DOM Events

Document Object Model events HTML-5.com is a great guide for web developers. TV Series & Actors and Actresses. Follow TV Series and HTML 5 on Google+.

HTML-5.com ➢ itemscopehttp://data-vocabulary.org/BreadcrumbHTML 5 ➢ itemscopehttp://data-vocabulary.org/BreadcrumbJavaScript Code ➢ itemscopehttp://data-vocabulary.org/BreadcrumbJavaScript DOM ➢ itemscopehttp://data-vocabulary.org/Breadcrumb

DOM Events

DOM Events

Various user actions can trigger an <dfn>event</dfn>. The <dfn>event target</dfn> is the innermost element at the relevant location for the event, such as the location of the cursor when a mouse button is clicked. For example, if the event occurs where text content is displayed, the target element is the parent element of the text node. The event is propagated through various elements in the DOM hierarchy initially from top to bottom, down to the target element of the event, and then back to the top:

1. The event propagates down from the Window to the Document then down through the ancestors of the target element to the parent of the target element. This is called the <dfn>capture phase</dfn>. As the event propagates through each node, any event listeners added with the useCapture addEventListener parameter set to `true` will be dispatched in the order that they were registered.
2. Propagation of the event reaches the target node. This is called the <dfn>at target phase</dfn>. The target's event handler will be dispatched, if any, followed by any registered event listeners in the order they were registered.
3. The event propagates back up from the parent of the target element to the Document and then the Window. This is called the <dfn>bubbling phase</dfn>. Some events may not bubble up. As the event propagates through each node, the node's event handler will be dispatched, if any, followed by any event listeners registered with the useCapture addEventListener parameter set to `false`.

Window

# #

Document

# #

<html>
HTMLHtmlElement

# #

<body>
HTMLBodyElement

. #

. .

. .

.

etc.
HTMLElement

. #

. .

. .

.

target's parent
HTMLElement

# #

target
HTMLElement

Events are dispatched to event handlers and event listeners to the registered event listeners and HTML elements based on the DOM element hierarchy at the time the event is initially dispatched. The event flow does not change even if listeners are registered or removed or if the DOM element hierarchy is changed by some of the event handling code.

back to top

Event Handlers

Note that <dfn>event handlers</dfn>, which are created by global `on...` event attributes, are *not* dispatched during the capture phase. They are only dispatched during the at target phase and bubbling phase.

onerror Event Handlers

When an **error** event is propagated to a node with an `onerror` event handler, the JavaScript code in the onerror attribute is invoked. In the onerror event handler code, an `arguments` array provides access to values from the event constructor and the `event` variable provides access to other details of the event and a boolean return value:

Parameters:

arguments[0] - the message being reported

arguments[1] - absolute URL of the resource (document) in which the error occurred

arguments[2] - in that resource, the line number where the error occurred

Return value:

`event.returnValue = false` if the error has been handled or
`event.returnValue = true` if not

onmouseover Event Handlers

When a **mouseover** event is propagated to a node with an `onmouseover` event handler, the JavaScript code in the onmouseover attribute is invoked. In the onmouseover event handler code, the `event` variable provides access to details of the event and a boolean return value:

Parameters:

event - the event

Return value:

`event.returnValue = true` if the event is to be cancelled or
`event.returnValue = false` if not

Note that the return value for mouseover event handlers is the opposite from the return value for other event handlers (below).

onbeforeunload Event Handlers

When a beforeUnload event is propagated to a node with an `onbeforeunload` event handler, the JavaScript code in the onbeforeunload attribute is invoked. In the onbeforeunload event handler code, the `event` variable provides access to details of the event and a return value that contains a dialog prompt string:

Parameters:

event - the event

Return value:

`event.returnValue = "Prompt text"` to display a prompt for the user to confirm leaving the page or `event.returnValue = ""` if not

Other Event Handlers

When any other event is propagated to a node with an event handler, the JavaScript code in the event handler attribute for the event type is invoked. In the event handler code, the `event` variable provides access to details of the event and a boolean return value:

Parameters:

event - the event

Return value:

`event.returnValue = false` if the event is to be cancelled or
`event.returnValue = true` if not

`event.returnValue=false` cancels almost any event:

Clicking on this link does *not* load the href page because the event is cancelled with `onclick="event.returnValue=false"`.

```
  <p>Clicking on <a href="http://www.ExampleOnly.com/" title="Just a tool tip"
     onclick="alert('clicked'); event.returnValue=false">this link</a> does
➥ <em>not</em> load the
     href page because the event is cancelled with
➥ <code>onclick="event.returnValue=false"</code>.
  </p>
```

`event.preventDefault()` is another way to cancel an event:

Clicking on this link does *not* load the href page because the event is cancelled with `onclick="event.preventDefault()"`.

```
  <p>Clicking on <a href="http://www.ExampleOnly.com/" title="Just a tool tip"
```

```
    onclick="alert('clicked'); event.preventDefault()">this link</a> does
➥ <em>not</em> load the
    href page because the event is cancelled with
➥ <code>onclick="event.preventDefault()"</code>.
  </p>
```

Using `event.preventDefault()` is the recommended method because it is more browser independent. There are some HTML 5 web browsers, such as Firefox 3, where the `event.returnValue=false` method does not cancel the event and the browser loads the page referenced by the <a href> attribute after the `alert` box is dismissed.

Without stopPropagation(), events on the inner element propagate to the outer element, changing both to a lighter color at the same time:

Try it:

```
  <div style="width: 90px; height: 90px; margin: auto; border: 1px solid black;
➥ padding: 0; background-color: blue"
    onmouseover="this.style.backgroundColor = 'lightblue'"
    onmouseout="this.style.backgroundColor = 'blue'"
  >
  <p style="width: 40px; height: 40px; margin: 0; border: 1px solid black;
➥ padding: 0; background-color: green"
    onmouseover="this.style.backgroundColor = 'lightgreen'"
    onmouseout="this.style.backgroundColor = 'green'"
  > </p>
  </div>
```

With stopPropagation(), events on the inner element do not affect the outer element so only one changes to a lighter color at a time:

Try it:

```
  <div style="width: 90px; height: 90px; margin: auto; border: 1px solid black;
➥ padding: 0; background-color: blue"
    onmouseover="this.style.backgroundColor = 'lightblue';
➥ event.stopPropagation()"
    onmouseout="this.style.backgroundColor = 'blue'; event.stopPropagation()"
  >
  <p style="width: 40px; height: 40px; margin: 0; border: 1px solid black;
➥ padding: 0; background-color: green"
    onmouseover="this.style.backgroundColor = 'lightgreen';
➥ event.stopPropagation()"
    onmouseout="this.style.backgroundColor = 'green'; event.stopPropagation()"
  > </p>
  </div>
```

back to top

Event Listeners

error Event Listeners

When an **error** event is propagated to a node, any registered event listener functions of type **error** are invoked. The error event listener function takes three parameters and returns a boolean value:

***fn*(DOMString message, DOMString url, DOMString lineNumber)**

The function to be dispatched when an error event is propagated to the node that the event listener is attached to.

Parameters:

message - the message being reported

url - absolute URL of the resource (document) in which the error occurred

lineNumber - in that resource, the line number where the error occurred

Return value:

`event.returnValue = false` if the error has been handled by this function or
`event.returnValue = true` if not

mouseover Event Listeners

When a **mouseover** event is propagated to a node, any registered event listener functions of type **mouseover** are invoked. The mouseover event listener function takes one parameter and returns a boolean value:

***fn*(Event event)**

The function to be dispatched when a mouseover event is propagated to the node that the event listener is attached to.

Parameters:

event - the event

Return value:

`event.returnValue = true` if the event is to be cancelled or
`event.returnValue = false` if not

Note that the return value for mouseover event listeners is the opposite from the return value for other event listeners (below).

beforeunload Event Listeners

When a **beforeunload** event is propagated to a node, any registered event listener functions of type **beforeunload** are invoked. the beforeunload event listener takes one parameter and returns a string value:

***fn*(Event event)**

The function to be dispatched when a beforeunload event is propagated to the node that the event listener is attached to.

Parameters:

event - the event

Return value:

`event.returnValue = "Prompt text"` to display a prompt for the user to confirm leaving the page or `event.returnValue = ""` if not

Other Event Listeners

When any other event is propagated to a node, any registered event listener functions of the matching type are invoked. The event listener function takes one parameter and returns a boolean value:

***fn*(Event event)**

The function to be dispatched when an event is propagated to the node that the event listener is attached to.

Parameters:

event - the event

Return value:

`event.returnValue = false` if the event is to be cancelled or
`event.returnValue = true` if not

back to top

THE END

