

Going Global: Global Events

In this lesson, let's learn how to handle global events.
Let's begin!


WE'LL COVER THE FOLLOWING



- Listing 6-14: Simple event handler
- Windows event attributes
- Keyboard event attributes
- Mouse event attributes
- Form event attributes
- Media event attributes

You already used a few events in exercises, such as the `onload` event when you set the `onload` attribute of the `<body>` tag, or the `onclick` event when you set the `onclick` attribute of a `<button>` element.

HTML5 defines dozens of events that are triggered either as a result of user interactions, or as a result of state changes in the document's lifecycle. You can attach event handler scripts to HTML elements. These handlers are fired when the specific event is triggered.

 **NOTE:** You can easily distinguish event attributes from other ones, as their name always begins with the `on` prefix, such as `onclick`, `onload`, `onblur`, etc.

The scripts you set as the value of an event attribute can be any valid JavaScript, a sequence of statements as well as a single call.

Listing 6-14 shows an example where you assign event handler code to two buttons using their `onclick` event attribute.

Listing 6-14: Simple event handler

```
<!DOCTYPE html>
<html>
<head>
  <title>Simple event handler</title>
</head>
<body>
  <button onclick="this.textContent='Clicked.'">
    Click me!
  </button>
  <br />
  <button onclick="nowMe()">
    Now, click me!
  </button>
  <script>
    function nowMe() {
      document
        .getElementsByName('button')[0]
        .setAttribute('hidden', true);
    }
  </script>
</body>
</html>
```

The first button's `onclick` attribute uses a single inline statement that sets the `textContent` property of the button to `"Clicked."`, while the second button's `onclick` attribute simply invokes the `nowMe()` method.

 Show Useful Info

There are events that are triggered by the window object which represents the browser window the current page is displayed in. You can apply the attributes representing these events to the `<body>` tag only.

The table below summarizes these events:

Windows event attributes

Event	It fires (when)...
onafterprint	After the document is printed
onbeforeprint	Before the document is printed
onbeforeunload	Before the document is unloaded (when you close the browser, navigate to another page, etc.)
onerror	An error occurs
onhaschange	The document has changed
onload	The page is finished loading
onmessage	The message is triggered
onoffline	The document goes offline (the internet connection gets broken)
ononline	The document goes online (the internet connection gets restored)
onpagehide	The window that contains the document gets hidden
onpageshow	The window that contains the document becomes visible
onpopstate	The window's history changes
onredo	The document performs a redo
onresize	The browser window is resized
onstorage	A web storage area is updated
onundo	The document performs an undo
onunload	The page has unloaded, or the browser window has been closed

As the table below shows, you can catch keyboard events. These events can be attached to any element, except `<style>`, `<script>`, `<title>`, `<head>`, `<param>`, `<meta>`, `
`, `<base>`, and `<html>`.

Keyboard event attributes

Event	It fires (when)...
onkeydown	A user is pressing a key
onkeypress	A user presses a key
onkeyup	A user releases a key

These events follow this order: `onkeydown`, `onkeypress`, `onkeyup`.

Elements that can accept keyboard events, can accept user interactions

triggered by a mouse, or similar pointing devices including a pen or touch. The table below shows these event attributes.

Mouse event attributes

Event	It fires (when)...
onclick	On a mouse click on the element
ondblclick	On a mouse double-click on the element
ondrag	An element is dragged
ondragend	At the end of a drag operation
ondragenter	An element has been dragged to a valid drop target
ondragleave	An element leaves a valid drop target
ondragover	An element is being dragged over a valid drop target
ondragstart	At the start of a drag operation
ondrop	The dragged element is being dropped
onmousedown	A mouse button is pressed down on an element
onmousemove	The mouse pointer moves over an element
onmouseout	The mouse pointer moves out of an element
onmouseover	The mouse pointer moves over an element
onmouseup	A mouse button is released over an element
onmousewheel	The mouse wheel is rotated
onscroll	An element's scrollbar is being scrolled

There are events that can be used within HTML forms, the table below summarizes them. Although they apply to almost all HTML elements, generally they are used in forms.

Form event attributes

Event	It fires (when)...
onblur	The moment the element loses the focus
onchange	The moment when the value of an element is changed
oncontextmenu	A context menu is triggered. As of this writing, no major browser supports this event attribute.
onfocus	The element gets focus
onformchange	A form changes
onforminput	A form gets user input
oninput	An element gets user input
oninvalid	An element is invalid
onselect	After some text has been selected in an element

onselect	After some text has been selected in an element
onsubmit	A form is submitted

As you learned in [Chapter 4](#) (Using Multimedia), HTML5 introduces new tags to handle audio and video. It is not surprising that you can use media events as summarized in the table below.

Most of these events can be used on almost all HTML elements, but they are most common in media elements like `<audio>`, `<embed>`, ``, `<object>`, and `<video>`.

Media event attributes

Event	It fires (when)...
onabort	Loading of media (such as an image or a video) has been aborted
oncanplay	A file is ready to start playing (when it has buffered enough to begin)
oncanplaythrough	A file can be played all the way to the end without pausing for buffering
ondurationchange	The length of the media changes
onemptied	Something bad happens and the file is suddenly unavailable (like unexpectedly disconnects)
onended	The media has reach the end
onerror	An error occurs while the media file or stream is being loaded
onloadeddata	Media data is loaded
onloadedmetadata	Meta data (like dimensions and duration) are loaded
onloadstart	Just as the file begins to load before anything is actually loaded
onpause	The media is paused either by the user or programmatically
onplay	The media is ready to start playing
onplaying	The media actually has started playing
onprogress	The browser is in the process of getting the media data
onratechange	Each time the playback rate changes (like when a user switches to a slow motion or fast forward mode)
onreadystatechange	Each time the ready state changes (the ready state tracks the state of the media data)
onseeked	The seeking attribute is set to false indicating that seeking has ended
onseeking	The seeking attribute is set to true indicating that seeking is active
onstalled	The browser is unable to fetch the media data for whatever reason
onsuspend	Fetching the media data is stopped before it is completely loaded for whatever reason

Now that we have covered all global events and attributes with sufficient

detail, let's move onto using event parameters in HTML in the *next lesson*.