JavaScript unload Event

Summary: in this tutorial, you will learn how to use the JavaScript `unload` event that fires when the document is fully unloaded.

> Note that you should never use the unload event unless you have a good reason to do so.

Introduction to the JavaScript unload event

The `unload` event fires when a document has been completely unloaded. Typically, the `unload` event fires when you navigate from one page to another.

The unload event is fired after:

- beforeunload event
- pagehide event

At this moment, the HTML document is in the following state:

- UI is not visible to the users and is not effective.
- All the resources like images, iframe, etc., still exist.
- An error won't stop the unloading flow.

In practice, you should never use the `unload` event because it is not reliable on mobile devices and causes an issue with bfcache.

Handling the JavaScript unload event

To handle the `unload` event, you can use the `addEventListener()` method:

```
addEventListener('unload', (event) => {
    console.log('The page is unloaded');
});
```

Alternatively, you can assign an event handler to the `onunload` property of the `window` object:

```
window.onunload = (event) => {
    console.log('The page is unloaded');
};
```

Or you assign an event handler to the `onunload` attribute of the `<body>` element:

```
<!DOCTYPE html>
<html>
<head>
    <title>JS unload Event Demo</title>
</head>
<body onunload="console.log('The page is unloaded')">

</body>
</html>
```

It's a good practice to use the `addEventListener()` to register the unload event handler.

Summary

- JavaScript fires the `unload` event when a document has been completely unloaded.
- In practice, do not use the `unload` event unless you have a good reason to do so.