Window: unload event

- Syntax
- Event type
- Event handler aliases
- Usage notes
- Examples
- Specifications
- Browser compatibility
- See also

> **WARNING:** Developers should avoid using this event. See "Usage notes" below.

The `unload` event is fired when the document or a child resource is being unloaded.

It is fired after:

- beforeunload (cancelable event)
- pagehide

The document is in the following state:

- All the resources still exist (img, iframe etc.)
- Nothing is visible anymore to the end user
- UI interactions are ineffective (window.open, alert, confirm, etc.)
- An error won't stop the unloading workflow

Please note that the unload event also follows the document tree: parent frame unload will happen **before** child frame `unload` (see example below).

Syntax

Use the event name in addEventListener():

```
addEventListener("unload", (event) => { })
```

Alternatively, assign to the event handler property of the interface:

```
onunload = (event) => { }
```

Event type

A generic Event.

Event handler aliases

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

- HTMLBodyElement
- HTMLFrameSetElement
- SVGSVGElement

Usage notes

Developers should avoid using this event.

Especially on mobile, the `unload` event is not reliably fired. For example, the `unload` 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.

Also, the `unload` 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, some browsers (such as Firefox) will not place pages in the bfcache if they have unload listeners, and this is bad for performance.

For these reasons, Chrome has stopped firing `unload` events by default. A page that still depends on `unload` can opt back in using the `unload` directive of the Permissions-Policy header.

Instead of `unload`, use the following events, both of which are compatible with the bfcache:

- The `visibilitychange` event is fired when Document.visibilityState changes from `visible` to `hidden`, or vice versa. This is the last event that is reliably fired, so it is the best place to save application state or send analytics data. Note that `visibilitychange` is fired when the user switches to another tab, leaves the current page, or closes it.
- The `pagehide` event is fired when the browser hides the current page in the process of presenting a different page from the session's history. This is useful if you're specifically trying to detect the user navigating away from the page. However, like `unload`, it is not reliably fired, especially on mobile, so prefer `visibilitychange` where possible.

For example, instead of sending data in an `unload` listener:

```
window.addEventListener("unload", () => {
  navigator.sendBeacon("/log", analyticsData);
});
```

Send it when the page becomes hidden:

```
document.addEventListener("visibilitychange", () => {
  if (document.visibilityState === "hidden") {
    navigator.sendBeacon("/log", analyticsData);
  }
});
```

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

Examples

```
<!doctype html>
<html lang="en-US">
  <head>
    <meta charset="UTF-8" />
    <title>Parent Frame</title>
    <script>
      window.addEventListener("beforeunload", (event) => {
```

```
        console.log("I am the 1st one.");
      });
      window.addEventListener("unload", (event) => {
        console.log("I am the 3rd one.");
      });
    </script>
  </head>
  <body>
    <iframe src="child-frame.html"></iframe>
  </body>
</html>
```

Below, the content of `child-frame.html`:

```
<!doctype html>
<html lang="en-US">
  <head>
    <meta charset="UTF-8" />
    <title>Child Frame</title>
    <script>
      window.addEventListener("beforeunload", (event) => {
        console.log("I am the 2nd one.");
      });
      window.addEventListener("unload", (event) => {
        console.log("I am the 4th and last one…");
      });
    </script>
  </head>
  <body>
    ☻
  </body>
</html>
```

When the parent frame is unloaded, events will be fired in the order described by the `console.log()` messages.

Specifications

Browser compatibility

See also

- Related events: DOMContentLoaded, readystatechange, load
- Unloading Documents — unload a document
- The `visibilitychange` event.
- 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.

□□□ #3
□ □□□ □□□ 2026-10-31 03:27:53 UTC
□ cocobook-sync □□□ 2026-11-04 17:35:24 UTC