The **`Document`** interface represents any web page loaded in the browser and serves as an entry point into the web page's content, which is the DOM tree. The DOM tree includes elements such as `<body>` and `<table>`, among many others. It provides functionality globally to the document, like how to obtain the page's URL and create new elements in the document.

The `Document` interface describes the common properties and methods for any kind of document. Depending on the document's type (e.g. HTML, XML, SVG, …), a larger API is available: HTML documents, served with the `"text/html"` content type, also implement the `HTMLDocument` interface, whereas XML and SVG documents implement the `XMLDocument` interface.

Constructor

`Document()`
Creates a new `Document` object.

Properties

This interface also inherits from the `Node` and `EventTarget` interfaces.

`Document.anchors` | Read only
Returns a list of all of the anchors in the document.

`Document.body`
Returns the `<body>` or `<frameset>` node of the current document.

`Document.characterSet` | Read only
Returns the character set being used by the document.

`Document.compatMode` | Read only
Indicates whether the document is rendered in *quirks* or *strict* mode.

`Document.contentType` | Read only
Returns the Content-Type from the MIME Header of the current document.

`Document.doctype` | Read only

Returns the Document Type Definition (DTD) of the current document.

`Document.documentElement` | Read only

Returns the `Element` that is a direct child of the document. For HTML documents, this is normally the `HTMLHtmlElement` object representing the document's `<html>` element.

`Document.documentURI` | Read only

Returns the document location as a string.

`Document.embeds` | Read only

Returns a list of the embedded `<embed>` elements within the current document.

`Document.fonts`

Returns the `FontFaceSet` interface of the current document.

`Document.forms` | Read only

Returns a list of the `<form>` elements within the current document.

`Document.head` | Read only

Returns the `<head>` element of the current document.

`Document.hidden` | Read only

Returns a Boolean value indicating if the page is considered hidden or not.

`Document.images` | Read only

Returns a list of the images in the current document.

`Document.implementation` | Read only

Returns the DOM implementation associated with the current document.

`Document.lastStyleSheetSet` | Read only

Returns the name of the style sheet set that was last enabled. Has the value `null` until the style sheet is changed by setting the value of `selectedStyleSheetSet`.

`Document.links` | Read only

Returns a list of all the hyperlinks in the document.

`Document.mozSyntheticDocument`

Returns a `Boolean` that is `true` only if this document is synthetic, such as a standalone image, video, audio file, or the like.

`Document.plugins` | Read only

Returns a list of the available plugins.

`Document.featurePolicy` | Read only

Returns the `FeaturePolicy` interface which provides a simple API for introspecting the feature policies applied to a specific document.

`Document.preferredStyleSheetSet` | Read only

Returns the preferred style sheet set as specified by the page author.

`Document.scripts` | Read only

Returns all the `<script>` elements on the document.

`Document.scrollingElement` | Read only

Returns a reference to the `Element` that scrolls the document.

`Document.selectedStyleSheetSet`

Returns which style sheet set is currently in use.

`Document.styleSheetSets` | Read only

Returns a list of the style sheet sets available on the document.

`Document.timeline` | Read only

Returns timeline as a special instance of `DocumentTimeline` that is automatically created on page load.

`Document.undoManager` | Read only

…

`Document.visibilityState` | Read only

Returns a `string` denoting the visibility state of the document. Possible values are `visible`, `hidden`, `prerender`, and `unloaded`.

The `Document` interface is extended with the `ParentNode` interface:

`ParentNode.childElementCount` | Read only

Returns the number of children of this `ParentNode` which are elements.

`ParentNode.children` | Read only

Returns a live `HTMLCollection` containing all of the `Element` objects that are children of this `ParentNode`, omitting all of its non-element nodes.

`ParentNode.firstElementChild` | Read only

Returns the first node which is both a child of this `ParentNode` *and* is also an `Element`, or `null` if there is none.

`ParentNode.lastElementChild` | Read only

Returns the last node which is both a child of this `ParentNode` *and* is an `Element`, or `null` if there is none.

Extensions for HTMLDocument

The `Document` interface for HTML documents inherits from the `HTMLDocument` interface or, since HTML5, is extended for such documents.

`Document.cookie`

Returns a semicolon-separated list of the cookies for that document or sets a single cookie.

`Document.defaultView` | Read only

Returns a reference to the window object.

`Document.designMode`

Gets/sets the ability to edit the whole document.

`Document.dir` | Read only

Gets/sets directionality (rtl/ltr) of the document.

`Document.domain`

Gets/sets the domain of the current document.

`Document.lastModified` | Read only

Returns the date on which the document was last modified.

`Document.location` | Read only

Returns the URI of the current document.

`Document.readyState` | Read only

Returns loading status of the document.

`Document.referrer` | Read only

Returns the URI of the page that linked to this page.

`Document.title`

Sets or gets the title of the current document.

Document.URL | Read only

Returns the document location as a string.

Properties included from DocumentOrShadowRoot

The `Document` interface includes the following properties defined on the `DocumentOrShadowRoot` mixin. Note that this is currently only implemented by Chrome; other browsers still implement them directly on the `Document` interface.

DocumentOrShadowRoot.activeElement | Read only

Returns the `Element` within the shadow tree that has focus.

Document.fullscreenElement | Read only

The element that's currently in full screen mode for this document.

DocumentOrShadowRoot.pointerLockElement | Read only

Returns the element set as the target for mouse events while the pointer is locked. `null` if lock is pending, pointer is unlocked, or if the target is in another document.

DocumentOrShadowRoot.styleSheets | Read only

Returns a `StyleSheetList` of `CSSStyleSheet` objects for stylesheets explicitly linked into, or embedded in a document.

Event handlers

Document.onafterscriptexecute

Represents the event handling code for the `afterscriptexecute` event.

Document.onbeforescriptexecute

Represents the event handling code for the `beforescriptexecute` event.

Document.oncopy

Represents the event handling code for the `copy` event.

Document.oncut

Represents the event handling code for the `cut` event.

Document.onfullscreenchange

Is an `EventHandler` representing the code to be called when the `fullscreenchange` event is raised.

Document.onfullscreenerror

Is an `EventHandler` representing the code to be called when the `fullscreenerror` event is raised.

`Document.onpaste`
Represents the event handling code for the `paste` event.

`Document.onreadystatechange`
Represents the event handling code for the `readystatechange` event.

`Document.onselectionchange`
Is an `EventHandler` representing the code to be called when the `selectionchange` event is raised.

`Document.onvisibilitychange`
Is an `EventHandler` representing the code to be called when the `visibilitychange` event is raised.

The `Document` interface is extended with the `GlobalEventHandlers` interface:

`GlobalEventHandlers.onabort`
Is an `EventHandler` representing the code to be called when the `abort` event is raised.

`GlobalEventHandlers.onanimationcancel`
An `EventHandler` called when an `animationcancel` event is sent, indicating that a running CSS animation has been canceled.

`GlobalEventHandlers.onanimationend`
An `EventHandler` called when an `animationend` event is sent, indicating that a CSS animationhas stopped playing.

`GlobalEventHandlers.onanimationiteration`
An `EventHandler` called when an `animationiteration` event has been sent, indicating that a CSS animation has begun playing a new iteration of the animation sequence.

`GlobalEventHandlers.onanimationstart`
An `EventHandler` called when an `animationstart` event is sent, indicating that a CSS animation has started playing.

`GlobalEventHandlers.onauxclick`
An `EventHandler` called when an `auxclick` event is sent, indicating that a non-primary button has been pressed on an input device (e.g. a middle mouse button).

`GlobalEventHandlers.onblur`
Is an `EventHandler` representing the code to be called when the `blur` event is raised.

`GlobalEventHandlers.onerror`
Is an `OnErrorEventHandler` representing the code to be called when the `error` event is raised.

GlobalEventHandlers.onfocus
Is an `EventHandler` representing the code to be called when the `focus` event is raised.

GlobalEventHandlers.oncancel
Is an `EventHandler` representing the code to be called when the `cancel` event is raised.

GlobalEventHandlers.oncanplay
Is an `EventHandler` representing the code to be called when the `canplay` event is raised.

GlobalEventHandlers.oncanplaythrough
Is an `EventHandler` representing the code to be called when the `canplaythrough` event is raised.

GlobalEventHandlers.onchange
Is an `EventHandler` representing the code to be called when the `change` event is raised.

GlobalEventHandlers.onclick
Is an `EventHandler` representing the code to be called when the `click` event is raised.

GlobalEventHandlers.onclose
Is an `EventHandler` representing the code to be called when the `close` event is raised.

GlobalEventHandlers.oncontextmenu
Is an `EventHandler` representing the code to be called when the `contextmenu` event is raised.

GlobalEventHandlers.oncuechange
Is an `EventHandler` representing the code to be called when the `cuechange` event is raised.

GlobalEventHandlers.ondblclick
Is an `EventHandler` representing the code to be called when the `dblclick` event is raised.

GlobalEventHandlers.ondrag
Is an `EventHandler` representing the code to be called when the `drag` event is raised.

GlobalEventHandlers.ondragend
Is an `EventHandler` representing the code to be called when the `dragend` event is raised.

GlobalEventHandlers.ondragenter
Is an `EventHandler` representing the code to be called when the `dragenter` event is raised.

GlobalEventHandlers.ondragexit
Is an `EventHandler` representing the code to be called when the `dragexit` event is raised.

GlobalEventHandlers.ondragleave
Is an `EventHandler` representing the code to be called when the `dragleave` event is raised.

GlobalEventHandlers.ondragover
Is an `EventHandler` representing the code to be called when the `dragover` event is raised.

GlobalEventHandlers.ondragstart
Is an `EventHandler` representing the code to be called when the `dragstart` event is raised.

GlobalEventHandlers.ondrop
Is an `EventHandler` representing the code to be called when the `drop` event is raised.

GlobalEventHandlers.ondurationchange
Is an `EventHandler` representing the code to be called when the `durationchange` event is raised.

GlobalEventHandlers.onemptied
Is an `EventHandler` representing the code to be called when the `emptied` event is raised.

GlobalEventHandlers.onended
Is an `EventHandler` representing the code to be called when the `ended` event is raised.

GlobalEventHandlers.onformdata
Is an `EventHandler` for processing `formdata` events, fired after the entry list representing the form's data is constructed.

GlobalEventHandlers.ongotpointercapture
Is an `EventHandler` representing the code to be called when the `gotpointercapture` event type is raised.

GlobalEventHandlers.oninput
Is an `EventHandler` representing the code to be called when the `input` event is raised.

GlobalEventHandlers.oninvalid
Is an `EventHandler` representing the code to be called when the `invalid` event is raised.

GlobalEventHandlers.onkeydown
Is an `EventHandler` representing the code to be called when the `keydown` event is raised.

GlobalEventHandlers.onkeypress
Is an `EventHandler` representing the code to be called when the `keypress` event is raised.

GlobalEventHandlers.onkeyup
Is an `EventHandler` representing the code to be called when the `keyup` event is raised.

GlobalEventHandlers.onload
Is an `EventHandler` representing the code to be called when the `load` event is raised.

GlobalEventHandlers.onloadeddata
Is an `EventHandler` representing the code to be called when the `loadeddata` event is raised.

GlobalEventHandlers.onloadedmetadata
Is an `EventHandler` representing the code to be called when the `loadedmetadata` event is raised.

GlobalEventHandlers.onloadend
Is an `EventHandler` representing the code to be called when the `loadend` event is raised (when progress has stopped on the loading of a resource.)

GlobalEventHandlers.onloadstart
Is an `EventHandler` representing the code to be called when the `loadstart` event is raised (when progress has begun on the loading of a resource.)

GlobalEventHandlers.onlostpointercapture
Is an `EventHandler` representing the code to be called when the `lostpointercapture` event type is raised.

GlobalEventHandlers.onmousedown
Is an `EventHandler` representing the code to be called when the `mousedown` event is raised.

GlobalEventHandlers.onmouseenter
Is an `EventHandler` representing the code to be called when the `mouseenter` event is raised.

GlobalEventHandlers.onmouseleave
Is an `EventHandler` representing the code to be called when the `mouseleave` event is raised.

GlobalEventHandlers.onmousemove
Is an `EventHandler` representing the code to be called when the `mousemove` event is raised.

GlobalEventHandlers.onmouseout
Is an `EventHandler` representing the code to be called when the `mouseout` event is raised.

GlobalEventHandlers.onmouseover
Is an `EventHandler` representing the code to be called when the `mouseover` event is raised.

GlobalEventHandlers.onmouseup
Is an `EventHandler` representing the code to be called when the `mouseup` event is raised.

GlobalEventHandlers.onmousewheel
Is an `EventHandler` representing the code to be called when the `mousewheel` event is raised. Deprecated. Use `onwheel` instead.

GlobalEventHandlers.onwheel
Is an `EventHandler` representing the code to be called when the `wheel` event is raised.

GlobalEventHandlers.onpause
Is an `EventHandler` representing the code to be called when the `pause` event is raised.

GlobalEventHandlers.onplay
Is an `EventHandler` representing the code to be called when the `play` event is raised.

GlobalEventHandlers.onplaying

Is an `EventHandler` representing the code to be called when the `playing` event is raised.

`GlobalEventHandlers.onpointerdown`
Is an `EventHandler` representing the code to be called when the `pointerdown` event is raised.

`GlobalEventHandlers.onpointermove`
Is an `EventHandler` representing the code to be called when the `pointermove` event is raised.

`GlobalEventHandlers.onpointerup`
Is an `EventHandler` representing the code to be called when the `pointerup` event is raised.

`GlobalEventHandlers.onpointercancel`
Is an `EventHandler` representing the code to be called when the `pointercancel` event is raised.

`GlobalEventHandlers.onpointerover`
Is an `EventHandler` representing the code to be called when the `pointerover` event is raised.

`GlobalEventHandlers.onpointerout`
Is an `EventHandler` representing the code to be called when the `pointerout` event is raised.

`GlobalEventHandlers.onpointerenter`
Is an `EventHandler` representing the code to be called when the `pointerenter` event is raised.

`GlobalEventHandlers.onpointerleave`
Is an `EventHandler` representing the code to be called when the `pointerleave` event is raised.

`GlobalEventHandlers.onpointerlockchange`
Is an `EventHandler` representing the code to be called when the `pointerlockchange` event is raised.

`GlobalEventHandlers.onpointerlockerror`
Is an `EventHandler` representing the code to be called when the `pointerlockerror` event is raised.

`GlobalEventHandlers.onprogress`
Is an `EventHandler` representing the code to be called when the `progress` event is raised.

`GlobalEventHandlers.onratechange`
Is an `EventHandler` representing the code to be called when the `ratechange` event is raised.

`GlobalEventHandlers.onreset`
Is an `EventHandler` representing the code to be called when the `reset` event is raised.

`GlobalEventHandlers.onresize`
Is an `EventHandler` representing the code to be called when the `resize` event is raised.

`GlobalEventHandlers.onscroll`

Is an `EventHandler` representing the code to be called when the `scroll` event is raised.

`GlobalEventHandlers.onseeked`
Is an `EventHandler` representing the code to be called when the `seeked` event is raised.

`GlobalEventHandlers.onseeking`
Is an `EventHandler` representing the code to be called when the `seeking` event is raised.

`GlobalEventHandlers.onselect`
Is an `EventHandler` representing the code to be called when the `select` event is raised.

`GlobalEventHandlers.onselectstart`
Is an `EventHandler` representing the code to be called when the `selectionchange` event is raised, i.e. when the user starts to make a new text selection on a web page.

`GlobalEventHandlers.onselectionchange`
Is an `EventHandler` representing the code to be called when the `selectionchange` event is raised, i.e. when the text selected on a web page changes.

`GlobalEventHandlers.onshow`
Is an `EventHandler` representing the code to be called when the `show` event is raised.

`GlobalEventHandlers.onsort`
Is an `EventHandler` representing the code to be called when the `sort` event is raised.

`GlobalEventHandlers.onstalled`
Is an `EventHandler` representing the code to be called when the `stalled` event is raised.

`GlobalEventHandlers.onsubmit`
Is an `EventHandler` representing the code to be called when the `submit` event is raised.

`GlobalEventHandlers.onsuspend`
Is an `EventHandler` representing the code to be called when the `suspend` event is raised.

`GlobalEventHandlers.ontimeupdate`
Is an `EventHandler` representing the code to be called when the `timeupdate` event is raised.

`GlobalEventHandlers.onvolumechange`
Is an `EventHandler` representing the code to be called when the `volumechange` event is raised.

`GlobalEventHandlers.ontouchcancel`
Is an `EventHandler` representing the code to be called when the `touchcancel` event is raised.

`GlobalEventHandlers.ontouchend`
Is an `EventHandler` representing the code to be called when the `touchend` event is raised.

`GlobalEventHandlers.ontouchmove`

Is an `EventHandler` representing the code to be called when the `touchmove` event is raised.

`GlobalEventHandlers.ontouchstart`
Is an `EventHandler` representing the code to be called when the `touchstart` event is raised.

`GlobalEventHandlers.ontransitioncancel`
An `EventHandler` called when a `transitioncancel` event is sent, indicating that a CSS transition has been cancelled.

`GlobalEventHandlers.ontransitionend`
An `EventHandler` called when a `transitionend` event is sent, indicating that a CSS transitionhas finished playing.

`GlobalEventHandlers.ontransitionrun`
An `EventHandler` called when a `transitionrun` event is sent, indicating that a CSS transitionis running, though not nessarilty started.

`GlobalEventHandlers.ontransitionstart`
An `EventHandler` called when a `transitionstart` event is sent, indicating that a CSS transition has started transitioning.

`GlobalEventHandlers.onwaiting`
Is an `EventHandler` representing the code to be called when the `waiting` event is raised.

Deprecated properties

`Document.alinkColor`
Returns or sets the color of active links in the document body.

`Document.all`
Provides access to all elements in the document — it returns an `HTMLAllCollection` rooted at the document node. This is a legacy, non-standard property and should not be used.

`Document.applets` | Read only
Returns an ordered list of the applets within a document.

`Document.bgColor`
Gets/sets the background color of the current document.

`Document.charset` | Read only
Alias of `Document.characterSet`. Use this property instead.

`Document.domConfig`
Should return a `DOMConfiguration` object.

`Document.fgColor`
Gets/sets the foreground color, or text color, of the current document.

`Document.fullscreen`
`true` when the document is in `full-screen mode`.

`Document.height`
Gets/sets the height of the current document.

`Document.inputEncoding` | Read only
Alias of `Document.characterSet`. Use this property instead.

`Document.linkColor`
Gets/sets the color of hyperlinks in the document.

`Document.rootElement`
Like `Document.documentElement`, but only for `<svg>` root elements. Use this property instead.

`Document.vlinkColor`
Gets/sets the color of visited hyperlinks.

`Document.width`
Returns the width of the current document.

`Document.xmlEncoding`
Returns the encoding as determined by the XML declaration.

`Document.xmlStandalone` | Obsolete since Gecko 10
Returns `true` if the XML declaration specifies the document to be standalone (*e.g.,* An external part of the DTD affects the document's content), else `false`.

`Document.xmlVersion` | Obsolete since Gecko 10
Returns the version number as specified in the XML declaration or `"1.0"` if the declaration is absent.

Methods

This interface also inherits from the `Node` and `EventTarget` interfaces.

`Document.adoptNode()`

Adopt node from an external document.

`Document.captureEvents()`
See `Window.captureEvents`.

`Document.caretRangeFromPoint()`
Gets a `Range` object for the document fragment under the specified coordinates.

`Document.createAttribute()`
Creates a new `Attr` object and returns it.

`Document.createAttributeNS()`
Creates a new attribute node in a given namespace and returns it.

`Document.createCDATASection()`
Creates a new CDATA node and returns it.

`Document.createComment()`
Creates a new comment node and returns it.

`Document.createDocumentFragment()`
Creates a new document fragment.

`Document.createElement()`
Creates a new element with the given tag name.

`Document.createElementNS()`
Creates a new element with the given tag name and namespace URI.

`Document.createEntityReference()`
Creates a new entity reference object and returns it.

`Document.createEvent()`
Creates an event object.

`Document.createNodeIterator()`
Creates a `NodeIterator` object.

`Document.createProcessingInstruction()`
Creates a new `ProcessingInstruction` object.

`Document.createRange()`
Creates a `Range` object.

`Document.createTextNode()`
Creates a text node.

`Document.createTouch()`
Creates a `Touch` object.

`Document.createTouchList()`
Creates a `TouchList` object.

`Document.createTreeWalker()`
Creates a `TreeWalker` object.

`Document.enableStyleSheetsForSet()`
Enables the style sheets for the specified style sheet set.

`Document.exitPointerLock()`
Release the pointer lock.

`Document.getAnimations()`
Returns an array of all `Animation` objects currently in effect, whose target elements are descendants of the `document`.

`Document.getElementsByClassName()`
Returns a list of elements with the given class name.

`Document.getElementsByTagName()`
Returns a list of elements with the given tag name.

`Document.getElementsByTagNameNS()`
Returns a list of elements with the given tag name and namespace.

`Document.hasStorageAccess()`
Returns a `Promise` that resolves with a boolean value indicating whether the document has access to its first-party storage.

`Document.importNode()`
Returns a clone of a node from an external document.

`Document.normalizeDocument()`
Replaces entities, normalizes text nodes, etc.

`Document.releaseCapture()`
Releases the current mouse capture if it's on an element in this document.

`Document.releaseEvents()`
See `Window.releaseEvents()`.

`Document.requestStorageAccess()`
Returns a `Promise` that resolves if the access to first-party storage was granted, and rejects if access was denied.

`Document.routeEvent()` | Obsolete since Gecko 24

See `Window.routeEvent()`.

`Document.mozSetImageElement()`
Allows you to change the element being used as the background image for a specified element ID.

The `Document` interface is extended with the `ParentNode` interface:

`document.getElementById(String id)`
Returns an object reference to the identified element.

`Document.querySelector()`
Returns the first Element node within the document, in document order, that matches the specified selectors.

`Document.querySelectorAll()`
Returns a list of all the Element nodes within the document that match the specified selectors.

The `Document` interface is extended with the `XPathEvaluator` interface:

`Document.createExpression()`
Compiles an `XPathExpression` which can then be used for (repeated) evaluations.

`Document.createNSResolver()`
Creates an `XPathNSResolver` object.

`Document.evaluate()`
Evaluates an XPath expression.

Extension for HTML documents

The `Document` interface for HTML documents inherit from the `HTMLDocument` interface or, since HTML5, is extended for such documents:

`Document.clear()`
In majority of modern browsers, including recent versions of Firefox and Internet Explorer, this method does nothing.

`Document.close()`
Closes a document stream for writing.

`Document.execCommand()`
On an editable document, executes a formating command.

`Document.getElementsByName()`
Returns a list of elements with the given name.

`Document.hasFocus()`
Returns `true` if the focus is currently located anywhere inside the specified document.

`Document.open()`
Opens a document stream for writing.

`Document.queryCommandEnabled()`
Returns true if the formating command can be executed on the current range.

`Document.queryCommandIndeterm()`
Returns true if the formating command is in an indeterminate state on the current range.

`Document.queryCommandState()`
Returns true if the formating command has been executed on the current range.

`Document.queryCommandSupported()`
Returns true if the formating command is supported on the current range.

`Document.queryCommandValue()`
Returns the current value of the current range for a formating command.

`Document.write()`
Writes text in a document.

`Document.writeln()`
Writes a line of text in a document.

Methods included from DocumentOrShadowRoot

The `Document` interface includes the following methods defined on the `DocumentOrShadowRoot` mixin. Note that this is currently only implemented by Chrome; other browsers still implement them on the `Document` interface.

`DocumentOrShadowRoot.getSelection()`
Returns a `Selection` object representing the range of text selected by the user, or the current position of the caret.

`DocumentOrShadowRoot.elementFromPoint()`
Returns the topmost element at the specified coordinates.

`DocumentOrShadowRoot.elementsFromPoint()`
Returns an array of all elements at the specified coordinates.

`DocumentOrShadowRoot.caretPositionFromPoint()`
Returns a `CaretPosition` object containing the DOM node containing the caret, and caret's character offset within that node.

Events

Listen to these events using `addEventListener()` or by assigning an event listener to the `on`*`eventname`* property of this interface.

`scroll`
Fired when the document view or an element has been scrolled.
Also available via the `onscroll` property.

`visibilitychange`
Fired when the content of a tab has become visible or has been hidden.
Also available via the `onvisibilitychange` property.

`wheel`
Fired when the user rotates a wheel button on a pointing device (typically a mouse).
Also available via the `onwheel` property.

Animation events

`animationcancel`
Fired when an animation unexpectedly aborts.
Also available via the `onanimationcancel` property.

`animationend`
Fired when an animation has completed normally.
Also available via the `onanimationend` property.

`animationiteration`
Fired when an animation iteration has completed.
Also available via the `onanimationiteration` property.

`animationstart`
Fired when an animation starts.
Also available via the `onanimationstart` property.

Clipboard events

copy
Fired when the user initiates a copy action through the browser's user interface.
Also available via the `oncopy` property.

cut
Fired when the user initiates a cut action through the browser's user interface.
Also available via the `oncut` property.

paste
Fired when the user initiates a paste action through the browser's user interface.
Also available via the `onpaste` property.

Drag & drop events

drag
Fired every few hundred milliseconds as an element or text selection is being dragged by the user.
Also available via the `ondrag` property.

dragend
Fired when a drag operation is being ended (by releasing a mouse button or hitting the escape key).
Also available via the `ondragend` property.

dragenter
Fired when a dragged element or text selection enters a valid drop target.
Also available via the `ondragenter` property.

dragexit
Fired when an element is no longer the drag operation's immediate selection target.
Also available via the `ondragexit` property.

dragleave
Fired when a dragged element or text selection leaves a valid drop target.
Also available via the `ondragleave` property.

dragover
Fired when an element or text selection is being dragged over a valid drop target (every few hundred milliseconds).
Also available via the `ondragover` property.

dragstart
Fired when the user starts dragging an element or text selection.
Also available via the `ondragstart` property.

drop

Fired when an element or text selection is dropped on a valid drop target.
Also available via the `ondrop` property.

Fullscreen events

`fullscreenchange`
Fired when the `Document` transitions into or out of full-screen mode.
Also available via the `onfullscreenchange` property.

`fullscreenerror`
Fired if an error occurs while attempting to switch into or out of full-screen mode.
Also available via the `onfullscreenerror` property.

Keyboard events

`keydown`
Fired when a key is pressed.
Also available via the `onkeydown` property.

`keypress`
Fired when a key that produces a character value is pressed down.
Also available via the `onkeypress` property.

`keyup`
Fired when a key is released.
Also available via the `onkeyup` property.

Load & unload events

`DOMContentLoaded`
Fired when the document has been completely loaded and parsed, without waiting for stylesheets, images, and subframes to finish loading.

`readystatechange`
Fired when the `readyState` attribute of a document has changed.
Also available via the `onreadystatechange` property.

Pointer events

`gotpointercapture`
Fired when when an element captures a pointer using `setPointerCapture()`.
Also available via the `ongotpointercapture` property.

`lostpointercapture`
Fired when a captured pointer is released.
Also available via the `onlostpointercapture` property.

`pointercancel`
Fired when a pointer event is canceled.
Also available via the `onpointercancel` property.

`pointerdown`
Fired when a pointer becomes active.
Also available via the `onpointerdown` property.

`pointerenter`
Fired when a pointer is moved into the hit test boundaries of an element or one of its descendants.
Also available via the `onpointerenter` property.

`pointerleave`
Fired when a pointer is moved out of the hit test boundaries of an element.
Also available via the `onpointerleave` property.

`pointerlockchange`
Fired when the pointer is locked/unlocked.
Also available via the `onpointerlockchange` property.

`pointerlockerror`
Fired when locking the pointer failed.
Also available via the `onpointerlockerror` property.

`pointermove`
Fired when a pointer changes coordinates.
Also available via the `onpointermove` property.

`pointerout`
Fired when a pointer is moved out of the *hit test* boundaries of an element (among other reasons).
Also available via the `onpointerout` property.

`pointerover`
Fired when a pointer is moved into an element's hit test boundaries.
Also available via the `onpointerover` property.

`pointerup`
Fired when a pointer is no longer active.
Also available via the `onpointerup` property.

Selection events

`selectionchange`
Fired when the current text selection on a document is changed.
Also available via the `onselectionchange` property.

`selectstart`
Fired when the user begins a new selection.
Also available via the `onselectstart` property.

Touch events

`touchcancel`
Fired when one or more touch points have been disrupted in an implementation-specific manner (for example, too many touch points are created).
Also available via the `ontouchcancel` property.

`touchend`
Fired when one or more touch points are removed from the touch surface.
Also available via the `ontouchend` property

`touchmove`
Fired when one or more touch points are moved along the touch surface.
Also available via the `ontouchmove` property

`touchstart`
Fired when one or more touch points are placed on the touch surface.
Also available via the `ontouchstart` property

Transition events

`transitioncancel`
Fired when a CSS transition is canceled.
Also available via the `ontransitioncancel` property.

`transitionend`
Fired when a CSS transition has completed.
Also available via the `ontransitionend` property.

`transitionrun`
Fired when a CSS transition is first created.
Also available via the `ontransitionrun` property.

`transitionstart`

Fired when a CSS transition has actually started.
Also available via the `ontransitionstart` property.

Non-standard extensions

> **Non-standard**
> This feature is non-standard and is not on a standards track. Do not use it on production sites facing the Web: it will not work for every user. There may also be large incompatibilities between implementations and the behavior may change in the future.

Firefox notes

Mozilla defines a set of non-standard properties made only for XUL content:

`Document.currentScript`
Returns the `<script>` element that is currently executing.

`Document.documentURIObject`
(**Mozilla add-ons only!**) Returns the `nsIURI` object representing the URI of the document. This property only has special meaning in privileged JavaScript code (with UniversalXPConnect privileges).

`Document.popupNode`
Returns the node upon which a popup was invoked.

`Document.tooltipNode`
Returns the node which is the target of the current tooltip.

Mozilla also define some non-standard methods:

`Document.execCommandShowHelp()` | Obsolete since Gecko 14
This method never did anything and always threw an exception, so it was removed in Gecko 14.0 (Firefox 14.0 / Thunderbird 14.0 / SeaMonkey 2.11).

`Document.getBoxObjectFor()`
Use the `Element.getBoundingClientRect()` method instead.

`Document.loadOverlay()` | Obsolete since Gecko 61
Loads a XUL overlay dynamically. This only works in XUL documents.

`Document.queryCommandText()` | Obsolete since Gecko 14

This method never did anything but throw an exception, and was removed in Gecko 14 (Firefox 14 / Thunderbird 14 / SeaMonkey 2.11).

Internet Explorer notes

Microsoft defines some non-standard properties:

`Document.fileSize`*

Returns size in bytes of the document. Starting with Internet Explorer 11, that property is no longer supported. See MSDN.

Internet Explorer does not support all methods from the `Node` interface in the `Document` interface:

`Document.contains`

As a work-around, `document.body.contains()` can be used.