.5.1
* [createNodeIterator(root)](https://dom.spec.whatwg.org/#dom-document-createnodeiterator), in §4.5
* [createNodeIterator(root, whatToShow)](https://dom.spec.whatwg.org/#dom-document-createnodeiterator), in §4.5
* [createNodeIterator(root, whatToShow, filter)](https://dom.spec.whatwg.org/#dom-document-createnodeiterator), in §4.5
* [createProcessingInstruction(target, data)](https://dom.spec.whatwg.org/#dom-document-createprocessinginstruction), in §4.5
* [createRange()](https://dom.spec.whatwg.org/#dom-document-createrange), in §4.5
* [createTextNode(data)](https://dom.spec.whatwg.org/#dom-document-createtextnode), in §4.5
* [createTreeWalker(root)](https://dom.spec.whatwg.org/#dom-document-createtreewalker), in §4.5
* [createTreeWalker(root, whatToShow)](https://dom.spec.whatwg.org/#dom-document-createtreewalker), in §4.5
* [createTreeWalker(root, whatToShow, filter)](https://dom.spec.whatwg.org/#dom-document-createtreewalker), in §4.5
* [creating an element](https://dom.spec.whatwg.org/#concept-create-element), in §4.9
* [currentNode](https://dom.spec.whatwg.org/#dom-treewalker-currentnode), in §6.2
* [currentTarget](https://dom.spec.whatwg.org/#dom-event-currenttarget), in §3.2
* [custom](https://dom.spec.whatwg.org/#concept-element-custom), in §4.9
* [custom element state](https://dom.spec.whatwg.org/#concept-element-custom-element-state), in §4.9
* [CustomEvent](https://dom.spec.whatwg.org/#customevent), in §3.3
* [CustomEventInit](https://dom.spec.whatwg.org/#dictdef-customeventinit), in §3.3
* [CustomEvent(type)](https://dom.spec.whatwg.org/#dom-customevent-customevent), in §3.3
* [CustomEvent(type, eventInitDict)](https://dom.spec.whatwg.org/#dom-customevent-customevent), in §3.3
* data
  + [dfn for CharacterData, Text, Comment, ProcessingInstruction](https://dom.spec.whatwg.org/#concept-cd-data), in §4.10
  + [attribute for CharacterData](https://dom.spec.whatwg.org/#dom-characterdata-data), in §4.10
* [defaultPrevented](https://dom.spec.whatwg.org/#dom-event-defaultprevented), in §3.2
* [defined](https://dom.spec.whatwg.org/#concept-element-defined), in §4.9
* [deleteContents()](https://dom.spec.whatwg.org/#dom-range-deletecontents), in §5.2
* [deleteData(offset, count)](https://dom.spec.whatwg.org/#dom-characterdata-deletedata), in §4.10
* [descendant](https://dom.spec.whatwg.org/#concept-tree-descendant), in §2.1
* detach()
  + [method for Range](https://dom.spec.whatwg.org/#dom-range-detach), in §5.2
  + [method for NodeIterator](https://dom.spec.whatwg.org/#dom-nodeiterator-detach), in §6.1
* detail
  + [dict-member for CustomEventInit](https://dom.spec.whatwg.org/#dom-customeventinit-detail), in §3.3
  + [attribute for CustomEvent](https://dom.spec.whatwg.org/#dom-customevent-detail), in §3.3
* [disconnect()](https://dom.spec.whatwg.org/#dom-mutationobserver-disconnect), in §4.3.1
* [dispatch](https://dom.spec.whatwg.org/#concept-event-dispatch), in §3.8
* [dispatchEvent(event)](https://dom.spec.whatwg.org/#dom-eventtarget-dispatchevent), in §3.6
* [dispatch flag](https://dom.spec.whatwg.org/#dispatch-flag), in §3.2
* doctype
  + [attribute for Document](https://dom.spec.whatwg.org/#dom-document-doctype), in §4.5
  + [definition of](https://dom.spec.whatwg.org/#concept-doctype), in §4.6
* [Document](https://dom.spec.whatwg.org/#document), in §4.5
* [document](https://dom.spec.whatwg.org/#concept-document), in §4.5
* [Document()](https://dom.spec.whatwg.org/#dom-document-document), in §4.5
* [documentElement](https://dom.spec.whatwg.org/#dom-document-documentelement), in §4.5
* [document element](https://dom.spec.whatwg.org/#document-element), in §4.2.1
* [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment), in §4.7
* [DocumentFragment()](https://dom.spec.whatwg.org/#dom-documentfragment-documentfragment), in §4.7
* [DOCUMENT\_FRAGMENT\_NODE](https://dom.spec.whatwg.org/#dom-node-document_fragment_node), in §4.4
* [DOCUMENT\_NODE](https://dom.spec.whatwg.org/#dom-node-document_node), in §4.4
* [DocumentOrShadowRoot](https://dom.spec.whatwg.org/#documentorshadowroot), in §4.2.5
* [DOCUMENT\_POSITION\_CONTAINED\_BY](https://dom.spec.whatwg.org/#dom-node-document_position_contained_by), in §4.4
* [DOCUMENT\_POSITION\_CONTAINS](https://dom.spec.whatwg.org/#dom-node-document_position_contains), in §4.4
* [DOCUMENT\_POSITION\_DISCONNECTED](https://dom.spec.whatwg.org/#dom-node-document_position_disconnected), in §4.4
* [DOCUMENT\_POSITION\_FOLLOWING](https://dom.spec.whatwg.org/#dom-node-document_position_following), in §4.4
* [DOCUMENT\_POSITION\_IMPLEMENTATION\_SPECIFIC](https://dom.spec.whatwg.org/#dom-node-document_position_implementation_specific), in §4.4
* [DOCUMENT\_POSITION\_PRECEDING](https://dom.spec.whatwg.org/#dom-node-document_position_preceding), in §4.4
* [document tree](https://dom.spec.whatwg.org/#concept-document-tree), in §4.2.1
* [DocumentType](https://dom.spec.whatwg.org/#documenttype), in §4.6
* [DOCUMENT\_TYPE\_NODE](https://dom.spec.whatwg.org/#dom-node-document_type_node), in §4.4
* [documentURI](https://dom.spec.whatwg.org/#dom-document-documenturi), in §4.5
* [domConfig](https://dom.spec.whatwg.org/#dom-document-domconfig), in §8.2
* [DOMConfiguration](https://dom.spec.whatwg.org/#domconfiguration), in §8.2
* [DOMError](https://dom.spec.whatwg.org/#domerror), in §8.2
* [DOMErrorHandler](https://dom.spec.whatwg.org/#domerrorhandler), in §8.2
* [DOMImplementation](https://dom.spec.whatwg.org/#domimplementation), in §4.5.1
* [DOMImplementationList](https://dom.spec.whatwg.org/#domimplementationlist), in §8.2
* [DOMImplementationSource](https://dom.spec.whatwg.org/#domimplementationsource), in §8.2
* [DOMLocator](https://dom.spec.whatwg.org/#domlocator), in §8.2
* [DOMObject](https://dom.spec.whatwg.org/#domobject), in §8.2
* [DOMStringList](https://dom.spec.whatwg.org/#domstringlist), in §8.2
* [DOMTokenList](https://dom.spec.whatwg.org/#domtokenlist), in §7.1
* [DOMUserData](https://dom.spec.whatwg.org/#domuserdata), in §8.2
* element
  + [definition of](https://dom.spec.whatwg.org/#concept-element), in §4.9
  + [dfn for NamedNodeMap](https://dom.spec.whatwg.org/#concept-namednodemap-element), in §4.9.1
  + [dfn for Attr](https://dom.spec.whatwg.org/#concept-attribute-element), in §4.9.2
* [Element](https://dom.spec.whatwg.org/#element), in §4.9
* [ElementCreationOptions](https://dom.spec.whatwg.org/#dictdef-elementcreationoptions), in §4.5
* [element interface](https://dom.spec.whatwg.org/#concept-element-interface), in §4.5
* [ELEMENT\_NODE](https://dom.spec.whatwg.org/#dom-node-element_node), in §4.4
* [empty](https://dom.spec.whatwg.org/#concept-node-empty), in §4.2
* [encoding](https://dom.spec.whatwg.org/#concept-document-encoding), in §4.5
* [end](https://dom.spec.whatwg.org/#concept-range-end), in §5.2
* [endContainer](https://dom.spec.whatwg.org/#dom-range-endcontainer), in §5.2
* [end node](https://dom.spec.whatwg.org/#concept-range-end-node), in §5.2
* [end offset](https://dom.spec.whatwg.org/#concept-range-end-offset), in §5.2
* [endOffset](https://dom.spec.whatwg.org/#dom-range-endoffset), in §5.2
* [END\_TO\_END](https://dom.spec.whatwg.org/#dom-range-end_to_end), in §5.2
* [END\_TO\_START](https://dom.spec.whatwg.org/#dom-range-end_to_start), in §5.2
* [ensure pre-insertion validity](https://dom.spec.whatwg.org/#concept-node-ensure-pre-insertion-validity), in §4.2.3
* [entities](https://dom.spec.whatwg.org/#dom-documenttype-entities), in §8.2
* [Entity](https://dom.spec.whatwg.org/#entity), in §8.2
* [ENTITY\_NODE](https://dom.spec.whatwg.org/#dom-node-entity_node), in §4.4
* [EntityReference](https://dom.spec.whatwg.org/#entityreference), in §8.2
* [ENTITY\_REFERENCE\_NODE](https://dom.spec.whatwg.org/#dom-node-entity_reference_node), in §4.4
* [equal](https://dom.spec.whatwg.org/#concept-range-bp-equal), in §5.2
* [equals](https://dom.spec.whatwg.org/#concept-node-equals), in §4.4
* [Event](https://dom.spec.whatwg.org/#event), in §3.2
* [event](https://dom.spec.whatwg.org/#concept-event), in §3.2
* [EventInit](https://dom.spec.whatwg.org/#dictdef-eventinit), in §3.2
* [event listener](https://dom.spec.whatwg.org/#concept-event-listener), in §3.6
* [EventListener](https://dom.spec.whatwg.org/#callbackdef-eventlistener), in §3.6
* [EventListenerOptions](https://dom.spec.whatwg.org/#dictdef-eventlisteneroptions), in §3.6
* [eventPhase](https://dom.spec.whatwg.org/#dom-event-eventphase), in §3.2
* [EventTarget](https://dom.spec.whatwg.org/#eventtarget), in §3.6
* [Event(type)](https://dom.spec.whatwg.org/#dom-event-event), in §3.2
* [Event(type, eventInitDict)](https://dom.spec.whatwg.org/#dom-event-event), in §3.2
* [expandEntityReferences](https://dom.spec.whatwg.org/#dom-nodeiterator-expandentityreferences), in §8.4
* [extract](https://dom.spec.whatwg.org/#concept-range-extract), in §5.2
* [extract a range](https://dom.spec.whatwg.org/#concept-range-extract), in §5.2
* [extractContents()](https://dom.spec.whatwg.org/#dom-range-extractcontents), in §5.2
* filter
  + [dfn for traversal](https://dom.spec.whatwg.org/#concept-traversal-filter), in §6
  + [dfn for Node](https://dom.spec.whatwg.org/#concept-node-filter), in §6
  + [attribute for NodeIterator](https://dom.spec.whatwg.org/#dom-nodeiterator-filter), in §6.1
  + [attribute for TreeWalker](https://dom.spec.whatwg.org/#dom-treewalker-filter), in §6.2
* [FILTER\_ACCEPT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_accept), in §6.3
* [FILTER\_REJECT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_reject), in §6.3
* [FILTER\_SKIP](https://dom.spec.whatwg.org/#dom-nodefilter-filter_skip), in §6.3
* [find a slot](https://dom.spec.whatwg.org/#find-a-slot), in §4.2.2.3
* [find flattened slotables](https://dom.spec.whatwg.org/#find-flattened-slotables), in §4.2.2.3
* [finding a slot](https://dom.spec.whatwg.org/#find-a-slot), in §4.2.2.3
* [finding flattened slotables](https://dom.spec.whatwg.org/#find-flattened-slotables), in §4.2.2.3
* [finding slotables](https://dom.spec.whatwg.org/#find-slotables), in §4.2.2.3
* [find slotables](https://dom.spec.whatwg.org/#find-slotables), in §4.2.2.3
* [fire an event](https://dom.spec.whatwg.org/#concept-event-fire), in §3.9
* [firstChild()](https://dom.spec.whatwg.org/#dom-treewalker-firstchild), in §6.2
* [firstChild](https://dom.spec.whatwg.org/#dom-node-firstchild), in §4.4
* [first child](https://dom.spec.whatwg.org/#concept-tree-first-child), in §2.1
* [firstElementChild](https://dom.spec.whatwg.org/#dom-parentnode-firstelementchild), in §4.2.6
* [flatten](https://dom.spec.whatwg.org/#concept-flatten-options), in §3.6
* [flatten more](https://dom.spec.whatwg.org/#event-flatten-more), in §3.6
* [following](https://dom.spec.whatwg.org/#concept-tree-following), in §2.1
* [get an attribute by name](https://dom.spec.whatwg.org/#concept-element-attributes-get-by-name), in §4.9
* [get an attribute by namespace and local name](https://dom.spec.whatwg.org/#concept-element-attributes-get-by-namespace), in §4.9
* [get an attribute value](https://dom.spec.whatwg.org/#concept-element-attributes-get-value), in §4.9
* [getAttributeNames()](https://dom.spec.whatwg.org/#dom-element-getattributenames), in §4.9
* [getAttributeNodeNS(namespace, localName)](https://dom.spec.whatwg.org/#dom-element-getattributenodens), in §4.9
* [getAttributeNode(qualifiedName)](https://dom.spec.whatwg.org/#dom-element-getattributenode), in §4.9
* [getAttributeNS(namespace, localName)](https://dom.spec.whatwg.org/#dom-element-getattributens), in §4.9
* [getAttribute(qualifiedName)](https://dom.spec.whatwg.org/#dom-element-getattribute), in §4.9
* [getElementById(elementId)](https://dom.spec.whatwg.org/#dom-nonelementparentnode-getelementbyid), in §4.2.4
* getElementsByClassName(classNames)
  + [method for Document](https://dom.spec.whatwg.org/#dom-document-getelementsbyclassname), in §4.5
  + [method for Element](https://dom.spec.whatwg.org/#dom-element-getelementsbyclassname), in §4.9
* getElementsByTagNameNS(namespace, localName)
  + [method for Document](https://dom.spec.whatwg.org/#dom-document-getelementsbytagnamens), in §4.5
  + [method for Element](https://dom.spec.whatwg.org/#dom-element-getelementsbytagnamens), in §4.9
* getElementsByTagName(qualifiedName)
  + [method for Document](https://dom.spec.whatwg.org/#dom-document-getelementsbytagname), in §4.5
  + [method for Element](https://dom.spec.whatwg.org/#dom-element-getelementsbytagname), in §4.9
* getFeature()
  + [method for Node](https://dom.spec.whatwg.org/#dom-node-getfeature), in §8.2
  + [method for DOMImplementation](https://dom.spec.whatwg.org/#dom-domimplementation-getfeature), in §8.2
* [getNamedItemNS(namespace, localName)](https://dom.spec.whatwg.org/#dom-namednodemap-getnameditemns), in §4.9.1
* [getNamedItem(qualifiedName)](https://dom.spec.whatwg.org/#dom-namednodemap-getnameditem), in §4.9.1
* [get the parent](https://dom.spec.whatwg.org/#get-the-parent), in §3.6
* [getUserData()](https://dom.spec.whatwg.org/#dom-node-getuserdata), in §8.2
* [handleEvent(event)](https://dom.spec.whatwg.org/#dom-eventlistener-handleevent), in §3.6
* [has an attribute](https://dom.spec.whatwg.org/#concept-element-attribute-has), in §4.9
* [hasAttributeNS(namespace, localName)](https://dom.spec.whatwg.org/#dom-element-hasattributens), in §4.9
* [hasAttribute(qualifiedName)](https://dom.spec.whatwg.org/#dom-element-hasattribute), in §4.9
* hasAttributes()
  + [method for Element](https://dom.spec.whatwg.org/#dom-element-hasattributes), in §4.9
  + [method for Node](https://dom.spec.whatwg.org/#dom-node-hasattributes), in §8.2
* [hasChildNodes()](https://dom.spec.whatwg.org/#dom-node-haschildnodes), in §4.4
* [hasFeature()](https://dom.spec.whatwg.org/#dom-domimplementation-hasfeature), in §4.5.1
* host
  + [dfn for DocumentFragment](https://dom.spec.whatwg.org/#concept-documentfragment-host), in §4.7
  + [attribute for ShadowRoot](https://dom.spec.whatwg.org/#dom-shadowroot-host), in §4.8
* [host-including inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-host-including-inclusive-ancestor), in §4.7
* [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection), in §4.2.10.2
* [HTML document](https://dom.spec.whatwg.org/#html-document), in §4.5
* [HTML namespace](https://dom.spec.whatwg.org/#html-namespace), in §2.5
* [id](https://dom.spec.whatwg.org/#dom-element-id), in §4.9
* [ID](https://dom.spec.whatwg.org/#concept-id), in §4.9
* [implementation](https://dom.spec.whatwg.org/#dom-document-implementation), in §4.5
* [importNode(node)](https://dom.spec.whatwg.org/#dom-document-importnode), in §4.5
* [importNode(node, deep)](https://dom.spec.whatwg.org/#dom-document-importnode), in §4.5
* [in a document](https://dom.spec.whatwg.org/#in-a-document), in §4.2.1
* [in a shadow-including document](https://dom.spec.whatwg.org/#in-a-shadow-including-document), in §4.2.2
* [inclusive ancestor](https://dom.spec.whatwg.org/#concept-tree-inclusive-ancestor), in §2.1
* [inclusive descendant](https://dom.spec.whatwg.org/#concept-tree-inclusive-descendant), in §2.1
* [inclusive sibling](https://dom.spec.whatwg.org/#concept-tree-inclusive-sibling), in §2.1
* [index](https://dom.spec.whatwg.org/#concept-tree-index), in §2.1
* [initCustomEvent(type, bubbles, cancelable, detail)](https://dom.spec.whatwg.org/#dom-customevent-initcustomevent), in §3.3
* [initEvent(type, bubbles, cancelable)](https://dom.spec.whatwg.org/#dom-event-initevent), in §3.2
* [initialize](https://dom.spec.whatwg.org/#concept-event-initialize), in §3.2
* [initialized flag](https://dom.spec.whatwg.org/#initialized-flag), in §3.2
* [inner invoke](https://dom.spec.whatwg.org/#concept-event-listener-inner-invoke), in §3.8
* [in passive listener flag](https://dom.spec.whatwg.org/#in-passive-listener-flag), in §3.2
* [inputEncoding](https://dom.spec.whatwg.org/#dom-document-inputencoding), in §4.5
* insert
  + [dfn for Node](https://dom.spec.whatwg.org/#concept-node-insert), in §4.2.3
  + [dfn for Range](https://dom.spec.whatwg.org/#concept-range-insert), in §5.2
* [insert adjacent](https://dom.spec.whatwg.org/#insert-adjacent), in §4.9
* [insertAdjacentElement(where, element)](https://dom.spec.whatwg.org/#dom-element-insertadjacentelement), in §4.9
* [insertAdjacentText(where, data)](https://dom.spec.whatwg.org/#dom-element-insertadjacenttext), in §4.9
* [insertBefore(node, child)](https://dom.spec.whatwg.org/#dom-node-insertbefore), in §4.4
* [insertData(offset, data)](https://dom.spec.whatwg.org/#dom-characterdata-insertdata), in §4.10
* [insertion steps](https://dom.spec.whatwg.org/#concept-node-insert-ext), in §4.2.3
* [insertNode(node)](https://dom.spec.whatwg.org/#dom-range-insertnode), in §5.2
* [internal createElementNS steps](https://dom.spec.whatwg.org/#internal-createelementns-steps), in §4.5
* [internalSubset](https://dom.spec.whatwg.org/#dom-documenttype-internalsubset), in §8.2
* [intersectsNode(node)](https://dom.spec.whatwg.org/#dom-range-intersectsnode), in §5.2
* [invoke](https://dom.spec.whatwg.org/#concept-event-listener-invoke), in §3.8
* [is](https://dom.spec.whatwg.org/#dom-elementcreationoptions-is), in §4.5
* [isConnected](https://dom.spec.whatwg.org/#dom-node-isconnected), in §4.4
* [isDefaultNamespace(namespace)](https://dom.spec.whatwg.org/#dom-node-isdefaultnamespace), in §4.4
* [isElementContentWhitespace](https://dom.spec.whatwg.org/#dom-text-iselementcontentwhitespace), in §8.2
* [isEqualNode(otherNode)](https://dom.spec.whatwg.org/#dom-node-isequalnode), in §4.4
* [isPointInRange(node, offset)](https://dom.spec.whatwg.org/#dom-range-ispointinrange), in §5.2
* [isSameNode(otherNode)](https://dom.spec.whatwg.org/#dom-node-issamenode), in §4.4
* [isSupported](https://dom.spec.whatwg.org/#dom-node-issupported), in §8.2
* [isTrusted](https://dom.spec.whatwg.org/#dom-event-istrusted), in §3.2
* item(index)
  + [method for NodeList](https://dom.spec.whatwg.org/#dom-nodelist-item), in §4.2.10.1
  + [method for HTMLCollection](https://dom.spec.whatwg.org/#dom-htmlcollection-item), in §4.2.10.2
  + [method for NamedNodeMap](https://dom.spec.whatwg.org/#dom-namednodemap-item), in §4.9.1
  + [method for DOMTokenList](https://dom.spec.whatwg.org/#dom-domtokenlist-item), in §7.1
* [iterator collection](https://dom.spec.whatwg.org/#iterator-collection), in §6.1
* [last child](https://dom.spec.whatwg.org/#concept-tree-last-child), in §2.1
* [lastChild()](https://dom.spec.whatwg.org/#dom-treewalker-lastchild), in §6.2
* [lastChild](https://dom.spec.whatwg.org/#dom-node-lastchild), in §4.4
* [lastElementChild](https://dom.spec.whatwg.org/#dom-parentnode-lastelementchild), in §4.2.6
* length
  + [dfn for Node](https://dom.spec.whatwg.org/#concept-node-length), in §4.2
  + [attribute for NodeList](https://dom.spec.whatwg.org/#dom-nodelist-length), in §4.2.10.1
  + [attribute for HTMLCollection](https://dom.spec.whatwg.org/#dom-htmlcollection-length), in §4.2.10.2
  + [attribute for NamedNodeMap](https://dom.spec.whatwg.org/#dom-namednodemap-length), in §4.9.1
  + [attribute for CharacterData](https://dom.spec.whatwg.org/#dom-characterdata-length), in §4.10
  + [attribute for DOMTokenList](https://dom.spec.whatwg.org/#dom-domtokenlist-length), in §7.1
* [light tree](https://dom.spec.whatwg.org/#concept-light-tree), in §4.2.2
* [limited-quirks mode](https://dom.spec.whatwg.org/#concept-document-limited-quirks), in §4.5
* [list of elements with class names classNames](https://dom.spec.whatwg.org/#concept-getelementsbyclassname), in §4.4
* [list of elements with namespace namespace and local name localName](https://dom.spec.whatwg.org/#concept-getelementsbytagnamens), in §4.4
* [list of elements with qualified name qualifiedName](https://dom.spec.whatwg.org/#concept-getelementsbytagname), in §4.4
* [live](https://dom.spec.whatwg.org/#concept-collection-live), in §4.2.10
* [live collection](https://dom.spec.whatwg.org/#concept-collection-live), in §4.2.10
* localName
  + [attribute for Element](https://dom.spec.whatwg.org/#dom-element-localname), in §4.9
  + [attribute for Attr](https://dom.spec.whatwg.org/#dom-attr-localname), in §4.9.2
  + [attribute for Node](https://dom.spec.whatwg.org/#dom-node-localname), in §8.2
* local name
  + [dfn for Element](https://dom.spec.whatwg.org/#concept-element-local-name), in §4.9
  + [dfn for Attr](https://dom.spec.whatwg.org/#concept-attribute-local-name), in §4.9.2
* [locate a namespace](https://dom.spec.whatwg.org/#locate-a-namespace), in §4.4
* [locate a namespace prefix](https://dom.spec.whatwg.org/#locate-a-namespace-prefix), in §4.4
* [locating a namespace prefix](https://dom.spec.whatwg.org/#locate-a-namespace-prefix), in §4.4
* [lookupNamespaceURI(prefix)](https://dom.spec.whatwg.org/#dom-node-lookupnamespaceuri), in §4.4
* [lookupPrefix(namespace)](https://dom.spec.whatwg.org/#dom-node-lookupprefix), in §4.4
* [match a relative selectors string](https://dom.spec.whatwg.org/#match-a-relative-selectors-string), in §2.4
* [matches(selectors)](https://dom.spec.whatwg.org/#dom-element-matches), in §4.9
* mode
  + [dfn for Document](https://dom.spec.whatwg.org/#concept-document-mode), in §4.5
  + [dfn for ShadowRoot](https://dom.spec.whatwg.org/#shadowroot-mode), in §4.8
  + [attribute for ShadowRoot](https://dom.spec.whatwg.org/#dom-shadowroot-mode), in §4.8
  + [dict-member for ShadowRootInit](https://dom.spec.whatwg.org/#dom-shadowrootinit-mode), in §4.9
* [MutationCallback](https://dom.spec.whatwg.org/#callbackdef-mutationcallback), in §4.3.1
* [MutationEvent](https://dom.spec.whatwg.org/#mutationevent), in §8.1
* [MutationNameEvent](https://dom.spec.whatwg.org/#mutationnameevent), in §8.1
* [MutationObserver](https://dom.spec.whatwg.org/#mutationobserver), in §4.3.1
* [MutationObserver(callback)](https://dom.spec.whatwg.org/#dom-mutationobserver-mutationobserver), in §4.3.1
* [mutation observer compound microtask queued flag](https://dom.spec.whatwg.org/#mutation-observer-compound-microtask-queued-flag), in §4.3
* [MutationObserverInit](https://dom.spec.whatwg.org/#dictdef-mutationobserverinit), in §4.3.1
* [MutationRecord](https://dom.spec.whatwg.org/#mutationrecord), in §4.3.3
* name
  + [dfn for slot](https://dom.spec.whatwg.org/#slot-name), in §4.2.2.1
  + [dfn for slotable](https://dom.spec.whatwg.org/#slotable-name), in §4.2.2.2
  + [dfn for DocumentType](https://dom.spec.whatwg.org/#concept-doctype-name), in §4.6
  + [attribute for DocumentType](https://dom.spec.whatwg.org/#dom-documenttype-name), in §4.6
  + [attribute for Attr](https://dom.spec.whatwg.org/#dom-attr-name), in §4.9.2
* [named attribute](https://dom.spec.whatwg.org/#concept-named-attribute), in §4.9.2
* [namedItem(name)](https://dom.spec.whatwg.org/#dom-htmlcollection-nameditem), in §4.2.10.2
* [NamedNodeMap](https://dom.spec.whatwg.org/#namednodemap), in §4.9.1
* [NameList](https://dom.spec.whatwg.org/#namelist), in §8.2
* namespace
  + [dfn for Element](https://dom.spec.whatwg.org/#concept-element-namespace), in §4.9
  + [dfn for Attr](https://dom.spec.whatwg.org/#concept-attribute-namespace), in §4.9.2
* namespace prefix
  + [dfn for Element](https://dom.spec.whatwg.org/#concept-element-namespace-prefix), in §4.9
  + [dfn for Attr](https://dom.spec.whatwg.org/#concept-attribute-namespace-prefix), in §4.9.2
* namespaceURI
  + [attribute for Element](https://dom.spec.whatwg.org/#dom-element-namespaceuri), in §4.9
  + [attribute for Attr](https://dom.spec.whatwg.org/#dom-attr-namespaceuri), in §4.9.2
  + [attribute for Node](https://dom.spec.whatwg.org/#dom-node-namespaceuri), in §8.2
* [nextElementSibling](https://dom.spec.whatwg.org/#dom-nondocumenttypechildnode-nextelementsibling), in §4.2.7
* nextNode()
  + [method for NodeIterator](https://dom.spec.whatwg.org/#dom-nodeiterator-nextnode), in §6.1
  + [method for TreeWalker](https://dom.spec.whatwg.org/#dom-treewalker-nextnode), in §6.2
* [next sibling](https://dom.spec.whatwg.org/#concept-tree-next-sibling), in §2.1
* [nextSibling()](https://dom.spec.whatwg.org/#dom-treewalker-nextsibling), in §6.2
* nextSibling
  + [attribute for MutationRecord](https://dom.spec.whatwg.org/#dom-mutationrecord-nextsibling), in §4.3.3
  + [attribute for Node](https://dom.spec.whatwg.org/#dom-node-nextsibling), in §4.4
* [Node](https://dom.spec.whatwg.org/#node), in §4.4
* [node document](https://dom.spec.whatwg.org/#concept-node-document), in §4.4
* [NodeFilter](https://dom.spec.whatwg.org/#callbackdef-nodefilter), in §6.3
* [NodeIterator](https://dom.spec.whatwg.org/#nodeiterator), in §6.1
* [NodeIterator pre-removing steps](https://dom.spec.whatwg.org/#nodeiterator-pre-removing-steps), in §6.1
* [NodeList](https://dom.spec.whatwg.org/#nodelist), in §4.2.10.1
* nodeName
  + [attribute for Node](https://dom.spec.whatwg.org/#dom-node-nodename), in §4.4
  + [attribute for Attr](https://dom.spec.whatwg.org/#dom-attr-nodename), in §4.9.2
* [nodes](https://dom.spec.whatwg.org/#concept-node), in §4.2
* [node tree](https://dom.spec.whatwg.org/#concept-node-tree), in §4.2
* [nodeType](https://dom.spec.whatwg.org/#dom-node-nodetype), in §4.4
* nodeValue
  + [attribute for Node](https://dom.spec.whatwg.org/#dom-node-nodevalue), in §4.4
  + [attribute for Attr](https://dom.spec.whatwg.org/#dom-attr-nodevalue), in §4.9.2
* [NonDocumentTypeChildNode](https://dom.spec.whatwg.org/#nondocumenttypechildnode), in §4.2.7
* [NONE](https://dom.spec.whatwg.org/#dom-event-none), in §3.2
* [NonElementParentNode](https://dom.spec.whatwg.org/#nonelementparentnode), in §4.2.4
* [no-quirks mode](https://dom.spec.whatwg.org/#concept-document-no-quirks), in §4.5
* [normalize()](https://dom.spec.whatwg.org/#dom-node-normalize), in §4.4
* [normalizeDocument()](https://dom.spec.whatwg.org/#dom-document-normalizedocument), in §8.2
* [Notation](https://dom.spec.whatwg.org/#notation), in §8.2
* [NOTATION\_NODE](https://dom.spec.whatwg.org/#dom-node-notation_node), in §4.4
* [notations](https://dom.spec.whatwg.org/#dom-documenttype-notations), in §8.2
* [notify mutation observers](https://dom.spec.whatwg.org/#notify-mutation-observers), in §4.3
* [observe(target, options)](https://dom.spec.whatwg.org/#dom-mutationobserver-observe), in §4.3.1
* [offset](https://dom.spec.whatwg.org/#concept-range-bp-offset), in §5.2
* [oldValue](https://dom.spec.whatwg.org/#dom-mutationrecord-oldvalue), in §4.3.3
* [once](https://dom.spec.whatwg.org/#dom-addeventlisteneroptions-once), in §3.6
* ["open"](https://dom.spec.whatwg.org/#dom-shadowrootmode-open), in §4.8
* [ordered set parser](https://dom.spec.whatwg.org/#concept-ordered-set-parser), in §2.3
* [ordered set serializer](https://dom.spec.whatwg.org/#concept-ordered-set-serializer), in §2.3
* [origin](https://dom.spec.whatwg.org/#dom-document-origin), in §4.5
* [other applicable specifications](https://dom.spec.whatwg.org/#other-applicable-specifications), in §1.2
* [ownerDocument](https://dom.spec.whatwg.org/#dom-node-ownerdocument), in §4.4
* [ownerElement](https://dom.spec.whatwg.org/#dom-attr-ownerelement), in §4.9.2
* [parent](https://dom.spec.whatwg.org/#concept-tree-parent), in §2.1
* [parentElement](https://dom.spec.whatwg.org/#dom-node-parentelement), in §4.4
* [parent element](https://dom.spec.whatwg.org/#parent-element), in §4.9
* [parentNode](https://dom.spec.whatwg.org/#dom-node-parentnode), in §4.4
* [ParentNode](https://dom.spec.whatwg.org/#parentnode), in §4.2.6
* [parentNode()](https://dom.spec.whatwg.org/#dom-treewalker-parentnode), in §6.2
* [partially contained](https://dom.spec.whatwg.org/#partially-contained), in §5.2
* [participate](https://dom.spec.whatwg.org/#concept-tree-participate), in §2.1
* [participate in a tree](https://dom.spec.whatwg.org/#concept-tree-participate), in §2.1
* [participates in a tree](https://dom.spec.whatwg.org/#concept-tree-participate), in §2.1
* [passive](https://dom.spec.whatwg.org/#dom-addeventlisteneroptions-passive), in §3.6
* [pointerBeforeReferenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-pointerbeforereferencenode), in §6.1
* [position](https://dom.spec.whatwg.org/#concept-range-bp-position), in §5.2
* [preceding](https://dom.spec.whatwg.org/#concept-tree-preceding), in §2.1
* prefix
  + [attribute for Element](https://dom.spec.whatwg.org/#dom-element-prefix), in §4.9
  + [attribute for Attr](https://dom.spec.whatwg.org/#dom-attr-prefix), in §4.9.2
  + [attribute for Node](https://dom.spec.whatwg.org/#dom-node-prefix), in §8.2
* [prefix match](https://dom.spec.whatwg.org/#prefix-match), in §2.2
* [pre-insert](https://dom.spec.whatwg.org/#concept-node-pre-insert), in §4.2.3
* [prepend(nodes...)](https://dom.spec.whatwg.org/#dom-parentnode-prepend), in §4.2.6
* [prepend(nodes)](https://dom.spec.whatwg.org/#dom-parentnode-prepend), in §4.2.6
* [pre-remove](https://dom.spec.whatwg.org/#concept-node-pre-remove), in §4.2.3
* [preventDefault()](https://dom.spec.whatwg.org/#dom-event-preventdefault), in §3.2
* [previousElementSibling](https://dom.spec.whatwg.org/#dom-nondocumenttypechildnode-previouselementsibling), in §4.2.7
* previousNode()
  + [method for NodeIterator](https://dom.spec.whatwg.org/#dom-nodeiterator-previousnode), in §6.1
  + [method for TreeWalker](https://dom.spec.whatwg.org/#dom-treewalker-previousnode), in §6.2
* [previousSibling()](https://dom.spec.whatwg.org/#dom-treewalker-previoussibling), in §6.2
* [previous sibling](https://dom.spec.whatwg.org/#concept-tree-previous-sibling), in §2.1
* previousSibling
  + [attribute for MutationRecord](https://dom.spec.whatwg.org/#dom-mutationrecord-previoussibling), in §4.3.3
  + [attribute for Node](https://dom.spec.whatwg.org/#dom-node-previoussibling), in §4.4
* [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction), in §4.12
* [PROCESSING\_INSTRUCTION\_NODE](https://dom.spec.whatwg.org/#dom-node-processing_instruction_node), in §4.4
* [publicId](https://dom.spec.whatwg.org/#dom-documenttype-publicid), in §4.6
* [public ID](https://dom.spec.whatwg.org/#concept-doctype-publicid), in §4.6
* qualified name
  + [dfn for Element](https://dom.spec.whatwg.org/#concept-element-qualified-name), in §4.9
  + [dfn for Attr](https://dom.spec.whatwg.org/#concept-attribute-qualified-name), in §4.9.2
* [querySelectorAll(selectors)](https://dom.spec.whatwg.org/#dom-parentnode-queryselectorall), in §4.2.6
* [querySelector(selectors)](https://dom.spec.whatwg.org/#dom-parentnode-queryselector), in §4.2.6
* [queue a mutation observer compound microtask](https://dom.spec.whatwg.org/#queue-a-mutation-observer-compound-microtask), in §4.3
* [queue a mutation record](https://dom.spec.whatwg.org/#queue-a-mutation-record), in §4.3.2
* [quirks mode](https://dom.spec.whatwg.org/#concept-document-quirks), in §4.5
* [Range](https://dom.spec.whatwg.org/#range), in §5.2
* [Range()](https://dom.spec.whatwg.org/#dom-range-range), in §5.2
* [range](https://dom.spec.whatwg.org/#concept-range), in §5.2
* [RangeException](https://dom.spec.whatwg.org/#rangeexception), in §8.3
* [record queue](https://dom.spec.whatwg.org/#concept-mo-queue), in §4.3.1
* [referenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-referencenode), in §6.1
* [reflect](https://dom.spec.whatwg.org/#concept-reflect), in §4.9
* [registered observer](https://dom.spec.whatwg.org/#registered-observer), in §4.3
* remove()
  + [method for ChildNode](https://dom.spec.whatwg.org/#dom-childnode-remove), in §4.2.8
  + [method for DOMTokenList](https://dom.spec.whatwg.org/#dom-domtokenlist-remove), in §7.1
* [remove](https://dom.spec.whatwg.org/#concept-node-remove), in §4.2.3
* [remove an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-remove), in §4.9
* [remove an attribute by name](https://dom.spec.whatwg.org/#concept-element-attributes-remove-by-name), in §4.9
* [remove an attribute by namespace and local name](https://dom.spec.whatwg.org/#concept-element-attributes-remove-by-namespace), in §4.9
* [removeAttributeNode(attr)](https://dom.spec.whatwg.org/#dom-element-removeattributenode), in §4.9
* [removeAttributeNS(namespace, localName)](https://dom.spec.whatwg.org/#dom-element-removeattributens), in §4.9
* [removeAttribute(qualifiedName)](https://dom.spec.whatwg.org/#dom-element-removeattribute), in §4.9
* [removeChild(child)](https://dom.spec.whatwg.org/#dom-node-removechild), in §4.4
* [removedNodes](https://dom.spec.whatwg.org/#dom-mutationrecord-removednodes), in §4.3.3
* [removeEventListener(type, callback)](https://dom.spec.whatwg.org/#dom-eventtarget-removeeventlistener), in §3.6
* [removeEventListener(type, callback, options)](https://dom.spec.whatwg.org/#dom-eventtarget-removeeventlistener), in §3.6
* [removeNamedItemNS(namespace, localName)](https://dom.spec.whatwg.org/#dom-namednodemap-removenameditemns), in §4.9.1
* [removeNamedItem(qualifiedName)](https://dom.spec.whatwg.org/#dom-namednodemap-removenameditem), in §4.9.1
* [remove(tokens)](https://dom.spec.whatwg.org/#dom-domtokenlist-remove), in §7.1
* [removing steps](https://dom.spec.whatwg.org/#concept-node-remove-ext), in §4.2.3
* [renameNode()](https://dom.spec.whatwg.org/#dom-document-renamenode), in §8.2
* [replace](https://dom.spec.whatwg.org/#concept-node-replace), in §4.2.3
* [replace all](https://dom.spec.whatwg.org/#concept-node-replace-all), in §4.2.3
* [replace an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-replace), in §4.9
* [replaceChild(node, child)](https://dom.spec.whatwg.org/#dom-node-replacechild), in §4.4
* [replace data](https://dom.spec.whatwg.org/#concept-cd-replace), in §4.10
* [replaceData(offset, count, data)](https://dom.spec.whatwg.org/#dom-characterdata-replacedata), in §4.10
* [replace(token, newToken)](https://dom.spec.whatwg.org/#dom-domtokenlist-replace), in §7.1
* [replaceWholeText()](https://dom.spec.whatwg.org/#dom-text-replacewholetext), in §8.2
* [replaceWith(nodes...)](https://dom.spec.whatwg.org/#dom-childnode-replacewith), in §4.2.8
* [replaceWith(nodes)](https://dom.spec.whatwg.org/#dom-childnode-replacewith), in §4.2.8
* [represented by the collection](https://dom.spec.whatwg.org/#represented-by-the-collection), in §4.2.10
* root
  + [dfn for tree](https://dom.spec.whatwg.org/#concept-tree-root), in §2.1
  + [dfn for Range](https://dom.spec.whatwg.org/#concept-range-root), in §5.2
  + [dfn for traversal](https://dom.spec.whatwg.org/#concept-traversal-root), in §6
  + [attribute for NodeIterator](https://dom.spec.whatwg.org/#dom-nodeiterator-root), in §6.1
  + [attribute for TreeWalker](https://dom.spec.whatwg.org/#dom-treewalker-root), in §6.2
* [rootNode](https://dom.spec.whatwg.org/#dom-node-rootnode), in §4.4
* [schemaTypeInfo](https://dom.spec.whatwg.org/#dom-element-schematypeinfo), in §8.2
* [scope-match a selectors string](https://dom.spec.whatwg.org/#scope-match-a-selectors-string), in §2.4
* [select](https://dom.spec.whatwg.org/#concept-range-select), in §5.2
* [selectNodeContents(node)](https://dom.spec.whatwg.org/#dom-range-selectnodecontents), in §5.2
* [selectNode(node)](https://dom.spec.whatwg.org/#dom-range-selectnode), in §5.2
* [serialize steps](https://dom.spec.whatwg.org/#concept-dtl-serialize), in §7.1
* [set an attribute](https://dom.spec.whatwg.org/#concept-element-attributes-set), in §4.9
* [set an attribute value](https://dom.spec.whatwg.org/#concept-element-attributes-set-value), in §4.9
* [setAttributeNode(attr)](https://dom.spec.whatwg.org/#dom-element-setattributenode), in §4.9
* [setAttributeNodeNS(attr)](https://dom.spec.whatwg.org/#dom-element-setattributenodens), in §4.9
* [setAttributeNS(namespace, qualifiedName, value)](https://dom.spec.whatwg.org/#dom-element-setattributens), in §4.9
* [setAttribute(qualifiedName, value)](https://dom.spec.whatwg.org/#dom-element-setattribute), in §4.9
* [setEndAfter(node)](https://dom.spec.whatwg.org/#dom-range-setendafter), in §5.2
* [setEndBefore(node)](https://dom.spec.whatwg.org/#dom-range-setendbefore), in §5.2
* [setEnd(node, offset)](https://dom.spec.whatwg.org/#dom-range-setend), in §5.2
* [setIdAttribute()](https://dom.spec.whatwg.org/#dom-element-setidattribute), in §8.2
* [setIdAttributeNode()](https://dom.spec.whatwg.org/#dom-element-setidattributenode), in §8.2
* [setIdAttributeNS()](https://dom.spec.whatwg.org/#dom-element-setidattributens), in §8.2
* [setNamedItem(attr)](https://dom.spec.whatwg.org/#dom-namednodemap-setnameditem), in §4.9.1
* [setNamedItemNS(attr)](https://dom.spec.whatwg.org/#dom-namednodemap-setnameditemns), in §4.9.1
* [setStartAfter(node)](https://dom.spec.whatwg.org/#dom-range-setstartafter), in §5.2
* [setStartBefore(node)](https://dom.spec.whatwg.org/#dom-range-setstartbefore), in §5.2
* [setStart(node, offset)](https://dom.spec.whatwg.org/#dom-range-setstart), in §5.2
* [set the end](https://dom.spec.whatwg.org/#concept-range-bp-set), in §5.2
* [set the start](https://dom.spec.whatwg.org/#concept-range-bp-set), in §5.2
* [setUserData()](https://dom.spec.whatwg.org/#dom-node-setuserdata), in §8.2
* [shadow host](https://dom.spec.whatwg.org/#element-shadow-host), in §4.9
* [shadow-including ancestor](https://dom.spec.whatwg.org/#concept-shadow-including-ancestor), in §4.8
* [shadow-including descendant](https://dom.spec.whatwg.org/#concept-shadow-including-descendant), in §4.8
* [shadow-including inclusive ancestor](https://dom.spec.whatwg.org/#concept-shadow-including-inclusive-ancestor), in §4.8
* [shadow-including inclusive descendant](https://dom.spec.whatwg.org/#concept-shadow-including-inclusive-descendant), in §4.8
* [shadow-including preorder, depth-first traversal](https://dom.spec.whatwg.org/#shadow-including-preorder-depth-first-traversal), in §4.8
* [shadow-including root](https://dom.spec.whatwg.org/#concept-shadow-including-root), in §4.8
* [shadow-including tree order](https://dom.spec.whatwg.org/#concept-shadow-including-tree-order), in §4.8
* [ShadowRoot](https://dom.spec.whatwg.org/#shadowroot), in §4.8
* shadow root
  + [definition of](https://dom.spec.whatwg.org/#concept-shadow-root), in §4.8
  + [dfn for Element](https://dom.spec.whatwg.org/#concept-element-shadow-root), in §4.9
* [shadowRoot](https://dom.spec.whatwg.org/#dom-element-shadowroot), in §4.9
* [ShadowRootInit](https://dom.spec.whatwg.org/#dictdef-shadowrootinit), in §4.9
* [ShadowRootMode](https://dom.spec.whatwg.org/#enumdef-shadowrootmode), in §4.8
* [shadow tree](https://dom.spec.whatwg.org/#concept-shadow-tree), in §4.2.2
* [SHOW\_ALL](https://dom.spec.whatwg.org/#dom-nodefilter-show_all), in §6.3
* [SHOW\_ATTRIBUTE](https://dom.spec.whatwg.org/#dom-nodefilter-show_attribute), in §6.3
* [SHOW\_CDATA\_SECTION](https://dom.spec.whatwg.org/#dom-nodefilter-show_cdata_section), in §6.3
* [SHOW\_COMMENT](https://dom.spec.whatwg.org/#dom-nodefilter-show_comment), in §6.3
* [SHOW\_DOCUMENT](https://dom.spec.whatwg.org/#dom-nodefilter-show_document), in §6.3
* [SHOW\_DOCUMENT\_FRAGMENT](https://dom.spec.whatwg.org/#dom-nodefilter-show_document_fragment), in §6.3
* [SHOW\_DOCUMENT\_TYPE](https://dom.spec.whatwg.org/#dom-nodefilter-show_document_type), in §6.3
* [SHOW\_ELEMENT](https://dom.spec.whatwg.org/#dom-nodefilter-show_element), in §6.3
* [SHOW\_ENTITY](https://dom.spec.whatwg.org/#dom-nodefilter-show_entity), in §6.3
* [SHOW\_ENTITY\_REFERENCE](https://dom.spec.whatwg.org/#dom-nodefilter-show_entity_reference), in §6.3
* [SHOW\_NOTATION](https://dom.spec.whatwg.org/#dom-nodefilter-show_notation), in §6.3
* [SHOW\_PROCESSING\_INSTRUCTION](https://dom.spec.whatwg.org/#dom-nodefilter-show_processing_instruction), in §6.3
* [SHOW\_TEXT](https://dom.spec.whatwg.org/#dom-nodefilter-show_text), in §6.3
* [sibling](https://dom.spec.whatwg.org/#concept-tree-sibling), in §2.1
* [signal a slot change](https://dom.spec.whatwg.org/#signal-a-slot-change), in §4.2.2.5
* [signal slot list](https://dom.spec.whatwg.org/#signal-slot-list), in §4.2.2.5
* [skip ASCII whitespace](https://dom.spec.whatwg.org/#skip-ascii-whitespace), in §2.3
* slot
  + [definition of](https://dom.spec.whatwg.org/#concept-slot), in §4.2.2.1
  + [attribute for Element](https://dom.spec.whatwg.org/#dom-element-slot), in §4.9
* [slotable](https://dom.spec.whatwg.org/#concept-slotable), in §4.2.2.2
* [Slotable](https://dom.spec.whatwg.org/#slotable), in §4.2.9
* [specified](https://dom.spec.whatwg.org/#dom-attr-specified), in §4.9.2
* [split a Text node](https://dom.spec.whatwg.org/#concept-text-split), in §4.11
* [splitText(offset)](https://dom.spec.whatwg.org/#dom-text-splittext), in §4.11
* [start](https://dom.spec.whatwg.org/#concept-range-start), in §5.2
* [startContainer](https://dom.spec.whatwg.org/#dom-range-startcontainer), in §5.2
* [start node](https://dom.spec.whatwg.org/#concept-range-start-node), in §5.2
* [startOffset](https://dom.spec.whatwg.org/#dom-range-startoffset), in §5.2
* [start offset](https://dom.spec.whatwg.org/#concept-range-start-offset), in §5.2
* [START\_TO\_END](https://dom.spec.whatwg.org/#dom-range-start_to_end), in §5.2
* [START\_TO\_START](https://dom.spec.whatwg.org/#dom-range-start_to_start), in §5.2
* [static collection](https://dom.spec.whatwg.org/#concept-collection-static), in §4.2.10
* [stopImmediatePropagation()](https://dom.spec.whatwg.org/#dom-event-stopimmediatepropagation), in §3.2
* [stop immediate propagation flag](https://dom.spec.whatwg.org/#stop-immediate-propagation-flag), in §3.2
* [stopPropagation()](https://dom.spec.whatwg.org/#dom-event-stoppropagation), in §3.2
* [stop propagation flag](https://dom.spec.whatwg.org/#stop-propagation-flag), in §3.2
* [strictErrorChecking](https://dom.spec.whatwg.org/#dom-document-stricterrorchecking), in §8.2
* stringification behavior
  + [dfn for Range](https://dom.spec.whatwg.org/#dom-range-stringifier), in §5.2
  + [dfn for DOMTokenList](https://dom.spec.whatwg.org/#dom-domtokenlist-stringifier), in §7.1
* [substring data](https://dom.spec.whatwg.org/#concept-cd-substring), in §4.10
* [substringData(offset, count)](https://dom.spec.whatwg.org/#dom-characterdata-substringdata), in §4.10
* [subtree](https://dom.spec.whatwg.org/#dom-mutationobserverinit-subtree), in §4.3.1
* [supported tokens](https://dom.spec.whatwg.org/#concept-supported-tokens), in §7.1
* [supports(token)](https://dom.spec.whatwg.org/#dom-domtokenlist-supports), in §7.1
* [surroundContents(newParent)](https://dom.spec.whatwg.org/#dom-range-surroundcontents), in §5.2
* [SVG namespace](https://dom.spec.whatwg.org/#svg-namespace), in §2.5
* [system ID](https://dom.spec.whatwg.org/#concept-doctype-systemid), in §4.6
* [systemId](https://dom.spec.whatwg.org/#dom-documenttype-systemid), in §4.6
* [tagName](https://dom.spec.whatwg.org/#dom-element-tagname), in §4.9
* [takeRecords()](https://dom.spec.whatwg.org/#dom-mutationobserver-takerecords), in §4.3.1
* target
  + [attribute for Event](https://dom.spec.whatwg.org/#dom-event-target), in §3.2
  + [attribute for MutationRecord](https://dom.spec.whatwg.org/#dom-mutationrecord-target), in §4.3.3
  + [dfn for ProcessingInstruction](https://dom.spec.whatwg.org/#concept-pi-target), in §4.12
  + [attribute for ProcessingInstruction](https://dom.spec.whatwg.org/#dom-processinginstruction-target), in §4.12
* [Text](https://dom.spec.whatwg.org/#text), in §4.11
* textContent
  + [attribute for Node](https://dom.spec.whatwg.org/#dom-node-textcontent), in §4.4
  + [attribute for Attr](https://dom.spec.whatwg.org/#dom-attr-textcontent), in §4.9.2
* [Text(data)](https://dom.spec.whatwg.org/#dom-text-text), in §4.11
* [TEXT\_NODE](https://dom.spec.whatwg.org/#dom-node-text_node), in §4.4
* [timeStamp](https://dom.spec.whatwg.org/#dom-event-timestamp), in §3.2
* [toggle(token)](https://dom.spec.whatwg.org/#dom-domtokenlist-toggle), in §7.1
* [toggle(token, force)](https://dom.spec.whatwg.org/#dom-domtokenlist-toggle), in §7.1
* [tokens](https://dom.spec.whatwg.org/#concept-dtl-tokens), in §7.1
* [transient registered observer](https://dom.spec.whatwg.org/#transient-registered-observer), in §4.3
* [traverse](https://dom.spec.whatwg.org/#concept-nodeiterator-traverse), in §6.1
* [traverse children](https://dom.spec.whatwg.org/#concept-traverse-children), in §6.2
* [Traverse children main step](https://dom.spec.whatwg.org/#concept-traverse-children-main), in §6.2
* [traverse siblings](https://dom.spec.whatwg.org/#concept-traverse-siblings), in §6.2
* [tree](https://dom.spec.whatwg.org/#concept-tree), in §2.1
* [tree order](https://dom.spec.whatwg.org/#concept-tree-order), in §2.1
* [TreeWalker](https://dom.spec.whatwg.org/#treewalker), in §6.2
* type
  + [attribute for Event](https://dom.spec.whatwg.org/#dom-event-type), in §3.2
  + [attribute for MutationRecord](https://dom.spec.whatwg.org/#dom-mutationrecord-type), in §4.3.3
  + [dfn for Document](https://dom.spec.whatwg.org/#concept-document-type), in §4.5
* [TypeInfo](https://dom.spec.whatwg.org/#typeinfo), in §8.2
* [update steps](https://dom.spec.whatwg.org/#concept-dtl-update), in §7.1
* URL
  + [dfn for Document](https://dom.spec.whatwg.org/#concept-document-url), in §4.5
  + [attribute for Document](https://dom.spec.whatwg.org/#dom-document-url), in §4.5
* [UserDataHandler](https://dom.spec.whatwg.org/#userdatahandler), in §8.2
* [validate](https://dom.spec.whatwg.org/#validate), in §2.5
* [validate and extract](https://dom.spec.whatwg.org/#validate-and-extract), in §2.5
* [validation steps](https://dom.spec.whatwg.org/#concept-domtokenlist-validation), in §7.1
* value
  + [dfn for Attr](https://dom.spec.whatwg.org/#concept-attribute-value), in §4.9.2
  + [attribute for Attr](https://dom.spec.whatwg.org/#dom-attr-value), in §4.9.2
  + [attribute for DOMTokenList](https://dom.spec.whatwg.org/#dom-domtokenlist-value), in §7.1
* [webkitMatchesSelector(selectors)](https://dom.spec.whatwg.org/#dom-element-webkitmatchesselector), in §4.9
* whatToShow
  + [dfn for traversal](https://dom.spec.whatwg.org/#concept-traversal-whattoshow), in §6
  + [attribute for NodeIterator](https://dom.spec.whatwg.org/#dom-nodeiterator-whattoshow), in §6.1
  + [attribute for TreeWalker](https://dom.spec.whatwg.org/#dom-treewalker-whattoshow), in §6.2
* [wholeText](https://dom.spec.whatwg.org/#dom-text-wholetext), in §4.11
* [XMLDocument](https://dom.spec.whatwg.org/#xmldocument), in §4.5
* [XML document](https://dom.spec.whatwg.org/#xml-document), in §4.5
* [xmlEncoding](https://dom.spec.whatwg.org/#dom-document-xmlencoding), in §8.2
* [XML namespace](https://dom.spec.whatwg.org/#xml-namespace), in §2.5
* [XMLNS namespace](https://dom.spec.whatwg.org/#xmlns-namespace), in §2.5
* [xmlStandalone](https://dom.spec.whatwg.org/#dom-document-xmlstandalone), in §8.2
* [xmlVersion](https://dom.spec.whatwg.org/#dom-document-xmlversion), in §8.2

**Terms defined by reference**

* [ECMASCRIPT] defines the following terms:
  + [Construct](https://tc39.github.io/ecma262/#sec-construct)
* [css-animations-1] defines the following terms:
  + [AnimationEvent](https://drafts.csswg.org/css-animations-1/#animationevent)
* [css-transitions-1] defines the following terms:
  + [TransitionEvent](https://drafts.csswg.org/css-transitions-1/#Events-TransitionEvent)
* [CSSOM-VIEW] defines the following terms:
  + [getBoundingClientRect()](https://drafts.csswg.org/cssom-view-1/#dom-range-getboundingclientrect)
  + [getClientRects()](https://drafts.csswg.org/cssom-view-1/#dom-range-getclientrects)
* [HTML] defines the following terms:
  + [:defined](https://html.spec.whatwg.org/multipage/scripting.html#selector-defined)
  + [BeforeUnloadEvent](https://html.spec.whatwg.org/multipage/browsers.html#beforeunloadevent)
  + [CloseEvent](https://html.spec.whatwg.org/multipage/comms.html#closeevent)
  + [DragEvent](https://html.spec.whatwg.org/multipage/interaction.html#dragevent)
  + [ErrorEvent](https://html.spec.whatwg.org/multipage/webappapis.html#errorevent)
  + [HTMLElement](https://html.spec.whatwg.org/multipage/dom.html#htmlelement)
  + [HTMLHtmlElement](https://html.spec.whatwg.org/multipage/semantics.html#htmlhtmlelement)
  + [HTMLSlotElement](https://html.spec.whatwg.org/multipage/scripting.html#htmlslotelement)
  + [HashChangeEvent](https://html.spec.whatwg.org/multipage/browsers.html#hashchangeevent)
  + [MessageEvent](https://html.spec.whatwg.org/multipage/comms.html#messageevent)
  + [PageTransitionEvent](https://html.spec.whatwg.org/multipage/browsers.html#pagetransitionevent)
  + [PopStateEvent](https://html.spec.whatwg.org/multipage/browsers.html#popstateevent)
  + [StorageEvent](https://html.spec.whatwg.org/multipage/webstorage.html#storageevent)
  + [TrackEvent](https://html.spec.whatwg.org/multipage/embedded-content.html#trackevent)
  + [Window](https://html.spec.whatwg.org/multipage/browsers.html#window)
  + [body](https://html.spec.whatwg.org/multipage/semantics.html#the-body-element)
  + [constructor](https://html.spec.whatwg.org/multipage/scripting.html#concept-custom-element-definition-constructor)
  + [customized built-in element](https://html.spec.whatwg.org/multipage/scripting.html#customized-built-in-element)
  + [enqueue a custom element callback reaction](https://html.spec.whatwg.org/multipage/scripting.html#enqueue-a-custom-element-callback-reaction)
  + [enqueue a custom element upgrade reaction](https://html.spec.whatwg.org/multipage/scripting.html#enqueue-a-custom-element-upgrade-reaction)
  + [head](https://html.spec.whatwg.org/multipage/semantics.html#the-head-element)
  + [html](https://html.spec.whatwg.org/multipage/semantics.html#the-html-element)
  + [input](https://html.spec.whatwg.org/multipage/forms.html#the-input-element)
  + [load(url)](https://html.spec.whatwg.org/multipage/dom.html#dom-xmldocument-load)
  + [local name](https://html.spec.whatwg.org/multipage/scripting.html#concept-custom-element-definition-local-name)
  + [look up a custom element definition](https://html.spec.whatwg.org/multipage/scripting.html#look-up-a-custom-element-definition)
  + [name](https://html.spec.whatwg.org/multipage/scripting.html#concept-custom-element-definition-name)
  + [opaque origin](https://html.spec.whatwg.org/multipage/browsers.html#concept-origin-opaque)
  + [script](https://html.spec.whatwg.org/multipage/scripting.html#script)
  + [slot](https://html.spec.whatwg.org/multipage/scripting.html#the-slot-element)
  + [template](https://html.spec.whatwg.org/multipage/scripting.html#the-template-element)
  + [title](https://html.spec.whatwg.org/multipage/semantics.html#the-title-element)
  + [try to upgrade an element](https://html.spec.whatwg.org/multipage/scripting.html#concept-try-upgrade)
  + [unicode serialisation of an origin](https://html.spec.whatwg.org/multipage/browsers.html#unicode-serialisation-of-an-origin)
  + [upgrade an element](https://html.spec.whatwg.org/multipage/scripting.html#concept-upgrade-an-element)
  + [valid custom element name](https://html.spec.whatwg.org/multipage/scripting.html#valid-custom-element-name)
* [selectors-4] defines the following terms:
  + [:scope element](https://drafts.csswg.org/selectors-4/#scope-element)
  + [evaluate a selector](https://drafts.csswg.org/selectors-4/#evaluate-a-selector)
  + [match a selector against an element](https://drafts.csswg.org/selectors-4/#match-a-selector-against-an-element)
  + [parse a relative selector](https://drafts.csswg.org/selectors-4/#parse-a-relative-selector)
  + [parse a selector](https://drafts.csswg.org/selectors-4/#parse-a-selector)
  + [scope-filtered](https://drafts.csswg.org/selectors-4/#scope-filtered)
  + [scoping root](https://drafts.csswg.org/selectors-4/#scoping-root)
* [SVG] defines the following terms:
  + [SVGZoomEvent](https://www.w3.org/TR/SVG/script.html#InterfaceSVGZoomEvent)
* [URL] defines the following terms:
  + [url](https://url.spec.whatwg.org/#concept-url)
  + [url serializer](https://url.spec.whatwg.org/#concept-url-serializer)
* [WEBIDL] defines the following terms:
  + [DOMException](https://heycam.github.io/webidl/#dfn-DOMException)
  + [DOMString](https://heycam.github.io/webidl/#idl-DOMString)
  + [HierarchyRequestError](https://heycam.github.io/webidl/#hierarchyrequesterror)
  + [InUseAttributeError](https://heycam.github.io/webidl/#inuseattributeerror)
  + [IndexSizeError](https://heycam.github.io/webidl/#indexsizeerror)
  + [InvalidCharacterError](https://heycam.github.io/webidl/#invalidcharactererror)
  + [InvalidNodeTypeError](https://heycam.github.io/webidl/#invalidnodetypeerror)
  + [InvalidStateError](https://heycam.github.io/webidl/#invalidstateerror)
  + [NamespaceError](https://heycam.github.io/webidl/#namespaceerror)
  + [NotFoundError](https://heycam.github.io/webidl/#notfounderror)
  + [NotSupportedError](https://heycam.github.io/webidl/#notsupportederror)
  + [SyntaxError](https://heycam.github.io/webidl/#syntaxerror)
  + [WrongDocumentError](https://heycam.github.io/webidl/#wrongdocumenterror)

**References**

**Normative References**

[CSS-ANIMATIONS-1]

CSS Animations Module Level 1 URL: <https://drafts.csswg.org/css-animations-1/>

[CSS-TRANSITIONS-1]

CSS Transitions Module Level 1 URL: <https://drafts.csswg.org/css-transitions-1/>

[CSS3-ANIMATIONS]

Dean Jackson; et al. [CSS Animations](http://www.w3.org/TR/css3-animations/). 19 February 2013. WD. URL: <http://www.w3.org/TR/css3-animations/>

[CSS3-TRANSITIONS]

Dean Jackson; et al. [CSS Transitions](http://dev.w3.org/csswg/css-transitions/). 19 November 2013. WD. URL: <http://dev.w3.org/csswg/css-transitions/>

[DEVICE-ORIENTATION]

Stephen Block; Andrei Popescu. [DeviceOrientation Event Specification](http://www.w3.org/TR/orientation-event/). 1 December 2011. LCWD. URL: <http://www.w3.org/TR/orientation-event/>

[ECMASCRIPT]

[ECMAScript Language Specification](https://tc39.github.io/ecma262/). URL: <https://tc39.github.io/ecma262/>

[ENCODING]

Anne van Kesteren. [Encoding Standard](https://encoding.spec.whatwg.org/). Living Standard. URL: <https://encoding.spec.whatwg.org/>

[HTML]

Ian Hickson. [HTML Standard](https://html.spec.whatwg.org/multipage/). Living Standard. URL: <https://html.spec.whatwg.org/multipage/>

[RFC2119]

S. Bradner. [Key words for use in RFCs to Indicate Requirement Levels](https://tools.ietf.org/html/rfc2119). March 1997. Best Current Practice. URL: <https://tools.ietf.org/html/rfc2119>

[SELECTORS-4]

Selectors Level 4 URL: <https://drafts.csswg.org/selectors-4/>

[SELECTORS4]

Elika Etemad; Tab Atkins Jr.. [Selectors Level 4](http://www.w3.org/TR/selectors4/). 2 May 2013. WD. URL: <http://www.w3.org/TR/selectors4/>

[SERVICE-WORKERS]

Alex Russell; Jungkee Song; Jake Archibald. [Service Workers](https://slightlyoff.github.io/ServiceWorker/spec/service_worker/). 25 June 2015. WD. URL: <https://slightlyoff.github.io/ServiceWorker/spec/service_worker/>

[SVG]

Jon Ferraiolo. [Scalable Vector Graphics (SVG) 1.0 Specification](http://www.w3.org/TR/SVG/). 4 September 2001. REC. URL: <http://www.w3.org/TR/SVG/>

[TOUCH-EVENTS]

Doug Schepers; et al. [Touch Events](http://dvcs.w3.org/hg/webevents/raw-file/v1/touchevents.html). 10 October 2013. REC. URL: <http://dvcs.w3.org/hg/webevents/raw-file/v1/touchevents.html>

[UIEVENTS]

Gary Kacmarcik; Travis Leithead. [UI Events Specification](https://w3c.github.io/uievents/). 15 December 2015. WD. URL: <https://w3c.github.io/uievents/>

[URL]

Anne van Kesteren; Sam Ruby. [URL Standard](https://url.spec.whatwg.org/). Living Standard. URL: <https://url.spec.whatwg.org/>

[WEBGL]

Chris Marrin (Apple Inc.). [WebGL Specification, Version 1.0](https://www.khronos.org/registry/webgl/specs/1.0/). 10 February 2011. URL: <https://www.khronos.org/registry/webgl/specs/1.0/>

[WEBIDL]

Cameron McCormack; Boris Zbarsky. [WebIDL Level 1](https://heycam.github.io/webidl/). 8 March 2016. CR. URL: <https://heycam.github.io/webidl/>

[XHR]

Anne van Kesteren. [XMLHttpRequest Standard](https://xhr.spec.whatwg.org/). Living Standard. URL: <https://xhr.spec.whatwg.org/>

[XML]

Tim Bray; et al. [Extensible Markup Language (XML) 1.0 (Fifth Edition)](http://www.w3.org/TR/xml). 26 November 2008. REC. URL: <http://www.w3.org/TR/xml>

[XML-NAMES]

Tim Bray; et al. [Namespaces in XML 1.0 (Third Edition)](http://www.w3.org/TR/xml-names). 8 December 2009. REC. URL: <http://www.w3.org/TR/xml-names>

**Informative References**

[CSSOM-VIEW]

Simon Pieters. [CSSOM View Module](https://drafts.csswg.org/cssom-view/). 17 March 2016. WD. URL: <https://drafts.csswg.org/cssom-view/>

[DOM-Level-2-Traversal-Range]

Joseph Kesselman; et al. [Document Object Model (DOM) Level 2 Traversal and Range Specification](http://www.w3.org/TR/DOM-Level-2-Traversal-Range/). 13 November 2000. REC. URL: <http://www.w3.org/TR/DOM-Level-2-Traversal-Range/>

[DOM-Level-3-Core]

Arnaud Le Hors; et al. [Document Object Model (DOM) Level 3 Core Specification](http://www.w3.org/TR/DOM-Level-3-Core/). 7 April 2004. REC. URL: <http://www.w3.org/TR/DOM-Level-3-Core/>

[DOM-Parsing]

Travis Leithead. [DOM Parsing and Serialization](https://dvcs.w3.org/hg/innerhtml/raw-file/tip/index.html). 17 June 2014. CR. URL: <https://dvcs.w3.org/hg/innerhtml/raw-file/tip/index.html>

[ELEMENTTRAVERSAL]

Doug Schepers; Robin Berjon. [Element Traversal Specification](http://www.w3.org/TR/ElementTraversal/). 22 December 2008. REC. URL: <http://www.w3.org/TR/ElementTraversal/>

[INDEXEDDB]

Nikunj Mehta; et al. [Indexed Database API](http://dvcs.w3.org/hg/IndexedDB/raw-file/tip/Overview.html). 8 January 2015. REC. URL: <http://dvcs.w3.org/hg/IndexedDB/raw-file/tip/Overview.html>

[SELECTORS-API2]

Lachlan Hunt. [Selectors API Level 2](http://dev.w3.org/2006/webapi/selectors-api2/). 17 October 2013. NOTE. URL: <http://dev.w3.org/2006/webapi/selectors-api2/>

[UIEVENTS-20031107]

Philippe Le Hégaret; Tim Pixley. [Document Object Model (DOM) Level 3 Events Specification](http://www.w3.org/TR/2003/NOTE-DOM-Level-3-Events-20031107). 7 November 2003. NOTE. URL: <http://www.w3.org/TR/2003/NOTE-DOM-Level-3-Events-20031107>

**IDL Index**

[[Constructor](https://dom.spec.whatwg.org/#dom-event-event)(DOMString [type](https://dom.spec.whatwg.org/#dom-event-event-type-eventinitdict-type), optional [EventInit](https://dom.spec.whatwg.org/#dictdef-eventinit) [eventInitDict](https://dom.spec.whatwg.org/#dom-event-event-type-eventinitdict-eventinitdict)),

Exposed=(Window,Worker)]

interface [Event](https://dom.spec.whatwg.org/#event) {

readonly attribute DOMString [type](https://dom.spec.whatwg.org/#dom-event-type);

readonly attribute [EventTarget](https://dom.spec.whatwg.org/#eventtarget)? [target](https://dom.spec.whatwg.org/#dom-event-target);

readonly attribute [EventTarget](https://dom.spec.whatwg.org/#eventtarget)? [currentTarget](https://dom.spec.whatwg.org/#dom-event-currenttarget);

const unsigned short [NONE](https://dom.spec.whatwg.org/#dom-event-none) = 0;

const unsigned short [CAPTURING\_PHASE](https://dom.spec.whatwg.org/#dom-event-capturing_phase) = 1;

const unsigned short [AT\_TARGET](https://dom.spec.whatwg.org/#dom-event-at_target) = 2;

const unsigned short [BUBBLING\_PHASE](https://dom.spec.whatwg.org/#dom-event-bubbling_phase) = 3;

readonly attribute unsigned short [eventPhase](https://dom.spec.whatwg.org/#dom-event-eventphase);

void [stopPropagation](https://dom.spec.whatwg.org/#dom-event-stoppropagation)();

void [stopImmediatePropagation](https://dom.spec.whatwg.org/#dom-event-stopimmediatepropagation)();

readonly attribute boolean [bubbles](https://dom.spec.whatwg.org/#dom-event-bubbles);

readonly attribute boolean [cancelable](https://dom.spec.whatwg.org/#dom-event-cancelable);

void [preventDefault](https://dom.spec.whatwg.org/#dom-event-preventdefault)();

readonly attribute boolean [defaultPrevented](https://dom.spec.whatwg.org/#dom-event-defaultprevented);

[Unforgeable] readonly attribute boolean [isTrusted](https://dom.spec.whatwg.org/#dom-event-istrusted);

readonly attribute [DOMTimeStamp](https://heycam.github.io/webidl/#common-domtimestamp) [timeStamp](https://dom.spec.whatwg.org/#dom-event-timestamp);

void [initEvent](https://dom.spec.whatwg.org/#dom-event-initevent)(DOMString [type](https://dom.spec.whatwg.org/#dom-event-initevent-type-bubbles-cancelable-type), boolean [bubbles](https://dom.spec.whatwg.org/#dom-event-initevent-type-bubbles-cancelable-bubbles), boolean [cancelable](https://dom.spec.whatwg.org/#dom-event-initevent-type-bubbles-cancelable-cancelable));

};

dictionary [EventInit](https://dom.spec.whatwg.org/#dictdef-eventinit) {

boolean [bubbles](https://dom.spec.whatwg.org/#dom-eventinit-bubbles) = false;

boolean [cancelable](https://dom.spec.whatwg.org/#dom-eventinit-cancelable) = false;

};

[[Constructor](https://dom.spec.whatwg.org/#dom-customevent-customevent)(DOMString [type](https://dom.spec.whatwg.org/#dom-customevent-customevent-type-eventinitdict-type), optional [CustomEventInit](https://dom.spec.whatwg.org/#dictdef-customeventinit) [eventInitDict](https://dom.spec.whatwg.org/#dom-customevent-customevent-type-eventinitdict-eventinitdict)),

Exposed=(Window,Worker)]

interface [CustomEvent](https://dom.spec.whatwg.org/#customevent) : [Event](https://dom.spec.whatwg.org/#event) {

readonly attribute any [detail](https://dom.spec.whatwg.org/#dom-customevent-detail);

void [initCustomEvent](https://dom.spec.whatwg.org/#dom-customevent-initcustomevent)(DOMString [type](https://dom.spec.whatwg.org/#dom-customevent-initcustomevent-type-bubbles-cancelable-detail-type), boolean [bubbles](https://dom.spec.whatwg.org/#dom-customevent-initcustomevent-type-bubbles-cancelable-detail-bubbles), boolean [cancelable](https://dom.spec.whatwg.org/#dom-customevent-initcustomevent-type-bubbles-cancelable-detail-cancelable), any [detail](https://dom.spec.whatwg.org/#dom-customevent-initcustomevent-type-bubbles-cancelable-detail-detail));

};

dictionary [CustomEventInit](https://dom.spec.whatwg.org/#dictdef-customeventinit) : [EventInit](https://dom.spec.whatwg.org/#dictdef-eventinit) {

any [detail](https://dom.spec.whatwg.org/#dom-customeventinit-detail) = null;

};

[Exposed=(Window,Worker)]

interface [EventTarget](https://dom.spec.whatwg.org/#eventtarget) {

void [addEventListener](https://dom.spec.whatwg.org/#dom-eventtarget-addeventlistener)(DOMString [type](https://dom.spec.whatwg.org/#dom-eventtarget-addeventlistener-type-callback-options-type), [EventListener](https://dom.spec.whatwg.org/#callbackdef-eventlistener)? callback, optional ([AddEventListenerOptions](https://dom.spec.whatwg.org/#dictdef-addeventlisteneroptions) or boolean) [options](https://dom.spec.whatwg.org/#dom-eventtarget-addeventlistener-type-callback-options-options));

void [removeEventListener](https://dom.spec.whatwg.org/#dom-eventtarget-removeeventlistener)(DOMString [type](https://dom.spec.whatwg.org/#dom-eventtarget-removeeventlistener-type-callback-options-type), [EventListener](https://dom.spec.whatwg.org/#callbackdef-eventlistener)? callback, optional ([EventListenerOptions](https://dom.spec.whatwg.org/#dictdef-eventlisteneroptions) or boolean) [options](https://dom.spec.whatwg.org/#dom-eventtarget-removeeventlistener-type-callback-options-options));

boolean [dispatchEvent](https://dom.spec.whatwg.org/#dom-eventtarget-dispatchevent)([Event](https://dom.spec.whatwg.org/#event) [event](https://dom.spec.whatwg.org/#dom-eventtarget-dispatchevent-event-event));

};

callback interface [EventListener](https://dom.spec.whatwg.org/#callbackdef-eventlistener) {

void [handleEvent](https://dom.spec.whatwg.org/#dom-eventlistener-handleevent)([Event](https://dom.spec.whatwg.org/#event) [event](https://dom.spec.whatwg.org/#dom-eventlistener-handleevent-event-event));

};

dictionary [EventListenerOptions](https://dom.spec.whatwg.org/#dictdef-eventlisteneroptions) {

boolean [capture](https://dom.spec.whatwg.org/#dom-eventlisteneroptions-capture) = false;

};

dictionary [AddEventListenerOptions](https://dom.spec.whatwg.org/#dictdef-addeventlisteneroptions) : [EventListenerOptions](https://dom.spec.whatwg.org/#dictdef-eventlisteneroptions) {

boolean [passive](https://dom.spec.whatwg.org/#dom-addeventlisteneroptions-passive) = false;

boolean [once](https://dom.spec.whatwg.org/#dom-addeventlisteneroptions-once) = false;

};

[NoInterfaceObject,

Exposed=Window]

interface [NonElementParentNode](https://dom.spec.whatwg.org/#nonelementparentnode) {

[Element](https://dom.spec.whatwg.org/#element)? [getElementById](https://dom.spec.whatwg.org/#dom-nonelementparentnode-getelementbyid)(DOMString [elementId](https://dom.spec.whatwg.org/#dom-nonelementparentnode-getelementbyid-elementid-elementid));

};

[Document](https://dom.spec.whatwg.org/#document) implements [NonElementParentNode](https://dom.spec.whatwg.org/#nonelementparentnode);

[DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) implements [NonElementParentNode](https://dom.spec.whatwg.org/#nonelementparentnode);

[NoInterfaceObject,

Exposed=Window]

interface [DocumentOrShadowRoot](https://dom.spec.whatwg.org/#documentorshadowroot) {

};

[Document](https://dom.spec.whatwg.org/#document) implements [DocumentOrShadowRoot](https://dom.spec.whatwg.org/#documentorshadowroot);

[ShadowRoot](https://dom.spec.whatwg.org/#shadowroot) implements [DocumentOrShadowRoot](https://dom.spec.whatwg.org/#documentorshadowroot);

[NoInterfaceObject,

Exposed=Window]

interface [ParentNode](https://dom.spec.whatwg.org/#parentnode) {

[SameObject] readonly attribute [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) [children](https://dom.spec.whatwg.org/#dom-parentnode-children);

readonly attribute [Element](https://dom.spec.whatwg.org/#element)? [firstElementChild](https://dom.spec.whatwg.org/#dom-parentnode-firstelementchild);

readonly attribute [Element](https://dom.spec.whatwg.org/#element)? [lastElementChild](https://dom.spec.whatwg.org/#dom-parentnode-lastelementchild);

readonly attribute unsigned long [childElementCount](https://dom.spec.whatwg.org/#dom-parentnode-childelementcount);

[CEReactions, Unscopable] void [prepend](https://dom.spec.whatwg.org/#dom-parentnode-prepend)(([Node](https://dom.spec.whatwg.org/#node) or DOMString)... [nodes](https://dom.spec.whatwg.org/#dom-parentnode-prepend-nodes-nodes));

[CEReactions, Unscopable] void [append](https://dom.spec.whatwg.org/#dom-parentnode-append)(([Node](https://dom.spec.whatwg.org/#node) or DOMString)... [nodes](https://dom.spec.whatwg.org/#dom-parentnode-append-nodes-nodes));

[Element](https://dom.spec.whatwg.org/#element)? [querySelector](https://dom.spec.whatwg.org/#dom-parentnode-queryselector)(DOMString [selectors](https://dom.spec.whatwg.org/#dom-parentnode-queryselector-selectors-selectors));

[NewObject] [NodeList](https://dom.spec.whatwg.org/#nodelist) [querySelectorAll](https://dom.spec.whatwg.org/#dom-parentnode-queryselectorall)(DOMString [selectors](https://dom.spec.whatwg.org/#dom-parentnode-queryselectorall-selectors-selectors));

};

[Document](https://dom.spec.whatwg.org/#document) implements [ParentNode](https://dom.spec.whatwg.org/#parentnode);

[DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) implements [ParentNode](https://dom.spec.whatwg.org/#parentnode);

[Element](https://dom.spec.whatwg.org/#element) implements [ParentNode](https://dom.spec.whatwg.org/#parentnode);

[NoInterfaceObject,

Exposed=Window]

interface [NonDocumentTypeChildNode](https://dom.spec.whatwg.org/#nondocumenttypechildnode) {

readonly attribute [Element](https://dom.spec.whatwg.org/#element)? [previousElementSibling](https://dom.spec.whatwg.org/#dom-nondocumenttypechildnode-previouselementsibling);

readonly attribute [Element](https://dom.spec.whatwg.org/#element)? [nextElementSibling](https://dom.spec.whatwg.org/#dom-nondocumenttypechildnode-nextelementsibling);

};

[Element](https://dom.spec.whatwg.org/#element) implements [NonDocumentTypeChildNode](https://dom.spec.whatwg.org/#nondocumenttypechildnode);

[CharacterData](https://dom.spec.whatwg.org/#characterdata) implements [NonDocumentTypeChildNode](https://dom.spec.whatwg.org/#nondocumenttypechildnode);

[NoInterfaceObject,

Exposed=Window]

interface [ChildNode](https://dom.spec.whatwg.org/#childnode) {

[CEReactions, Unscopable] void [before](https://dom.spec.whatwg.org/#dom-childnode-before)(([Node](https://dom.spec.whatwg.org/#node) or DOMString)... [nodes](https://dom.spec.whatwg.org/#dom-childnode-before-nodes-nodes));

[CEReactions, Unscopable] void [after](https://dom.spec.whatwg.org/#dom-childnode-after)(([Node](https://dom.spec.whatwg.org/#node) or DOMString)... [nodes](https://dom.spec.whatwg.org/#dom-childnode-after-nodes-nodes));

[CEReactions, Unscopable] void [replaceWith](https://dom.spec.whatwg.org/#dom-childnode-replacewith)(([Node](https://dom.spec.whatwg.org/#node) or DOMString)... [nodes](https://dom.spec.whatwg.org/#dom-childnode-replacewith-nodes-nodes));

[CEReactions, Unscopable] void [remove](https://dom.spec.whatwg.org/#dom-childnode-remove)();

};

[DocumentType](https://dom.spec.whatwg.org/#documenttype) implements [ChildNode](https://dom.spec.whatwg.org/#childnode);

[Element](https://dom.spec.whatwg.org/#element) implements [ChildNode](https://dom.spec.whatwg.org/#childnode);

[CharacterData](https://dom.spec.whatwg.org/#characterdata) implements [ChildNode](https://dom.spec.whatwg.org/#childnode);

[NoInterfaceObject,

Exposed=Window]

interface [Slotable](https://dom.spec.whatwg.org/#slotable) {

readonly attribute [HTMLSlotElement](https://html.spec.whatwg.org/multipage/scripting.html#htmlslotelement)? [assignedSlot](https://dom.spec.whatwg.org/#dom-slotable-assignedslot);

};

[Element](https://dom.spec.whatwg.org/#element) implements [Slotable](https://dom.spec.whatwg.org/#slotable);

[Text](https://dom.spec.whatwg.org/#text) implements [Slotable](https://dom.spec.whatwg.org/#slotable);

[Exposed=Window]

interface [NodeList](https://dom.spec.whatwg.org/#nodelist) {

getter [Node](https://dom.spec.whatwg.org/#node)? [item](https://dom.spec.whatwg.org/#dom-nodelist-item)(unsigned long [index](https://dom.spec.whatwg.org/#dom-nodelist-item-index-index));

readonly attribute unsigned long [length](https://dom.spec.whatwg.org/#dom-nodelist-length);

iterable<[Node](https://dom.spec.whatwg.org/#node)>;

};

[Exposed=Window, LegacyUnenumerableNamedProperties]

interface [HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) {

readonly attribute unsigned long [length](https://dom.spec.whatwg.org/#dom-htmlcollection-length);

getter [Element](https://dom.spec.whatwg.org/#element)? [item](https://dom.spec.whatwg.org/#dom-htmlcollection-item)(unsigned long [index](https://dom.spec.whatwg.org/#dom-htmlcollection-item-index-index));

getter [Element](https://dom.spec.whatwg.org/#element)? [namedItem](https://dom.spec.whatwg.org/#dom-htmlcollection-nameditem)(DOMString [name](https://dom.spec.whatwg.org/#dom-htmlcollection-nameditem-name-name));

};

[[Constructor](https://dom.spec.whatwg.org/#dom-mutationobserver-mutationobserver)([MutationCallback](https://dom.spec.whatwg.org/#callbackdef-mutationcallback) callback)]

interface [MutationObserver](https://dom.spec.whatwg.org/#mutationobserver) {

void [observe](https://dom.spec.whatwg.org/#dom-mutationobserver-observe)([Node](https://dom.spec.whatwg.org/#node) [target](https://dom.spec.whatwg.org/#dom-mutationobserver-observe-target-options-target), [MutationObserverInit](https://dom.spec.whatwg.org/#dictdef-mutationobserverinit) [options](https://dom.spec.whatwg.org/#dom-mutationobserver-observe-target-options-options));

void [disconnect](https://dom.spec.whatwg.org/#dom-mutationobserver-disconnect)();

sequence<[MutationRecord](https://dom.spec.whatwg.org/#mutationrecord)> [takeRecords](https://dom.spec.whatwg.org/#dom-mutationobserver-takerecords)();

};

callback [MutationCallback](https://dom.spec.whatwg.org/#callbackdef-mutationcallback) = void (sequence<[MutationRecord](https://dom.spec.whatwg.org/#mutationrecord)> [mutations](https://dom.spec.whatwg.org/#dom-mutationcallback-mutations), [MutationObserver](https://dom.spec.whatwg.org/#mutationobserver) [observer](https://dom.spec.whatwg.org/#dom-mutationcallback-observer));

dictionary [MutationObserverInit](https://dom.spec.whatwg.org/#dictdef-mutationobserverinit) {

boolean [childList](https://dom.spec.whatwg.org/#dom-mutationobserverinit-childlist) = false;

boolean [attributes](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributes);

boolean [characterData](https://dom.spec.whatwg.org/#dom-mutationobserverinit-characterdata);

boolean [subtree](https://dom.spec.whatwg.org/#dom-mutationobserverinit-subtree) = false;

boolean [attributeOldValue](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributeoldvalue);

boolean [characterDataOldValue](https://dom.spec.whatwg.org/#dom-mutationobserverinit-characterdataoldvalue);

sequence<DOMString> [attributeFilter](https://dom.spec.whatwg.org/#dom-mutationobserverinit-attributefilter);

};

[Exposed=Window]

interface [MutationRecord](https://dom.spec.whatwg.org/#mutationrecord) {

readonly attribute DOMString [type](https://dom.spec.whatwg.org/#dom-mutationrecord-type);

[SameObject] readonly attribute [Node](https://dom.spec.whatwg.org/#node) [target](https://dom.spec.whatwg.org/#dom-mutationrecord-target);

[SameObject] readonly attribute [NodeList](https://dom.spec.whatwg.org/#nodelist) [addedNodes](https://dom.spec.whatwg.org/#dom-mutationrecord-addednodes);

[SameObject] readonly attribute [NodeList](https://dom.spec.whatwg.org/#nodelist) [removedNodes](https://dom.spec.whatwg.org/#dom-mutationrecord-removednodes);

readonly attribute [Node](https://dom.spec.whatwg.org/#node)? [previousSibling](https://dom.spec.whatwg.org/#dom-mutationrecord-previoussibling);

readonly attribute [Node](https://dom.spec.whatwg.org/#node)? [nextSibling](https://dom.spec.whatwg.org/#dom-mutationrecord-nextsibling);

readonly attribute DOMString? [attributeName](https://dom.spec.whatwg.org/#dom-mutationrecord-attributename);

readonly attribute DOMString? [attributeNamespace](https://dom.spec.whatwg.org/#dom-mutationrecord-attributenamespace);

readonly attribute DOMString? [oldValue](https://dom.spec.whatwg.org/#dom-mutationrecord-oldvalue);

};

[Exposed=Window]

interface [Node](https://dom.spec.whatwg.org/#node) : [EventTarget](https://dom.spec.whatwg.org/#eventtarget) {

const unsigned short [ELEMENT\_NODE](https://dom.spec.whatwg.org/#dom-node-element_node) = 1;

const unsigned short [ATTRIBUTE\_NODE](https://dom.spec.whatwg.org/#dom-node-attribute_node) = 2; // historical

const unsigned short [TEXT\_NODE](https://dom.spec.whatwg.org/#dom-node-text_node) = 3;

const unsigned short [CDATA\_SECTION\_NODE](https://dom.spec.whatwg.org/#dom-node-cdata_section_node) = 4; // historical

const unsigned short [ENTITY\_REFERENCE\_NODE](https://dom.spec.whatwg.org/#dom-node-entity_reference_node) = 5; // historical

const unsigned short [ENTITY\_NODE](https://dom.spec.whatwg.org/#dom-node-entity_node) = 6; // historical

const unsigned short [PROCESSING\_INSTRUCTION\_NODE](https://dom.spec.whatwg.org/#dom-node-processing_instruction_node) = 7;

const unsigned short [COMMENT\_NODE](https://dom.spec.whatwg.org/#dom-node-comment_node) = 8;

const unsigned short [DOCUMENT\_NODE](https://dom.spec.whatwg.org/#dom-node-document_node) = 9;

const unsigned short [DOCUMENT\_TYPE\_NODE](https://dom.spec.whatwg.org/#dom-node-document_type_node) = 10;

const unsigned short [DOCUMENT\_FRAGMENT\_NODE](https://dom.spec.whatwg.org/#dom-node-document_fragment_node) = 11;

const unsigned short [NOTATION\_NODE](https://dom.spec.whatwg.org/#dom-node-notation_node) = 12; // historical

readonly attribute unsigned short [nodeType](https://dom.spec.whatwg.org/#dom-node-nodetype);

readonly attribute DOMString [nodeName](https://dom.spec.whatwg.org/#dom-node-nodename);

readonly attribute DOMString [baseURI](https://dom.spec.whatwg.org/#dom-node-baseuri);

readonly attribute boolean [isConnected](https://dom.spec.whatwg.org/#dom-node-isconnected);

readonly attribute [Document](https://dom.spec.whatwg.org/#document)? [ownerDocument](https://dom.spec.whatwg.org/#dom-node-ownerdocument);

readonly attribute [Node](https://dom.spec.whatwg.org/#node) [rootNode](https://dom.spec.whatwg.org/#dom-node-rootnode);

readonly attribute [Node](https://dom.spec.whatwg.org/#node)? [parentNode](https://dom.spec.whatwg.org/#dom-node-parentnode);

readonly attribute [Element](https://dom.spec.whatwg.org/#element)? [parentElement](https://dom.spec.whatwg.org/#dom-node-parentelement);

boolean [hasChildNodes](https://dom.spec.whatwg.org/#dom-node-haschildnodes)();

[SameObject] readonly attribute [NodeList](https://dom.spec.whatwg.org/#nodelist) [childNodes](https://dom.spec.whatwg.org/#dom-node-childnodes);

readonly attribute [Node](https://dom.spec.whatwg.org/#node)? [firstChild](https://dom.spec.whatwg.org/#dom-node-firstchild);

readonly attribute [Node](https://dom.spec.whatwg.org/#node)? [lastChild](https://dom.spec.whatwg.org/#dom-node-lastchild);

readonly attribute [Node](https://dom.spec.whatwg.org/#node)? [previousSibling](https://dom.spec.whatwg.org/#dom-node-previoussibling);

readonly attribute [Node](https://dom.spec.whatwg.org/#node)? [nextSibling](https://dom.spec.whatwg.org/#dom-node-nextsibling);

[CEReactions] attribute DOMString? [nodeValue](https://dom.spec.whatwg.org/#dom-node-nodevalue);

[CEReactions] attribute DOMString? [textContent](https://dom.spec.whatwg.org/#dom-node-textcontent);

[CEReactions] void [normalize](https://dom.spec.whatwg.org/#dom-node-normalize)();

[CEReactions, NewObject] [Node](https://dom.spec.whatwg.org/#node) [cloneNode](https://dom.spec.whatwg.org/#dom-node-clonenode)(optional boolean [deep](https://dom.spec.whatwg.org/#dom-node-clonenode-deep-deep) = false);

boolean [isEqualNode](https://dom.spec.whatwg.org/#dom-node-isequalnode)([Node](https://dom.spec.whatwg.org/#node)? [otherNode](https://dom.spec.whatwg.org/#dom-node-isequalnode-othernode-othernode));

boolean [isSameNode](https://dom.spec.whatwg.org/#dom-node-issamenode)([Node](https://dom.spec.whatwg.org/#node)? [otherNode](https://dom.spec.whatwg.org/#dom-node-issamenode-othernode-othernode)); // historical alias of ===

const unsigned short [DOCUMENT\_POSITION\_DISCONNECTED](https://dom.spec.whatwg.org/#dom-node-document_position_disconnected) = 0x01;

const unsigned short [DOCUMENT\_POSITION\_PRECEDING](https://dom.spec.whatwg.org/#dom-node-document_position_preceding) = 0x02;

const unsigned short [DOCUMENT\_POSITION\_FOLLOWING](https://dom.spec.whatwg.org/#dom-node-document_position_following) = 0x04;

const unsigned short [DOCUMENT\_POSITION\_CONTAINS](https://dom.spec.whatwg.org/#dom-node-document_position_contains) = 0x08;

const unsigned short [DOCUMENT\_POSITION\_CONTAINED\_BY](https://dom.spec.whatwg.org/#dom-node-document_position_contained_by) = 0x10;

const unsigned short [DOCUMENT\_POSITION\_IMPLEMENTATION\_SPECIFIC](https://dom.spec.whatwg.org/#dom-node-document_position_implementation_specific) = 0x20;

unsigned short [compareDocumentPosition](https://dom.spec.whatwg.org/#dom-node-comparedocumentposition)([Node](https://dom.spec.whatwg.org/#node) [other](https://dom.spec.whatwg.org/#dom-node-comparedocumentposition-other-other));

boolean [contains](https://dom.spec.whatwg.org/#dom-node-contains)([Node](https://dom.spec.whatwg.org/#node)? [other](https://dom.spec.whatwg.org/#dom-node-contains-other-other));

DOMString? [lookupPrefix](https://dom.spec.whatwg.org/#dom-node-lookupprefix)(DOMString? [namespace](https://dom.spec.whatwg.org/#dom-node-lookupprefix-namespace-namespace));

DOMString? [lookupNamespaceURI](https://dom.spec.whatwg.org/#dom-node-lookupnamespaceuri)(DOMString? [prefix](https://dom.spec.whatwg.org/#dom-node-lookupnamespaceuri-prefix-prefix));

boolean [isDefaultNamespace](https://dom.spec.whatwg.org/#dom-node-isdefaultnamespace)(DOMString? [namespace](https://dom.spec.whatwg.org/#dom-node-isdefaultnamespace-namespace-namespace));

[CEReactions] [Node](https://dom.spec.whatwg.org/#node) [insertBefore](https://dom.spec.whatwg.org/#dom-node-insertbefore)([Node](https://dom.spec.whatwg.org/#node) [node](https://dom.spec.whatwg.org/#dom-node-insertbefore-node-child-node), [Node](https://dom.spec.whatwg.org/#node)? [child](https://dom.spec.whatwg.org/#dom-node-insertbefore-node-child-child));

[CEReactions] [Node](https://dom.spec.whatwg.org/#node) [appendChild](https://dom.spec.whatwg.org/#dom-node-appendchild)([Node](https://dom.spec.whatwg.org/#node) [node](https://dom.spec.whatwg.org/#dom-node-appendchild-node-node));

[CEReactions] [Node](https://dom.spec.whatwg.org/#node) [replaceChild](https://dom.spec.whatwg.org/#dom-node-replacechild)([Node](https://dom.spec.whatwg.org/#node) [node](https://dom.spec.whatwg.org/#dom-node-replacechild-node-child-node), [Node](https://dom.spec.whatwg.org/#node) [child](https://dom.spec.whatwg.org/#dom-node-replacechild-node-child-child));

[CEReactions] [Node](https://dom.spec.whatwg.org/#node) [removeChild](https://dom.spec.whatwg.org/#dom-node-removechild)([Node](https://dom.spec.whatwg.org/#node) [child](https://dom.spec.whatwg.org/#dom-node-removechild-child-child));

};

[[Constructor](https://dom.spec.whatwg.org/#dom-document-document),

Exposed=Window]

interface [Document](https://dom.spec.whatwg.org/#document) : [Node](https://dom.spec.whatwg.org/#node) {

[SameObject] readonly attribute [DOMImplementation](https://dom.spec.whatwg.org/#domimplementation) [implementation](https://dom.spec.whatwg.org/#dom-document-implementation);

readonly attribute DOMString [URL](https://dom.spec.whatwg.org/#dom-document-url);

readonly attribute DOMString [documentURI](https://dom.spec.whatwg.org/#dom-document-documenturi);

readonly attribute DOMString [origin](https://dom.spec.whatwg.org/#dom-document-origin);

readonly attribute DOMString [compatMode](https://dom.spec.whatwg.org/#dom-document-compatmode);

readonly attribute DOMString [characterSet](https://dom.spec.whatwg.org/#dom-document-characterset);

readonly attribute DOMString [charset](https://dom.spec.whatwg.org/#dom-document-charset); // historical alias of .characterSet

readonly attribute DOMString [inputEncoding](https://dom.spec.whatwg.org/#dom-document-inputencoding); // historical alias of .characterSet

readonly attribute DOMString [contentType](https://dom.spec.whatwg.org/#dom-document-contenttype);

readonly attribute [DocumentType](https://dom.spec.whatwg.org/#documenttype)? [doctype](https://dom.spec.whatwg.org/#dom-document-doctype);

readonly attribute [Element](https://dom.spec.whatwg.org/#element)? [documentElement](https://dom.spec.whatwg.org/#dom-document-documentelement);

[HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) [getElementsByTagName](https://dom.spec.whatwg.org/#dom-document-getelementsbytagname)(DOMString [qualifiedName](https://dom.spec.whatwg.org/#dom-document-getelementsbytagname-qualifiedname-qualifiedname));

[HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) [getElementsByTagNameNS](https://dom.spec.whatwg.org/#dom-document-getelementsbytagnamens)(DOMString? [namespace](https://dom.spec.whatwg.org/#dom-document-getelementsbytagnamens-namespace-localname-namespace), DOMString [localName](https://dom.spec.whatwg.org/#dom-document-getelementsbytagnamens-namespace-localname-localname));

[HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) [getElementsByClassName](https://dom.spec.whatwg.org/#dom-document-getelementsbyclassname)(DOMString [classNames](https://dom.spec.whatwg.org/#dom-document-getelementsbyclassname-classnames-classnames));

[NewObject] [Element](https://dom.spec.whatwg.org/#element) [createElement](https://dom.spec.whatwg.org/#dom-document-createelement)(DOMString [localName](https://dom.spec.whatwg.org/#dom-document-createelement-localname-options-localname), optional [ElementCreationOptions](https://dom.spec.whatwg.org/#dictdef-elementcreationoptions) [options](https://dom.spec.whatwg.org/#dom-document-createelement-localname-options-options));

[NewObject] [Element](https://dom.spec.whatwg.org/#element) [createElementNS](https://dom.spec.whatwg.org/#dom-document-createelementns)(DOMString? [namespace](https://dom.spec.whatwg.org/#dom-document-createelementns-namespace-qualifiedname-options-namespace), DOMString [qualifiedName](https://dom.spec.whatwg.org/#dom-document-createelementns-namespace-qualifiedname-options-qualifiedname), optional [ElementCreationOptions](https://dom.spec.whatwg.org/#dictdef-elementcreationoptions) [options](https://dom.spec.whatwg.org/#dom-document-createelementns-namespace-qualifiedname-options-options));

[NewObject] [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [createDocumentFragment](https://dom.spec.whatwg.org/#dom-document-createdocumentfragment)();

[NewObject] [Text](https://dom.spec.whatwg.org/#text) [createTextNode](https://dom.spec.whatwg.org/#dom-document-createtextnode)(DOMString [data](https://dom.spec.whatwg.org/#dom-document-createtextnode-data-data));

[NewObject] [Comment](https://dom.spec.whatwg.org/#comment) [createComment](https://dom.spec.whatwg.org/#dom-document-createcomment)(DOMString [data](https://dom.spec.whatwg.org/#dom-document-createcomment-data-data));

[NewObject] [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction) [createProcessingInstruction](https://dom.spec.whatwg.org/#dom-document-createprocessinginstruction)(DOMString [target](https://dom.spec.whatwg.org/#dom-document-createprocessinginstruction-target-data-target), DOMString [data](https://dom.spec.whatwg.org/#dom-document-createprocessinginstruction-target-data-data));

[CEReactions, NewObject] [Node](https://dom.spec.whatwg.org/#node) [importNode](https://dom.spec.whatwg.org/#dom-document-importnode)([Node](https://dom.spec.whatwg.org/#node) [node](https://dom.spec.whatwg.org/#dom-document-importnode-node-deep-node), optional boolean [deep](https://dom.spec.whatwg.org/#dom-document-importnode-node-deep-deep) = false);

[CEReactions] [Node](https://dom.spec.whatwg.org/#node) [adoptNode](https://dom.spec.whatwg.org/#dom-document-adoptnode)([Node](https://dom.spec.whatwg.org/#node) [node](https://dom.spec.whatwg.org/#dom-document-adoptnode-node-node));

[NewObject] [Attr](https://dom.spec.whatwg.org/#attr) [createAttribute](https://dom.spec.whatwg.org/#dom-document-createattribute)(DOMString [localName](https://dom.spec.whatwg.org/#dom-document-createattribute-localname-localname));

[NewObject] [Attr](https://dom.spec.whatwg.org/#attr) [createAttributeNS](https://dom.spec.whatwg.org/#dom-document-createattributens)(DOMString? [namespace](https://dom.spec.whatwg.org/#dom-document-createattributens-namespace-qualifiedname-namespace), DOMString [qualifiedName](https://dom.spec.whatwg.org/#dom-document-createattributens-namespace-qualifiedname-qualifiedname));

[NewObject] [Event](https://dom.spec.whatwg.org/#event) [createEvent](https://dom.spec.whatwg.org/#dom-document-createevent)(DOMString interface);

[NewObject] [Range](https://dom.spec.whatwg.org/#range) [createRange](https://dom.spec.whatwg.org/#dom-document-createrange)();

// NodeFilter.SHOW\_ALL = 0xFFFFFFFF

[NewObject] [NodeIterator](https://dom.spec.whatwg.org/#nodeiterator) [createNodeIterator](https://dom.spec.whatwg.org/#dom-document-createnodeiterator)([Node](https://dom.spec.whatwg.org/#node) [root](https://dom.spec.whatwg.org/#dom-document-createnodeiterator-root-whattoshow-filter-root), optional unsigned long [whatToShow](https://dom.spec.whatwg.org/#dom-document-createnodeiterator-root-whattoshow-filter-whattoshow) = 0xFFFFFFFF, optional [NodeFilter](https://dom.spec.whatwg.org/#callbackdef-nodefilter)? [filter](https://dom.spec.whatwg.org/#dom-document-createnodeiterator-root-whattoshow-filter-filter) = null);

[NewObject] [TreeWalker](https://dom.spec.whatwg.org/#treewalker) [createTreeWalker](https://dom.spec.whatwg.org/#dom-document-createtreewalker)([Node](https://dom.spec.whatwg.org/#node) [root](https://dom.spec.whatwg.org/#dom-document-createtreewalker-root-whattoshow-filter-root), optional unsigned long [whatToShow](https://dom.spec.whatwg.org/#dom-document-createtreewalker-root-whattoshow-filter-whattoshow) = 0xFFFFFFFF, optional [NodeFilter](https://dom.spec.whatwg.org/#callbackdef-nodefilter)? [filter](https://dom.spec.whatwg.org/#dom-document-createtreewalker-root-whattoshow-filter-filter) = null);

};

[Exposed=Window]

interface [XMLDocument](https://dom.spec.whatwg.org/#xmldocument) : [Document](https://dom.spec.whatwg.org/#document) {};

dictionary [ElementCreationOptions](https://dom.spec.whatwg.org/#dictdef-elementcreationoptions) {

DOMString [is](https://dom.spec.whatwg.org/#dom-elementcreationoptions-is);

};

[Exposed=Window]

interface [DOMImplementation](https://dom.spec.whatwg.org/#domimplementation) {

[NewObject] [DocumentType](https://dom.spec.whatwg.org/#documenttype) [createDocumentType](https://dom.spec.whatwg.org/#dom-domimplementation-createdocumenttype)(DOMString [qualifiedName](https://dom.spec.whatwg.org/#dom-domimplementation-createdocumenttype-qualifiedname-publicid-systemid-qualifiedname), DOMString [publicId](https://dom.spec.whatwg.org/#dom-domimplementation-createdocumenttype-qualifiedname-publicid-systemid-publicid), DOMString [systemId](https://dom.spec.whatwg.org/#dom-domimplementation-createdocumenttype-qualifiedname-publicid-systemid-systemid));

[NewObject] [XMLDocument](https://dom.spec.whatwg.org/#xmldocument) [createDocument](https://dom.spec.whatwg.org/#dom-domimplementation-createdocument)(DOMString? [namespace](https://dom.spec.whatwg.org/#dom-domimplementation-createdocument-namespace-qualifiedname-doctype-namespace), [TreatNullAs=EmptyString] DOMString [qualifiedName](https://dom.spec.whatwg.org/#dom-domimplementation-createdocument-namespace-qualifiedname-doctype-qualifiedname), optional [DocumentType](https://dom.spec.whatwg.org/#documenttype)? [doctype](https://dom.spec.whatwg.org/#dom-domimplementation-createdocument-namespace-qualifiedname-doctype-doctype) = null);

[NewObject] [Document](https://dom.spec.whatwg.org/#document) [createHTMLDocument](https://dom.spec.whatwg.org/#dom-domimplementation-createhtmldocument)(optional DOMString [title](https://dom.spec.whatwg.org/#dom-domimplementation-createhtmldocument-title-title));

boolean [hasFeature](https://dom.spec.whatwg.org/#dom-domimplementation-hasfeature)(); // useless; always returns true

};

[Exposed=Window]

interface [DocumentType](https://dom.spec.whatwg.org/#documenttype) : [Node](https://dom.spec.whatwg.org/#node) {

readonly attribute DOMString [name](https://dom.spec.whatwg.org/#dom-documenttype-name);

readonly attribute DOMString [publicId](https://dom.spec.whatwg.org/#dom-documenttype-publicid);

readonly attribute DOMString [systemId](https://dom.spec.whatwg.org/#dom-documenttype-systemid);

};

[[Constructor](https://dom.spec.whatwg.org/#dom-documentfragment-documentfragment),

Exposed=Window]

interface [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) : [Node](https://dom.spec.whatwg.org/#node) {

};

[Exposed=Window]

interface [ShadowRoot](https://dom.spec.whatwg.org/#shadowroot) : [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) {

readonly attribute [ShadowRootMode](https://dom.spec.whatwg.org/#enumdef-shadowrootmode) [mode](https://dom.spec.whatwg.org/#dom-shadowroot-mode);

readonly attribute [Element](https://dom.spec.whatwg.org/#element) [host](https://dom.spec.whatwg.org/#dom-shadowroot-host);

};

enum [ShadowRootMode](https://dom.spec.whatwg.org/#enumdef-shadowrootmode) { ["open"](https://dom.spec.whatwg.org/#dom-shadowrootmode-open), ["closed"](https://dom.spec.whatwg.org/#dom-shadowrootmode-closed) };

[Exposed=Window]

interface [Element](https://dom.spec.whatwg.org/#element) : [Node](https://dom.spec.whatwg.org/#node) {

readonly attribute DOMString? [namespaceURI](https://dom.spec.whatwg.org/#dom-element-namespaceuri);

readonly attribute DOMString? [prefix](https://dom.spec.whatwg.org/#dom-element-prefix);

readonly attribute DOMString [localName](https://dom.spec.whatwg.org/#dom-element-localname);

readonly attribute DOMString [tagName](https://dom.spec.whatwg.org/#dom-element-tagname);

[CEReactions] attribute DOMString [id](https://dom.spec.whatwg.org/#dom-element-id);

[CEReactions] attribute DOMString [className](https://dom.spec.whatwg.org/#dom-element-classname);

[CEReactions, SameObject, PutForwards=[value](https://dom.spec.whatwg.org/#dom-domtokenlist-value)] readonly attribute [DOMTokenList](https://dom.spec.whatwg.org/#domtokenlist) [classList](https://dom.spec.whatwg.org/#dom-element-classlist);

[CEReactions] attribute DOMString [slot](https://dom.spec.whatwg.org/#dom-element-slot);

boolean [hasAttributes](https://dom.spec.whatwg.org/#dom-element-hasattributes)();

[SameObject] readonly attribute [NamedNodeMap](https://dom.spec.whatwg.org/#namednodemap) [attributes](https://dom.spec.whatwg.org/#dom-element-attributes);

sequence<DOMString> [getAttributeNames](https://dom.spec.whatwg.org/#dom-element-getattributenames)();

DOMString? [getAttribute](https://dom.spec.whatwg.org/#dom-element-getattribute)(DOMString [qualifiedName](https://dom.spec.whatwg.org/#dom-element-getattribute-qualifiedname-qualifiedname));

DOMString? [getAttributeNS](https://dom.spec.whatwg.org/#dom-element-getattributens)(DOMString? [namespace](https://dom.spec.whatwg.org/#dom-element-getattributens-namespace-localname-namespace), DOMString [localName](https://dom.spec.whatwg.org/#dom-element-getattributens-namespace-localname-localname));

[CEReactions] void [setAttribute](https://dom.spec.whatwg.org/#dom-element-setattribute)(DOMString [qualifiedName](https://dom.spec.whatwg.org/#dom-element-setattribute-qualifiedname-value-qualifiedname), DOMString [value](https://dom.spec.whatwg.org/#dom-element-setattribute-qualifiedname-value-value));

[CEReactions] void [setAttributeNS](https://dom.spec.whatwg.org/#dom-element-setattributens)(DOMString? [namespace](https://dom.spec.whatwg.org/#dom-element-setattributens-namespace-qualifiedname-value-namespace), DOMString [qualifiedName](https://dom.spec.whatwg.org/#dom-element-setattributens-namespace-qualifiedname-value-qualifiedname), DOMString [value](https://dom.spec.whatwg.org/#dom-element-setattributens-namespace-qualifiedname-value-value));

[CEReactions] void [removeAttribute](https://dom.spec.whatwg.org/#dom-element-removeattribute)(DOMString [qualifiedName](https://dom.spec.whatwg.org/#dom-element-removeattribute-qualifiedname-qualifiedname));

[CEReactions] void [removeAttributeNS](https://dom.spec.whatwg.org/#dom-element-removeattributens)(DOMString? [namespace](https://dom.spec.whatwg.org/#dom-element-removeattributens-namespace-localname-namespace), DOMString [localName](https://dom.spec.whatwg.org/#dom-element-removeattributens-namespace-localname-localname));

boolean [hasAttribute](https://dom.spec.whatwg.org/#dom-element-hasattribute)(DOMString [qualifiedName](https://dom.spec.whatwg.org/#dom-element-hasattribute-qualifiedname-qualifiedname));

boolean [hasAttributeNS](https://dom.spec.whatwg.org/#dom-element-hasattributens)(DOMString? [namespace](https://dom.spec.whatwg.org/#dom-element-hasattributens-namespace-localname-namespace), DOMString [localName](https://dom.spec.whatwg.org/#dom-element-hasattributens-namespace-localname-localname));

[Attr](https://dom.spec.whatwg.org/#attr)? [getAttributeNode](https://dom.spec.whatwg.org/#dom-element-getattributenode)(DOMString [qualifiedName](https://dom.spec.whatwg.org/#dom-element-getattributenode-qualifiedname-qualifiedname));

[Attr](https://dom.spec.whatwg.org/#attr)? [getAttributeNodeNS](https://dom.spec.whatwg.org/#dom-element-getattributenodens)(DOMString? [namespace](https://dom.spec.whatwg.org/#dom-element-getattributenodens-namespace-localname-namespace), DOMString [localName](https://dom.spec.whatwg.org/#dom-element-getattributenodens-namespace-localname-localname));

[CEReactions] [Attr](https://dom.spec.whatwg.org/#attr)? [setAttributeNode](https://dom.spec.whatwg.org/#dom-element-setattributenode)([Attr](https://dom.spec.whatwg.org/#attr) [attr](https://dom.spec.whatwg.org/#dom-element-setattributenode-attr-attr));

[CEReactions] [Attr](https://dom.spec.whatwg.org/#attr)? [setAttributeNodeNS](https://dom.spec.whatwg.org/#dom-element-setattributenodens)([Attr](https://dom.spec.whatwg.org/#attr) [attr](https://dom.spec.whatwg.org/#dom-element-setattributenodens-attr-attr));

[CEReactions] [Attr](https://dom.spec.whatwg.org/#attr) [removeAttributeNode](https://dom.spec.whatwg.org/#dom-element-removeattributenode)([Attr](https://dom.spec.whatwg.org/#attr) [attr](https://dom.spec.whatwg.org/#dom-element-removeattributenode-attr-attr));

[ShadowRoot](https://dom.spec.whatwg.org/#shadowroot) [attachShadow](https://dom.spec.whatwg.org/#dom-element-attachshadow)([ShadowRootInit](https://dom.spec.whatwg.org/#dictdef-shadowrootinit) [init](https://dom.spec.whatwg.org/#dom-element-attachshadow-init-init));

readonly attribute [ShadowRoot](https://dom.spec.whatwg.org/#shadowroot)? [shadowRoot](https://dom.spec.whatwg.org/#dom-element-shadowroot);

[Element](https://dom.spec.whatwg.org/#element)? [closest](https://dom.spec.whatwg.org/#dom-element-closest)(DOMString [selectors](https://dom.spec.whatwg.org/#dom-element-closest-selectors-selectors));

boolean [matches](https://dom.spec.whatwg.org/#dom-element-matches)(DOMString [selectors](https://dom.spec.whatwg.org/#dom-element-matches-selectors-selectors));

boolean [webkitMatchesSelector](https://dom.spec.whatwg.org/#dom-element-webkitmatchesselector)(DOMString [selectors](https://dom.spec.whatwg.org/#dom-element-webkitmatchesselector-selectors-selectors)); // historical alias of .matches

[HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) [getElementsByTagName](https://dom.spec.whatwg.org/#dom-element-getelementsbytagname)(DOMString [qualifiedName](https://dom.spec.whatwg.org/#dom-element-getelementsbytagname-qualifiedname-qualifiedname));

[HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) [getElementsByTagNameNS](https://dom.spec.whatwg.org/#dom-element-getelementsbytagnamens)(DOMString? [namespace](https://dom.spec.whatwg.org/#dom-element-getelementsbytagnamens-namespace-localname-namespace), DOMString [localName](https://dom.spec.whatwg.org/#dom-element-getelementsbytagnamens-namespace-localname-localname));

[HTMLCollection](https://dom.spec.whatwg.org/#htmlcollection) [getElementsByClassName](https://dom.spec.whatwg.org/#dom-element-getelementsbyclassname)(DOMString [classNames](https://dom.spec.whatwg.org/#dom-element-getelementsbyclassname-classnames-classnames));

[CEReactions] [Element](https://dom.spec.whatwg.org/#element)? [insertAdjacentElement](https://dom.spec.whatwg.org/#dom-element-insertadjacentelement)(DOMString [where](https://dom.spec.whatwg.org/#dom-element-insertadjacentelement-where-element-where), [Element](https://dom.spec.whatwg.org/#element) [element](https://dom.spec.whatwg.org/#dom-element-insertadjacentelement-where-element-element)); // historical

[CEReactions] void [insertAdjacentText](https://dom.spec.whatwg.org/#dom-element-insertadjacenttext)(DOMString [where](https://dom.spec.whatwg.org/#dom-element-insertadjacenttext-where-data-where), DOMString [data](https://dom.spec.whatwg.org/#dom-element-insertadjacenttext-where-data-data)); // historical

};

dictionary [ShadowRootInit](https://dom.spec.whatwg.org/#dictdef-shadowrootinit) {

required [ShadowRootMode](https://dom.spec.whatwg.org/#enumdef-shadowrootmode) [mode](https://dom.spec.whatwg.org/#dom-shadowrootinit-mode);

};

[Exposed=Window, LegacyUnenumerableNamedProperties]

interface [NamedNodeMap](https://dom.spec.whatwg.org/#namednodemap) {

readonly attribute unsigned long [length](https://dom.spec.whatwg.org/#dom-namednodemap-length);

getter [Attr](https://dom.spec.whatwg.org/#attr)? [item](https://dom.spec.whatwg.org/#dom-namednodemap-item)(unsigned long [index](https://dom.spec.whatwg.org/#dom-namednodemap-item-index-index));

getter [Attr](https://dom.spec.whatwg.org/#attr)? [getNamedItem](https://dom.spec.whatwg.org/#dom-namednodemap-getnameditem)(DOMString [qualifiedName](https://dom.spec.whatwg.org/#dom-namednodemap-getnameditem-qualifiedname-qualifiedname));

[Attr](https://dom.spec.whatwg.org/#attr)? [getNamedItemNS](https://dom.spec.whatwg.org/#dom-namednodemap-getnameditemns)(DOMString? [namespace](https://dom.spec.whatwg.org/#dom-namednodemap-getnameditemns-namespace-localname-namespace), DOMString [localName](https://dom.spec.whatwg.org/#dom-namednodemap-getnameditemns-namespace-localname-localname));

[CEReactions] [Attr](https://dom.spec.whatwg.org/#attr)? [setNamedItem](https://dom.spec.whatwg.org/#dom-namednodemap-setnameditem)([Attr](https://dom.spec.whatwg.org/#attr) [attr](https://dom.spec.whatwg.org/#dom-namednodemap-setnameditem-attr-attr));

[CEReactions] [Attr](https://dom.spec.whatwg.org/#attr)? [setNamedItemNS](https://dom.spec.whatwg.org/#dom-namednodemap-setnameditemns)([Attr](https://dom.spec.whatwg.org/#attr) [attr](https://dom.spec.whatwg.org/#dom-namednodemap-setnameditemns-attr-attr));

[CEReactions] [Attr](https://dom.spec.whatwg.org/#attr) [removeNamedItem](https://dom.spec.whatwg.org/#dom-namednodemap-removenameditem)(DOMString [qualifiedName](https://dom.spec.whatwg.org/#dom-namednodemap-removenameditem-qualifiedname-qualifiedname));

[CEReactions] [Attr](https://dom.spec.whatwg.org/#attr) [removeNamedItemNS](https://dom.spec.whatwg.org/#dom-namednodemap-removenameditemns)(DOMString? [namespace](https://dom.spec.whatwg.org/#dom-namednodemap-removenameditemns-namespace-localname-namespace), DOMString [localName](https://dom.spec.whatwg.org/#dom-namednodemap-removenameditemns-namespace-localname-localname));

};

[Exposed=Window]

interface [Attr](https://dom.spec.whatwg.org/#attr) {

readonly attribute DOMString? [namespaceURI](https://dom.spec.whatwg.org/#dom-attr-namespaceuri);

readonly attribute DOMString? [prefix](https://dom.spec.whatwg.org/#dom-attr-prefix);

readonly attribute DOMString [localName](https://dom.spec.whatwg.org/#dom-attr-localname);

readonly attribute DOMString [name](https://dom.spec.whatwg.org/#dom-attr-name);

readonly attribute DOMString [nodeName](https://dom.spec.whatwg.org/#dom-attr-nodename); // historical alias of .name

[CEReactions] attribute DOMString [value](https://dom.spec.whatwg.org/#dom-attr-value);

[CEReactions, TreatNullAs=EmptyString] attribute DOMString [nodeValue](https://dom.spec.whatwg.org/#dom-attr-nodevalue); // historical alias of .value

[CEReactions, TreatNullAs=EmptyString] attribute DOMString [textContent](https://dom.spec.whatwg.org/#dom-attr-textcontent); // historical alias of .value

readonly attribute [Element](https://dom.spec.whatwg.org/#element)? [ownerElement](https://dom.spec.whatwg.org/#dom-attr-ownerelement);

readonly attribute boolean [specified](https://dom.spec.whatwg.org/#dom-attr-specified); // useless; always returns true

};

[Exposed=Window]

interface [CharacterData](https://dom.spec.whatwg.org/#characterdata) : [Node](https://dom.spec.whatwg.org/#node) {

[TreatNullAs=EmptyString] attribute DOMString [data](https://dom.spec.whatwg.org/#dom-characterdata-data);

readonly attribute unsigned long [length](https://dom.spec.whatwg.org/#dom-characterdata-length);

DOMString [substringData](https://dom.spec.whatwg.org/#dom-characterdata-substringdata)(unsigned long [offset](https://dom.spec.whatwg.org/#dom-characterdata-substringdata-offset-count-offset), unsigned long [count](https://dom.spec.whatwg.org/#dom-characterdata-substringdata-offset-count-count));

void [appendData](https://dom.spec.whatwg.org/#dom-characterdata-appenddata)(DOMString [data](https://dom.spec.whatwg.org/#dom-characterdata-appenddata-data-data));

void [insertData](https://dom.spec.whatwg.org/#dom-characterdata-insertdata)(unsigned long [offset](https://dom.spec.whatwg.org/#dom-characterdata-insertdata-offset-data-offset), DOMString [data](https://dom.spec.whatwg.org/#dom-characterdata-insertdata-offset-data-data));

void [deleteData](https://dom.spec.whatwg.org/#dom-characterdata-deletedata)(unsigned long [offset](https://dom.spec.whatwg.org/#dom-characterdata-deletedata-offset-count-offset), unsigned long [count](https://dom.spec.whatwg.org/#dom-characterdata-deletedata-offset-count-count));

void [replaceData](https://dom.spec.whatwg.org/#dom-characterdata-replacedata)(unsigned long [offset](https://dom.spec.whatwg.org/#dom-characterdata-replacedata-offset-count-data-offset), unsigned long [count](https://dom.spec.whatwg.org/#dom-characterdata-replacedata-offset-count-data-count), DOMString [data](https://dom.spec.whatwg.org/#dom-characterdata-replacedata-offset-count-data-data));

};

[[Constructor](https://dom.spec.whatwg.org/#dom-text-text)(optional DOMString [data](https://dom.spec.whatwg.org/#dom-text-text-data-data) = ""),

Exposed=Window]

interface [Text](https://dom.spec.whatwg.org/#text) : [CharacterData](https://dom.spec.whatwg.org/#characterdata) {

[NewObject] [Text](https://dom.spec.whatwg.org/#text) [splitText](https://dom.spec.whatwg.org/#dom-text-splittext)(unsigned long [offset](https://dom.spec.whatwg.org/#dom-text-splittext-offset-offset));

readonly attribute DOMString [wholeText](https://dom.spec.whatwg.org/#dom-text-wholetext);

};

[Exposed=Window]

interface [ProcessingInstruction](https://dom.spec.whatwg.org/#processinginstruction) : [CharacterData](https://dom.spec.whatwg.org/#characterdata) {

readonly attribute DOMString [target](https://dom.spec.whatwg.org/#dom-processinginstruction-target);

};

[[Constructor](https://dom.spec.whatwg.org/#dom-comment-comment)(optional DOMString [data](https://dom.spec.whatwg.org/#dom-comment-comment-data-data) = ""),

Exposed=Window]

interface [Comment](https://dom.spec.whatwg.org/#comment) : [CharacterData](https://dom.spec.whatwg.org/#characterdata) {

};

[[Constructor](https://dom.spec.whatwg.org/#dom-range-range),

Exposed=Window]

interface [Range](https://dom.spec.whatwg.org/#range) {

readonly attribute [Node](https://dom.spec.whatwg.org/#node) [startContainer](https://dom.spec.whatwg.org/#dom-range-startcontainer);

readonly attribute unsigned long [startOffset](https://dom.spec.whatwg.org/#dom-range-startoffset);

readonly attribute [Node](https://dom.spec.whatwg.org/#node) [endContainer](https://dom.spec.whatwg.org/#dom-range-endcontainer);

readonly attribute unsigned long [endOffset](https://dom.spec.whatwg.org/#dom-range-endoffset);

readonly attribute boolean [collapsed](https://dom.spec.whatwg.org/#dom-range-collapsed);

readonly attribute [Node](https://dom.spec.whatwg.org/#node) [commonAncestorContainer](https://dom.spec.whatwg.org/#dom-range-commonancestorcontainer);

void [setStart](https://dom.spec.whatwg.org/#dom-range-setstart)([Node](https://dom.spec.whatwg.org/#node) [node](https://dom.spec.whatwg.org/#dom-range-setstart-node-offset-node), unsigned long [offset](https://dom.spec.whatwg.org/#dom-range-setstart-node-offset-offset));

void [setEnd](https://dom.spec.whatwg.org/#dom-range-setend)([Node](https://dom.spec.whatwg.org/#node) [node](https://dom.spec.whatwg.org/#dom-range-setend-node-offset-node), unsigned long [offset](https://dom.spec.whatwg.org/#dom-range-setend-node-offset-offset));

void [setStartBefore](https://dom.spec.whatwg.org/#dom-range-setstartbefore)([Node](https://dom.spec.whatwg.org/#node) [node](https://dom.spec.whatwg.org/#dom-range-setstartbefore-node-node));

void [setStartAfter](https://dom.spec.whatwg.org/#dom-range-setstartafter)([Node](https://dom.spec.whatwg.org/#node) [node](https://dom.spec.whatwg.org/#dom-range-setstartafter-node-node));

void [setEndBefore](https://dom.spec.whatwg.org/#dom-range-setendbefore)([Node](https://dom.spec.whatwg.org/#node) [node](https://dom.spec.whatwg.org/#dom-range-setendbefore-node-node));

void [setEndAfter](https://dom.spec.whatwg.org/#dom-range-setendafter)([Node](https://dom.spec.whatwg.org/#node) [node](https://dom.spec.whatwg.org/#dom-range-setendafter-node-node));

void [collapse](https://dom.spec.whatwg.org/#dom-range-collapse)(optional boolean [toStart](https://dom.spec.whatwg.org/#dom-range-collapse-tostart-tostart) = false);

void [selectNode](https://dom.spec.whatwg.org/#dom-range-selectnode)([Node](https://dom.spec.whatwg.org/#node) [node](https://dom.spec.whatwg.org/#dom-range-selectnode-node-node));

void [selectNodeContents](https://dom.spec.whatwg.org/#dom-range-selectnodecontents)([Node](https://dom.spec.whatwg.org/#node) [node](https://dom.spec.whatwg.org/#dom-range-selectnodecontents-node-node));

const unsigned short [START\_TO\_START](https://dom.spec.whatwg.org/#dom-range-start_to_start) = 0;

const unsigned short [START\_TO\_END](https://dom.spec.whatwg.org/#dom-range-start_to_end) = 1;

const unsigned short [END\_TO\_END](https://dom.spec.whatwg.org/#dom-range-end_to_end) = 2;

const unsigned short [END\_TO\_START](https://dom.spec.whatwg.org/#dom-range-end_to_start) = 3;

short [compareBoundaryPoints](https://dom.spec.whatwg.org/#dom-range-compareboundarypoints)(unsigned short [how](https://dom.spec.whatwg.org/#dom-range-compareboundarypoints-how-sourcerange-how), [Range](https://dom.spec.whatwg.org/#range) [sourceRange](https://dom.spec.whatwg.org/#dom-range-compareboundarypoints-how-sourcerange-sourcerange));

[CEReactions] void [deleteContents](https://dom.spec.whatwg.org/#dom-range-deletecontents)();

[CEReactions, NewObject] [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [extractContents](https://dom.spec.whatwg.org/#dom-range-extractcontents)();

[CEReactions, NewObject] [DocumentFragment](https://dom.spec.whatwg.org/#documentfragment) [cloneContents](https://dom.spec.whatwg.org/#dom-range-clonecontents)();

[CEReactions] void [insertNode](https://dom.spec.whatwg.org/#dom-range-insertnode)([Node](https://dom.spec.whatwg.org/#node) [node](https://dom.spec.whatwg.org/#dom-range-insertnode-node-node));

[CEReactions] void [surroundContents](https://dom.spec.whatwg.org/#dom-range-surroundcontents)([Node](https://dom.spec.whatwg.org/#node) [newParent](https://dom.spec.whatwg.org/#dom-range-surroundcontents-newparent-newparent));

[NewObject] [Range](https://dom.spec.whatwg.org/#range) [cloneRange](https://dom.spec.whatwg.org/#dom-range-clonerange)();

void [detach](https://dom.spec.whatwg.org/#dom-range-detach)();

boolean [isPointInRange](https://dom.spec.whatwg.org/#dom-range-ispointinrange)([Node](https://dom.spec.whatwg.org/#node) [node](https://dom.spec.whatwg.org/#dom-range-ispointinrange-node-offset-node), unsigned long [offset](https://dom.spec.whatwg.org/#dom-range-ispointinrange-node-offset-offset));

short [comparePoint](https://dom.spec.whatwg.org/#dom-range-comparepoint)([Node](https://dom.spec.whatwg.org/#node) [node](https://dom.spec.whatwg.org/#dom-range-comparepoint-node-offset-node), unsigned long [offset](https://dom.spec.whatwg.org/#dom-range-comparepoint-node-offset-offset));

boolean [intersectsNode](https://dom.spec.whatwg.org/#dom-range-intersectsnode)([Node](https://dom.spec.whatwg.org/#node) [node](https://dom.spec.whatwg.org/#dom-range-intersectsnode-node-node));

[stringifier](https://dom.spec.whatwg.org/#dom-range-stringifier);

};

[Exposed=Window]

interface [NodeIterator](https://dom.spec.whatwg.org/#nodeiterator) {

[SameObject] readonly attribute [Node](https://dom.spec.whatwg.org/#node) [root](https://dom.spec.whatwg.org/#dom-nodeiterator-root);

readonly attribute [Node](https://dom.spec.whatwg.org/#node) [referenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-referencenode);

readonly attribute boolean [pointerBeforeReferenceNode](https://dom.spec.whatwg.org/#dom-nodeiterator-pointerbeforereferencenode);

readonly attribute unsigned long [whatToShow](https://dom.spec.whatwg.org/#dom-nodeiterator-whattoshow);

readonly attribute [NodeFilter](https://dom.spec.whatwg.org/#callbackdef-nodefilter)? [filter](https://dom.spec.whatwg.org/#dom-nodeiterator-filter);

[Node](https://dom.spec.whatwg.org/#node)? [nextNode](https://dom.spec.whatwg.org/#dom-nodeiterator-nextnode)();

[Node](https://dom.spec.whatwg.org/#node)? [previousNode](https://dom.spec.whatwg.org/#dom-nodeiterator-previousnode)();

void [detach](https://dom.spec.whatwg.org/#dom-nodeiterator-detach)();

};

[Exposed=Window]

interface [TreeWalker](https://dom.spec.whatwg.org/#treewalker) {

[SameObject] readonly attribute [Node](https://dom.spec.whatwg.org/#node) [root](https://dom.spec.whatwg.org/#dom-treewalker-root);

readonly attribute unsigned long [whatToShow](https://dom.spec.whatwg.org/#dom-treewalker-whattoshow);

readonly attribute [NodeFilter](https://dom.spec.whatwg.org/#callbackdef-nodefilter)? [filter](https://dom.spec.whatwg.org/#dom-treewalker-filter);

attribute [Node](https://dom.spec.whatwg.org/#node) [currentNode](https://dom.spec.whatwg.org/#dom-treewalker-currentnode);

[Node](https://dom.spec.whatwg.org/#node)? [parentNode](https://dom.spec.whatwg.org/#dom-treewalker-parentnode)();

[Node](https://dom.spec.whatwg.org/#node)? [firstChild](https://dom.spec.whatwg.org/#dom-treewalker-firstchild)();

[Node](https://dom.spec.whatwg.org/#node)? [lastChild](https://dom.spec.whatwg.org/#dom-treewalker-lastchild)();

[Node](https://dom.spec.whatwg.org/#node)? [previousSibling](https://dom.spec.whatwg.org/#dom-treewalker-previoussibling)();

[Node](https://dom.spec.whatwg.org/#node)? [nextSibling](https://dom.spec.whatwg.org/#dom-treewalker-nextsibling)();

[Node](https://dom.spec.whatwg.org/#node)? [previousNode](https://dom.spec.whatwg.org/#dom-treewalker-previousnode)();

[Node](https://dom.spec.whatwg.org/#node)? [nextNode](https://dom.spec.whatwg.org/#dom-treewalker-nextnode)();

};

[Exposed=Window]

callback interface [NodeFilter](https://dom.spec.whatwg.org/#callbackdef-nodefilter) {

// Constants for acceptNode()

const unsigned short [FILTER\_ACCEPT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_accept) = 1;

const unsigned short [FILTER\_REJECT](https://dom.spec.whatwg.org/#dom-nodefilter-filter_reject) = 2;

const unsigned short [FILTER\_SKIP](https://dom.spec.whatwg.org/#dom-nodefilter-filter_skip) = 3;

// Constants for whatToShow

const unsigned long [SHOW\_ALL](https://dom.spec.whatwg.org/#dom-nodefilter-show_all) = 0xFFFFFFFF;

const unsigned long [SHOW\_ELEMENT](https://dom.spec.whatwg.org/#dom-nodefilter-show_element) = 0x1;

const unsigned long [SHOW\_ATTRIBUTE](https://dom.spec.whatwg.org/#dom-nodefilter-show_attribute) = 0x2; // historical

const unsigned long [SHOW\_TEXT](https://dom.spec.whatwg.org/#dom-nodefilter-show_text) = 0x4;

const unsigned long [SHOW\_CDATA\_SECTION](https://dom.spec.whatwg.org/#dom-nodefilter-show_cdata_section) = 0x8; // historical

const unsigned long [SHOW\_ENTITY\_REFERENCE](https://dom.spec.whatwg.org/#dom-nodefilter-show_entity_reference) = 0x10; // historical

const unsigned long [SHOW\_ENTITY](https://dom.spec.whatwg.org/#dom-nodefilter-show_entity) = 0x20; // historical

const unsigned long [SHOW\_PROCESSING\_INSTRUCTION](https://dom.spec.whatwg.org/#dom-nodefilter-show_processing_instruction) = 0x40;

const unsigned long [SHOW\_COMMENT](https://dom.spec.whatwg.org/#dom-nodefilter-show_comment) = 0x80;

const unsigned long [SHOW\_DOCUMENT](https://dom.spec.whatwg.org/#dom-nodefilter-show_document) = 0x100;

const unsigned long [SHOW\_DOCUMENT\_TYPE](https://dom.spec.whatwg.org/#dom-nodefilter-show_document_type) = 0x200;

const unsigned long [SHOW\_DOCUMENT\_FRAGMENT](https://dom.spec.whatwg.org/#dom-nodefilter-show_document_fragment) = 0x400;

const unsigned long [SHOW\_NOTATION](https://dom.spec.whatwg.org/#dom-nodefilter-show_notation) = 0x800; // historical

unsigned short [acceptNode](https://dom.spec.whatwg.org/#dom-nodefilter-acceptnode)([Node](https://dom.spec.whatwg.org/#node) [node](https://dom.spec.whatwg.org/#dom-nodefilter-acceptnode-node-node));

};

interface [DOMTokenList](https://dom.spec.whatwg.org/#domtokenlist) {

readonly attribute unsigned long [length](https://dom.spec.whatwg.org/#dom-domtokenlist-length);

getter DOMString? [item](https://dom.spec.whatwg.org/#dom-domtokenlist-item)(unsigned long [index](https://dom.spec.whatwg.org/#dom-domtokenlist-item-index-index));

boolean [contains](https://dom.spec.whatwg.org/#dom-domtokenlist-contains)(DOMString [token](https://dom.spec.whatwg.org/#dom-domtokenlist-contains-token-token));

[CEReactions] void [add](https://dom.spec.whatwg.org/#dom-domtokenlist-add)(DOMString... [tokens](https://dom.spec.whatwg.org/#dom-domtokenlist-add-tokens-tokens));

[CEReactions] void [remove](https://dom.spec.whatwg.org/#dom-domtokenlist-remove)(DOMString... [tokens](https://dom.spec.whatwg.org/#dom-domtokenlist-remove-tokens-tokens));

[CEReactions] boolean [toggle](https://dom.spec.whatwg.org/#dom-domtokenlist-toggle)(DOMString [token](https://dom.spec.whatwg.org/#dom-domtokenlist-toggle-token-force-token), optional boolean [force](https://dom.spec.whatwg.org/#dom-domtokenlist-toggle-token-force-force));

[CEReactions] void [replace](https://dom.spec.whatwg.org/#dom-domtokenlist-replace)(DOMString [token](https://dom.spec.whatwg.org/#dom-domtokenlist-replace-token-newtoken-token), DOMString [newToken](https://dom.spec.whatwg.org/#dom-domtokenlist-replace-token-newtoken-newtoken));

boolean [supports](https://dom.spec.whatwg.org/#dom-domtokenlist-supports)(DOMString [token](https://dom.spec.whatwg.org/#dom-domtokenlist-supports-token-token));

[CEReactions] attribute DOMString [value](https://dom.spec.whatwg.org/#dom-domtokenlist-value);

[stringifier](https://dom.spec.whatwg.org/#dom-domtokenlist-stringifier);

iterable<DOMString>;

};