Window: beforeunload event

The **`beforeunload`** event is fired when the window, the document and its resources are about to be unloaded. The document is still visible and the event is still cancelable at this point.

This event enables a web page to trigger a confirmation dialog asking the user if they really want to leave the page. If the user confirms, the browser navigates to the new page, otherwise it cancels the navigation.

According to the specification, to show the confirmation dialog an event handler should call `preventDefault()` on the event.

To combat unwanted pop-ups, browsers may not display prompts created in `beforeunload` event handlers unless the page has been interacted with, or may even not display them at all.

The HTML specification states that calls to `window.alert()`, `window.confirm()`, and `window.prompt()` methods may be ignored during this event. See the HTML specification for more details.

Syntax

Use the event name in methods like `addEventListener()`, or set an event handler property.

```
addEventListener('beforeunload', event => { });
onbeforeunload = event => { };
```

Event type

A generic `Event`.

Event handler aliases

In addition to the `Window` interface, the event handler property `onbeforeunload` is also available on the following targets:

- `HTMLBodyElement`
- `HTMLFrameSetElement`
- `SVGSVGElement`

Usage notes

The `beforeunload` event suffers from the same problems as the `unload` event.

Especially on mobile, the `beforeunload` event is not reliably fired. For example, the `beforeunload` event is not fired at all in the following scenario:

1. A mobile user visits your page.
2. The user then switches to a different app.
3. Later, the user closes the browser from the app manager.

The `beforeunload` event is not compatible with the back/forward cache (bfcache), because many pages using this event assume that the page will not continue to exist after the event is fired. To combat this, browsers will not place pages in the bfcache if they have `beforeunload` listeners, and this is bad for performance.

However, unlike the `unload` event, there is a legitimate use case for the `beforeunload` event: the scenario where the user has entered unsaved data that will be lost if the page is unloaded.

It is recommended that developers listen for `beforeunload` only in this scenario, and only when they actually have unsaved changes, so as to minimize the effect on performance. See the Examples section below for an example of this.

See the Page Lifecycle API guide for more information about the problems associated with the `beforeunload` event.

Examples

In this example a page listens for changes to a text `input`. If the element contains a value, it adds a listener for `beforeunload`. If the element is empty, it removes the listener:

```
const beforeUnloadListener = (event) => {
  event.preventDefault();
  return event.returnValue = "Are you sure you want to exit?";
};

const nameInput = document.querySelector("#name");

nameInput.addEventListener("input", (event) => {
  if (event.target.value !== "") {
    addEventListener("beforeunload", beforeUnloadListener, {capture: true});
  } else {
    removeEventListener("beforeunload", beforeUnloadListener, {capture: true});
  }
});
```

Specifications

Specification
HTML Standard # event-beforeunload
HTML Standard # handler-window-onbeforeunload

Browser compatibility

Report problems with this compatibility data on GitHub

	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	WebView Android	Chrome Android	Firefox for Android	
`beforeunload` event	Chrome1	Edge12	Firefox1	Internet 4 Explorer	Opera12	Safari3	WebView 1 Android	Chrome 18 Android	Firefox 4 for Android	O Ar
Activation using `event.returnValue = "string";`	Chrome30	Edge12	Firefox6	Internet 9 Explorer	Opera17	Safari8	WebView 4.4 Android	Chrome 30 Android	Firefox 6 for Android	O Ar
Activation using `event.preventDefault()`	ChromeNo	Edge12–79	Firefox6	Internet 9 Explorer	OperaNo	Safari11	WebView No Android	ChromeNo Android	Firefox 6 for Android	O Ar

	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	WebView Android	Chrome Android	Firefox for Android	
Activation using `return "string";`	Chrome1	Edge12	Firefox1	Internet Explorer 9	Opera12	Safari3	WebView Android 1	Chrome Android 18	Firefox for Android 4	O Ar

Full support No support Deprecated. Not for use in new websites.

Compatibility notes

The HTML specification states that authors should use the `Event.preventDefault()` method instead of using `Event.returnValue` to prompt the user. However, this is not yet supported by all browsers.

When this event returns (or sets the `returnValue` property to) a value other than `null` or `undefined`, the user will be prompted to confirm the page unload. In older browsers, the return value of the event is displayed in this dialog. Starting with Firefox 44, Chrome 51, Opera 38, and Safari 9.1, a generic string not under the control of the webpage will be shown instead of the returned string. For example:

- Firefox displays the string, "This page is asking you to confirm that you want to leave - data you have entered may not be saved." (see bug 588292).
- Chrome displays the string, "Do you want to leave this site? Changes you made may not be saved." (see Chrome Platform Status).

Internet Explorer does not respect the `null` return value and will display this to users as "null" text. You have to use `undefined` to skip the prompt.

In some browsers, calls to `window.alert()`, `window.confirm()`, and `window.prompt()` may be ignored during this event. See the HTML specification for more details.

Note also, that various browsers ignore the result of the event and do not ask the user for confirmation at all. In such cases, the document will always be unloaded automatically. Firefox has a switch named `dom.disable_beforeunload` in *about:config* to enable this behavior. As of Chrome 60, the confirmation will be skipped if the user has not performed a gesture in the frame or page since it was loaded. Pressing F5 in the page seems to count as user interaction, whereas mouse-clicking the refresh arrow or pressing F5 with Chrome DevTools focused does not count as user interaction (as of Chrome 81).

See also

- Related events: `DOMContentLoaded`, `readystatechange`, `load`, `unload`
- Unloading Documents — Prompt to unload a document
- Remove Custom Messages in onbeforeload Dialogs after Chrome 51
- Don't lose user and app state, use Page Visibility explains in detail why you should use `visibilitychange`, not `beforeunload`/`unload`.
- Page Lifecycle API gives best-practices guidance on handling page lifecycle behavior in your web applications.
- PageLifecycle.js : a JavaScript library that deals with cross-browser inconsistencies in page lifecycle behavior.
- Back/forward cache explains what the back/forward cache is, and its implications for various page lifecycle events.

Last modified: Apr 27, 2022, by MDN contributors