XHTML™ Events

An updated events syntax for XHTML

W3C Working Draft 28 August 2000

This version:
    http://www.w3.org/TR/2000/WD-xhtml-events-20000828
Latest version:
    http://www.w3.org/TR/xhtml-events
Previous version:
    http://www.w3.org/TR/1999/WD-xhtml-events-19991221
Editors:
    Ted Wugofski, Phone.com (previously with Gateway)

Abstract

This specification defines the XHTML Event Module, a module that provides XHTML host languages [XMOD] with the ability to uniformly integrate behaviors with Document Object Model (DOM) Level 2 event interfaces [DOM2].

This specification also defines the XHTML Basic Event Module, a module which subsets the XHTML Event Module for simpler applications and simpler client devices, and the XHTML Event Types Module, a module defining XHTML language event types.

The DOM specifies an event model that provides the following features:

- the event system is generic,
- a means is provided for registering event handlers,
- events may be routed through a tree structure, and
- context information for each event is available.

In addition, the DOM provides an event flow architecture that describes how events are captured, bubbled, and canceled. In summary, event flow is the

process through which an event originates from the DOM implementation and is passed into the document object model. The methods of event capture and event bubbling, along with various event listener registration techniques, allow the event to then be handled in a number of ways. It can be handled locally at the target node level or centrally from a node higher in the document tree.

The XHTML Event Module contains an `onevent` element is used to represent the DOM event listener. As with the DOM Level 2 event interfaces, the XHTML Event Module provides a means for authors to listen to events during the capturing and bubbling phases, as well as when an event reaches its target node.

Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the W3C.

This is the second public working draft of the of the XHTML Event Module specification. It is guaranteed to change; anyone implementing it should realize that we will not allow ourselves to be restricted by experimental implementations when deciding whether to change the specifications.

This specification is a Working Draft of the HTML Working Group for review by W3C members and other interested parties. Publication as a Working Draft does not imply endorsement by the W3C membership, nor of members of the HTML, SYMM, nor DOM working groups.

This document may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use W3C Working Drafts as reference material or to cite them as other than "work in progress".

This document has been produced as part of the W3C HTML Activity.

This document is for public review. Comments on the normative aspects of this document or the integration with XHTML should be sent to the public mailing list www-html@w3.org.

A list of current W3C Recommendations and other technical documents can be found at http://www.w3.org/TR.

Special acknowledgements to: Mark Baker (Sun Microsystems), Wayne Carr (Intel Corporation), Warner ten Kate (Philips Electronics), Shane P. McCarron (Applied Testing and Technology), Patrick Schmitz (Microsoft), and Peter Stark (Ericsson) for their significant contributions to the evolution of this specification.

Contents

1. The XHTML Basic Events Module

This section is normative.

This working draft proposes two event processing modules (in addition to the event types): an XHTML Basic Events Module and an XHTML Events Module. The XHTML Basic Events Module provides simple DOM2 event model support for simple applications and simple devices. The XHTML Events Module provides full DOM Level 2 event model support.

These two modules differ in that the XHTML Basic Events Module does not include all of the `onevent` element's attributes:

Attribute	XHTML Basic Events	XHTML Events
`id`	Yes	Yes
`type`	Yes	Yes
`eventsource`	No	Yes
`registerwith`	No	Yes
`onphase`	Yes	Yes
`capture`	Yes	Yes

The XHTML Basic Events Module does not support the `eventsource` and the `registerwith` attributes in order to simplify the binding of event listeners to target nodes.

Binding is simplified because event listeners, as embodied by the `onevent` element, only appear as child elements of the target node. Therefore, there is no reason to resolve bindings with elements that have not yet been loaded into the document tree.

2. Event Module Elements

This section is normative.

The `onevent` element supports a subset of the DOM's `eventlistener` interface.

The requirements for the XHTML Event Module are defined as:

- Expose the DOM event model to an XML document [XML]
- Provide for new event types without requiring modification to the DOM or the DTD.
- Don't require non-XML parsers.
- Be able to integrate with other XML languages

2.1 The onevent Element

```
<!ENTITY % Boolean "(true | false )" >
<!ENTITY % onevent-content "((action,stopevent?)|(script,stopevent?)|sto
<!ELEMENT onevent %onevent-content;>
<!ATTLIST onevent
   id               ID          #IMPLIED
   type             NMTOKEN     #REQUIRED
   eventsource      IDREF       #REQUIRED
   registerwith     IDREF       #IMPLIED
   onphase          (capturing|bubbling|target) #IMPLIED
   capture          %Boolean;   #IMPLIED
>
```

The `onevent` element provides a means for a document to declaratively specify an event listener that is registered to either the `onevent` element's parent element or the element refered to by the `onevent` element's `registerwith` attribute. The `onevent` element is said to be registered to the *`onevent` element's target node*.

Following the DOM2 convention, when multiple, identical event listeners register to the same target node, only the last registration will be kept and the previous will be discarded. No specification is made to the order in which multiple non-identical event listeners are processed.

As specified in DOM2, an event of a given type is dispatched from the top of the document tree and propagates in a direct line between the top of the document tree and the event's target node. When the event reaches its target node, the event propagates back to the top of the document tree in a direct line.

Therefore, an `onevent` element may "see" an event in three conditions:

1. When the event is propagating down the document tree and the event propagates through the `onevent` element's target node on its way to the event's target node. This condition occurs during the event's capturing phase.
2. When the event's target node is the same as the `onevent` element's target

node and the event reaches its target node.
3. When the event is propagating back up the document tree and the event propagates through the `onevent` element's target node on its way from the event's target node up to the document's root node. This condition occurs during the event's bubbling phase.

If the `onevent` element's `onphase` attribute is "bubbling" or "target", it will not see the event during the capturing phase.

If the `onevent` element's `onphase` attribute is "capturing" or "target", it will not see the event during the bubbling phase.

The semantics of the `onevent` element are such that when the `onevent` element sees the event, the host language bindings to the `onevent` element should process the event.

Note that even though an `onevent` element may register with a target node as a child of the target node or using the `onevent` element's `registerwith` attribute, these two mechanisms do not result in the same document tree structure. This means that the user agent must rely on the DOM interfaces for managing events, rather than simply relying on the natural flow of events through the document tree. If a user only relied on the natural flow of events through the document tree, the `onevent` element would never see an event targeting the `onevent` element's target node.

Certain event types may restrict whether the event participates in all phases of propagation.

In XHTML, the `onevent` element may contain an `action` element or a `script` element. The `action` element specifies a link to an event handler external to the `onevent` element. The `script` element contains a fragment of script that serves as the event handler.

It is not required that a language using the XHTML Event Module support the script child element if that language does not support the XHTML Scripting Module [XMOD].

The `onevent` element has the following attributes:

id
: The `id` attribute is a document-unique identifier. The value of this identifier is often used to manipulate the element through a DOM interface.

type
: The `type` attribute specifies the event type for which the content author is registering. As specified by [DOM2], the value of the type attribute should be an XML Name [XML]. The `onevent` element's *desired event* is an event whose event type matches the value of `type` attribute.

eventsource
: The `eventsource` attribute specifies the source of an event (i.e., which node in the document dispatched the event into the document tree). If the

eventsource attribute is specified, only events that match both the type attribute value and the eventsource value are considered.

registerwith

The onevent element registers an event listener with the element specified by the registerwith attribute. The default value for the registerwith attribute is "empty", which means that the onevent element registers an event listener with the onevent element’s parent. Otherwise, registerwith is the value of desired target element’s id attribute.

onphase

The onphase attribute specifies when (during which DOM 2 event propagation phase) the onevent element may see the desired event. If the value of the onphase attribute is "oncapture", the onevent element will only see an event during the capturing phase. If the value of the onphase attribute is "bubbling", the onevent element will only see an event during the bubbling phase. If the value of the onphase attribute is "target", the onevent element will only see an event when the event reaches the event’s target node (and this is the same node as the onevent element’s target node). The default behavior is for the onevent element to see events when they reach the target node (i.e., target).

2.2 The action Element

```
<!ENTITY % action-content EMPTY>
<!ELEMENT action %action-content;>
<!ATTLIST action
   id        ID             #IMPLIED
   href      %URI           #REQUIRED
     type      %ContentType   #IMPLIED
>
```

The action element provides a generic means of binding an event handler to an event listener. The action element is always a child element of an onevent element. When the onevent element sees a desired event, it will invoke the behaviour linked to by the enclosed action element.

The language integrating the Event Module will determine what content types may serve as appropriate behaviours.

The action element has the following attributes:

id

The id attribute is a document-unique identifier. The value of this identifier is often used to manipulate the element through a DOM interface.

href

The href attribute is a link to the associated behavior. This link may be a to an internal or external behavior. If the link is to an external behavior, it is processed as if the behavior was imported into the current document at the location of the action element.

type

The type attribute is a hint of the content type at the other end of the link specified by the href attribute. If the type attribute is not specified, the default content type is the same as the default scripting content type.

2.3 The stopevent Element

```
<!ENTITY % stopevent-content EMPTY>
<!ELEMENT stopevent %stopevent-content;>
<!ATTLIST stopevent
   id       ID             #IMPLIED
>
```

The `stopevent` element provides a means of stopping the propagation of an event after it has been handled. The `stopevent` element is equivalent to the `stopPropagation` method in DOM2.

The `stopevent` element is useful if a descendent event handler wishes to stop an event from bubbling back to an ancestral event handler (i.e., the descendent effectively overrides the ancestor’s behaviour).

id

The `id` attribute is a document-unique identifier. The value of this identifier is often used to manipulate the element through a DOM interface.

2.4 The XHTML Event Module Namespace

The XHTML Event Module will use the "http://www.w3.org/1999/xhtml" namespace as specified in XML Namespaces [NAME].

3. Naming Event Types

This section is informative

While, the XHTML Event Module does not normatively specify how a language designer should name their events (i.e., the values stored in the `onevent` element’s `type` attribute). However, this specification does make a recommendation on how these events should be named.

It is recommended that event types be a string containing a prefix followed with a hyphen character ("-") followed with the name of the event:

```
event_prefix-event_type_name
```

The event’s prefix is a lightweight mechanism for qualifying event types.

To avoid confusion between event type names and the syntax for qualified names, -according to XML namespaces [NAME]-, that sometimes are used in attribute values, it is recommended that the event type names not include the colon (’:’).

4. XHTML Event Types Module

This section is informative

The XHTML Event Types Module defines the following event names. Refer to [DOM2] for the semantics of these HTML events.

XHTML Event Name	DOM2 Event Name
`html-load`	`load`
`html-unload`	`unload`
`html-abort`	`abort`
`html-error`	`error`
`html-select`	`select`
`html-change`	`change`
`html-submit`	`submit`
`html-reset`	`reset`
`html-focus`	`focus`
`html-blur`	`blur`
`html-resize`	`resize`
`html-scroll`	`scroll`
`dom-click`	`click`
`html-dblclick`	not specified
`dom-mousedown`	`click`
`dom-mouseup`	`click`
`dom-mouseover`	`click`
`dom-mousemove`	`click`
`dom-mouseout`	`click`
`html-keypress`	not specified
`html-keydown`	not specified
`html-keyup`	not-specified

5. Using the XHTML Event Module in XHTML

This section is informative.

The XHTML Event Module may be integrated into XHTML to add extensibility to the event handling already present through a variety of properties. This section is informative: it is provided as a way of explaining how the Event Module may be used with XHTML.

This section describes how to use the XHTML Event Module in XHTML. This section does not formalize how the XHTML Event Module is integrated into the XHTML DTD or schema.

5.1 Registering an Event Listener

The `onevent` element provides a means of registering an event listener to an element, and in the process, associating an event hander with that listener. For example, if you wanted to associate an event listener to an `img` element, you would write:

```
<img id="b" ...>
  <onevent id="a" type="dom-click">
    ... the desired event handler ...
  </onevent>
</img>
```

In this example, an `onevent` element is registered with the `img` element by making the `onevent` element a child of the `img` element. A second approach would have been to use the `onevent` element's `registerwith` attribute:

```
<onevent id="a" registerwith="b" type="dom-click">
   ... the desired event handler ...
</onevent>
...
<img id="b" .../>
...
```

In this example, the `onevent` element is registered to the `img` element by setting the `onevent` element's `registerwith` attribute to "b".

In both examples, the "desired event handler" will be processed when, and only when, an "dom-click" event targets the `img` element. This is because the default behavior is for the `onevent` element to only see a desired event when it reaches the target node (i.e., the same behavior that would occur had the `onevent` element's `onphase` attribute been set to "target").

5.2. Listening to a Bubbling Event

If you want to see an event after it is processed by an element, you should make the `onevent` element a child of an ancestor to the event's target node. This is quite useful if you want to invoke the same behavior for an event that could target several child elements. For example:

```
<div id="a">
  <onevent id="b" type="dom-click" onphase="bubbling">
    ... the desired event handler ...
  </onevent>
    ...
  <img id="foo" ... />
    <div id="c">
      <img id="bar" ... />
    </div>
</div>
```

In this example, the `onevent` element is implicitly registered with the `div` element because the `onevent` element is a child of the `div` element. The `onevent` element's `onphase` attribute is set to "bubbling" so that it will only see events during the bubbling phase of event propagation.

The two `img` elements are also descendents of the "a" `div` element. Therefore, when a `dom-click` event targets the "foo" image, the event propagates from "a" to "foo" and then back to "a" (before working its way back to the top of the document). Likewise, when a `dom-click` event targets the "bar" image, the event propagates from "a" to "c" to "bar" and then back to "a".

5.3. Overriding Event Handlers

Sometimes you will want to have a default behaviour in which you occassionally override. Consider the previous example in which you had two images and a default handler:

```
<div id="a">
  <onevent id="b" type="dom-click" onphase="bubbling">
    ... the desired event handler ...
  </onevent>
    ...
  <img id="foo" ... />
  <div id="c">
    <img id="bar" ... />
  </div>
</div>
```

If you wanted to add a third image you might write:

```
<div id="a">
  <onevent id="b" type="dom-click" onphase="bubbling">
    ... the desired event handler ...
  </onevent>
    ...
  <img id="foo" ... />
  <div id="c">
    <img id="bar" ... />
  </div>
  <img id="new" ... />
</div>
```

As written, the "b" `onevent` element will handle `dom-click` events that target any of the three images. If you wanted to have a different behavior for the "new" `img` element, you could write:

```
<div id="a">
  <onevent id="b" type="dom-click" onphase="bubbling">
    ... the desired event handler ...
  </onevent>
    ...
  <img id="foo" ... />
  <div id="c">
    <img id="bar" ... />
  </div>
  <img id="new" ... >
    <onevent id="c" type="dom-click">
    ... the desired event handler ...
      <stopevent/>
    </onevent>
  </img>
```

```
</div>
```

In this example, the "c" `onevent` element will see the `dom-click` event when the event targets the "new" `img` element. After the `onevent` element’s behaviour is invoked, the event `stopevent` element causes the `dom-click` event to stop propagating through the document.

A. References

A.1 Normative References

[XML]
"Extensible Markup Language (XML) 1.0". W3C Recommendation. See http://www.w3.org/TR/1998/REC-xml-19980210

[NAME]
"Namespaces in XML", . Bray T., et.al. W3C Recommendation. See http://www.w3.org/TR/1999/REC-xml-names-19990114

A.2 Other References

[DOM2]
"Document Object Model (DOM) Level 2 Specification", Wood L., et.al. See http://www.w3.org/TR/2000/CR-DOM-Level-2-20000510 (This document is a work in progress).

[XHTML]
"XHTML™ 1.0: The Extensible HyperText Markup Language". Pemberton S., et.al. W3C Recommendation. See http://www.w3.org/TR/2000/REC-xhtml1-20000216

[XMOD]
"Modularization of XHTML™". Altheim M., et.al. See http://www.w3.org/TR/xhtml-modularization (This document is a work in progress).