XHTML Events

An updated events syntax for XHTML™ and friends

W3C Working Draft 08 June 2001

This version:
http://www.w3.org/TR/2001/WD-xhtml-events-20010608
(PostScript version, PDF version, ZIP archive, or Gzip'd TAR archive)
Latest version:
http://www.w3.org/TR/xhtml-events
Previous version:
http://www.w3.org/TR/2000/WD-xhtml-events-20000828
Diff-marked version:
xhtml-events-diff.html
Editors:
Shane McCarron, Applied Testing and Technology, Inc.
T. V. Raman, IBM

Abstract

The XHTML Events module defined in this specification provides XHTML host languages [XHTMLMOD] [p.23] the ability to uniformly integrate event listeners and associated event handlers with Document Object Model (DOM) Level 2 event interfaces [DOM2] [p.23] . The result is to provide XHTML based languages an interoperable way of associating behaviors with document-level markup.

In addition, this specification defines a subset of the XHTML Events module called *basic events* for use on simpler client devices. Finally, the XHTML Event Types Module defines the XHTML language event types.

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 third 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.

Note that we expect to change the name of this document from XHTML Events to a more inclusive name at the next iteration.

This specification is a Working Draft of the HTML Working Group for review by W3C members and other interested parties. It has been updated from its previous version. A diff-marked version is available. Publication as a Working Draft does not imply endorsement by the W3C membership, nor of members of the HTML, XForms, 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.

Contents

1. Introduction

This section is normative.

This working draft proposes two event processing modules: an XHTML Events Module and a Basic XHTML Events Module. The XHTML Events Module provides full DOM Level 2 event model support. The Basic XHTML Events Module subsets this module to provide DOM2 event model support for simple applications and simple devices. In addition this specification defines some standard event types.

The design requirements for the XHTML Events modules were the following:

- Expose the DOM event model to an XML document [XML] [p.23]
- Provide for new event types without requiring modification to the DOM or the DTD.
- Be able to integrate with other XML languages

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

- A generic event system,
- Means for registering event listeners and handlers,
- Means for routing events through a tree structure, and
- Access to context information for each event.

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 by which an event originates from the DOM implementation and is passed into the document object model. The methods for event registration, event capture and event bubbling allow events to be handled in a number of ways. Events can be handled locally at the *target node* (i.e. the document node where the event was received) or at a node higher in the document tree; the latter permits event handling to be *centralized*.

Element `onevent` defined in the XHTML Events module represents a DOM level 2 event listener. Element `onevent` and its associated markup encapsulate the various aspects of the the DOM level 2 event interface, thereby providing markup level access for specifying the actions to be taken during the various phases of event propagation.

2. XHTML Events Modules

This section is normative.

This specification defines two XHTML Events modules - XHTML Events, and Basic XHTML Events. The XHTML Events modules use the XML Namespaces [NAME] [p.23] identifier: `xmlns="http://www.w3.org/1999/xhtml"`. *Note that we expect to change this namespace to something that is XHTML Events specific.* The remainder of this section describes the elements and attributes in these modules, their semantics, and provides an abstract module definition for each as required in [XHTMLMOD] [p.23] .

2.1. XHTML Events Module

The XHTML Events Module supports the following elements and attributes:

Elements	Attributes	Minimal Content Model
onevent	eventsource* (IDREF), id (ID), onphase ("capturing" | "bubbling" | "target"*), registerwith (IDREF), type* (NMTOKEN)	((action, stopevent?) | (do, stopevent?) | (script, stopevent?) | stopevent)
action	href* (URI), id (ID), type (ContentType)	EMPTY
do	id (ID)	ANY
stopevent	id (ID)	EMPTY

Note that the `script` element in the minimal content model is only required with the XHTML Document Scripting Module is also selected (see [XHTMLMOD [p.23]]).

Implementation: DTD [p.21]

2.1.1. The onevent Element

Element `onevent` [p.7] supports a subset of the DOM's `eventlistener` interface.

Element `onevent` is used to declare event listeners and register them with specific nodes in the DOM. The node that an `onevent` event listener registered with is called its *observer*. By default, the *observer* is the parent node of the `onevent` element. Optional attribute `registerwith` of element `onevent` can be used to register the event listener with a node other than its immediate parent. When multiple event listeners register for the same event with the same target node, only the last registration is retained; (this follows the same convention as the [DOM2 [p.23]]). The order in which multiple non-identical event listeners are processed is unspecified.

As specified in [DOM2 [p.23]], an event is dispatched from the top of the document tree and propagates along a direct line to the event's target node. When the event reaches its target node, the event propagates back to the top of the document tree along this line.

Therefore, an event listener declared by `<onevent id="e1" type="event_1">` may *receives* event `event_1` under three possible conditions:

1. As event `event_1` propagates through the *observer* of node `e1` on its way from the document root to its target node. This condition occurs during the event's capturing phase.
2. When event `event_1` reaches its target node.
3. As event `event_1` propagates through the *observer* `e1` on its way back to the document's root node. This condition occurs during the event's bubbling phase.

Attribute `onphase` of element `onevent` can be used to refine this behavior. When attribute `onphase` is set to `bubbling` or `target`, the event listener will not receive the event during the capture phase. When attribute `onphase` is set to `capturing` or `target`, the event listener will not receive the event during the bubbling phase. The default behavior is for an event listener to *receive* an event only if the *observer* is also the event's target node.

When all conditions for an event listener to *receive* an event have been satisfied, the associated event handler is invoked. The `onevent` element may contain an `action` element or a `script` element that encapsulates the event handler. The `action` element may specify a link to an event handler external to the `onevent` element. The `script` can contain a script that serves as the event handler. Markup Languages using the XHTML Events Module need support the script child element only if that language also supports the XHTML Document Scripting Module [[XHTMLMOD] [p.23]].

Note that the two means for registering an `onevent` with its observer --- implicitly with its parent node or explicitly by setting attribute `registerwith` --- result in different document tree structure in cases where the *observer* is not the immediate parent of element `onevent`. This means that in the general case, the user agent must rely on the DOM interfaces for managing events. Relying on just the tree structure would result in `onevent` elements that register explicitly with a node other than their immediate parentnever receiving events that target the *observer* node.

Element `onevent` has the following attributes:

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

type
: Attribute `type` specifies the event type for which the content author is registering. As specified by [DOM2 [p.23]], the value of the type attribute should be an XML Name [XML [p.23]]. The `onevent` element's *desired event* is an event whose event type matches the value of `type` attribute.

eventsource
: Attribute `eventsource` specifies the source of an event (i.e., the node that dispatched the event). If an event listener specifies a value for attribute `eventsource`, only events that match both the `type` and `eventsource` attributes will be processed by the associated event handler.

registerwith
: Attribute `registerwith` specifies the `id` of the target node with which the event listener is to be registered. The default is to register the event listener *implicitly* with the parent of node `onevent`.

onphase
: The `onphase` attribute specifies when (during which DOM 2 event propagation phase) the `onevent` element receives the desired event.

  `oncapture`
  : Event is only seen during capturing phase.

  `bubbling`
  : Event is only seen during bubbling phase.

  `target`
  : Event is only seen if the target node of the event listener matches the target node of the event.

  The default behavior is `onphase="target"`.

2.1.2. The action Element

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 behavior specified by the contained `action` element.

The `action` element has the following attributes:

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

href
: attribute `href` 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
: Attribute `type` is a hint of the content type at the other end of the link specified by the `href` attribute. If attribute `type` is not specified, the default content type is the same as the default scripting content type. The Markup Language using the XHTML Events module must define the content types that may serve as appropriate behaviors. If a Markup Language does not support the type of an associated behavior, the behavior is implementation defined.

2.1.3. The stopevent Element

Element `stopevent` is used within an `onevent` element to stop further propagation of an event once it has been handled by that event listener. Element `stopevent` is equivalent to method `stopPropagation` in [DOM2 [p.23]]. As an example, a descendant wishing to override an ancestor’s behavior can do so by including an event listener that contains a `stopevent` element --see the example [p.17] that shows how element `stopevent` can be used to override default behavior for a descendant node. Note that `stopevent` cannot be used to stop propagation during the capture phase.

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

2.2. The Basic XHTML Events Module

The Basic XHTML Events Module subsets the XHTML Events module by not including all of the `onevent` [p.7] element's attributes.

Elements	Attributes	Minimal Content Model
onevent	id (ID), onphase ("capturing" | "bubbling" | "target"*), type* (NMTOKEN)	((action, stopevent?) | (do, stopevent?) | (script, stopevent?) | stopevent)
action	href* (URI), id (ID), type (ContentType)	EMPTY
do	id (ID), type (ContentType)	ANY
stopevent	id (ID),	EMPTY

The Basic XHTML Events Module does not support the `eventsource` and the `registerwith` attributes. The purpose of this is to simplify support for the binding of event listeners to its observer node. Binding in this module is easier to support since there is no need to resolve `idref` references to nodes that have not yet been loaded into the document tree.

Note that the `script` element in the minimal content model is only required with the XHTML Document Scripting Module is also selected (see [XHTMLMOD [p.23]]).

Implementation: DTD [p.21]

3. Naming Event Types

This section is informative

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 the name of the event, where the key characters of the event name, including the first character, are in upper-case:

```
event_prefixEventTypeName
```

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] [p.23] -, 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] [p.23] for the semantics of these HTML events.

XHTML Event Name	DOM2 Event Name
htmlLoad	load
htmlUnload	unload
htmlAbort	abort
htmlError	error
htmlSelect	select
htmlChange	change
htmlSubmit	submit
htmlReset	reset
htmlFocus	focus
htmlBlur	blur
htmlResize	resize
htmlScroll	scroll
domClick	click
htmlDblclick	not specified
domMousedown	click
domMouseup	click
domMouseover	click
domMousemove	click
domMouseout	click
htmlKeypress	not specified
htmlKeydown	not specified
htmlKeyup	not-specified

5. Using the XHTML Event Module in XHTML

This section is informative.

The XHTML Events Module may be integrated into XHTML [p.23] 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 [p.23] .

This section describes how to use the XHTML Event Module in XHTML [p.23] . This section does not formalize how the XHTML Event Module is integrated into the XHTML DTD or schema.

5.1. Registering an Event Listener

To register an event listener with an `img` element, one would write:

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

Here, the event listener identified by `id="imageButtonClick"` is registered with the image element identified by `id="imageButton"`. Here, registration is *implicit* since element `onevent` is made a child of element `img`. *Note that this requies that the default content model for the XHTML img element be changed from* `EMPTY` *to* `(onevent?)`. Alternatively, the registration could be *explicit* by setting attribute `registerwith` on element `onevent`.

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

Notice that using attribute `registerwith` allows the event listener to appear *outside* the node with which it is being registered.

In both examples, the *event handler* will be processed **if and only if** an `dom-click` event targets `img` element `id="imageButton"`. This is because the default behavior is for the `onevent` element to only receive events that target its *observer*, i.e., the same behavior as `onphase='target'`.

5.2. Listening to a Bubbling Event

To see an event after it is processed by its target node, the document can register an event listener with an ancestor of the target node. This is quite useful when invoking the same behavior for an event e.g., to enable centralized processing of specific events across multiple child nodes. For example:

```
<div id="outer">
<onevent id="defaultListener" type="dom-click" onphase="bubbling">
... the desired event handler ...
</onevent>
...
<img id="innerImage1" ... />
<div id="innerDiv">
<img id="innerImage2" ... />
</div>
</div>
```

Here, event listener `defaultListener` is registered with its parent using *implicit* registration. Its `onphase` attribute is set to `bubbling` so that it only sees events during the bubbling phase of event propagation. The two `img` elements are also descendants of the `outer div` element. Therefore, when a `dom-click` event targets the `innerImage1` image, the event propagates from node `outer` to image `innerImage1` during the capture phase, and then flows back through node`outer` during the bubbling phase. Likewise, when a `dom-click` event targets image `innerImage2`, the event propagates from node `outer` through node `innerDiv` to `innerImage2` before bubbling back through node `outer`.

5.3. Overriding Event Handlers

Sometimes, there is a need to have a default behavior that is occasionally overridden. Consider previous example where event listener `defaultListener` provided *default behavior* for the two image elements by registering itself with their common ancestor during the bubbling phase. A third image might be added by writing:

```
<div id="outer">
<onevent id="defaultListener" type="dom-click" onphase="bubbling">
... the desired event handler ...
</onevent>
...
<img id="innerImage1" ... />
<div id="innerDiv">
<img id="innerImage2" ... />
</div>
<img id="special" ... />
</div>
```

As written, the `defaultListener` event listener will handle `dom-click` events that target any of the three images. To override this default behavior for image `special` one could write:

```
<div id="outer">
<onevent id="defaultListener" type="dom-click" onphase="bubbling">
... the desired event handler ...
</onevent>
...
<img id="innerImage1" ... />
<div id="innerDiv">
<img id="innerImage2" ... />
</div>
<img id="special" ... >
<onevent id="override" type="dom-click">
... the desired event handler ...
<stopevent/>
</onevent>
</img>
</div>
```

In this example event listener `override` will receive `dom-click` events targeting image `special`. After the `onevent` element's behavior is invoked, the event `stopevent` element causes the `dom-click` event to stop propagating up the document, thereby preventing the default behavior from being invoked.

A. DTD Implementation

This appendix is *normative*.

The DTD implementation of XHTML Events conforms to the requirements defined in [XHTMLMOD] [p.23] . Consequently, it provides a Qualified Names sub-module, and individual module files for each of the XHTML Events modules defined in this recommendation.

A.1. Qualified Names Module

```
<!-- ...................................................................... -->
<!-- XHTML Events Qname Module  ............................................ -->
<!-- file: xhtml-events-qname-1.mod

     This is XHTML Events - the Events Module for XHTML and Friends,
     a redefinition of access to the DOM events model.

     Copyright 2000-2001 W3C (MIT, INRIA, Keio), All Rights Reserved.

     This DTD module is identified by the PUBLIC and SYSTEM identifiers:

       PUBLIC "-//W3C//ENTITIES XHTML Events Qnames 1.0//EN"
       SYSTEM "http://www.w3.org/TR/xhtml-events/DTD/xhtml-events-qname-1.mod"

     Revisions:
     (none)
     ....................................................................... -->

<!-- Xevengts Qname (Qualified Name) Module

     This module is contained in two parts, labeled Section 'A' and 'B':

       Section A declares parameter entities to support namespace-
       qualified names, namespace declarations, and name prefixing
       for XHTML Events and extensions.

       Section B declares parameter entities used to provide
       namespace-qualified names for all XHTML Events element types:

         %action.qname;   the xmlns-qualified name for <action>
         %onevent.qname;     the xmlns-qualified name for <onevent>
         ...

     XHTML Events extensions would create a module similar to this one.
     Included in the XHTML distribution is a template module
     ('template-qname-1.mod') suitable for this purpose.
-->

<!-- Section A: XHTML Events XML Namespace Framework :::::::::::::::::::: -->

<!-- 1. Declare a %XHTML-EVENTS.prefixed; conditional section keyword, used
        to activate namespace prefixing. The default value should
        inherit '%NS.prefixed;' from the DTD driver, so that unless
        overridden, the default behaviour follows the overall DTD
```

```
        prefixing scheme.
-->
<!ENTITY % NS.prefixed "IGNORE" >
<!ENTITY % XHTML-EVENTS.prefixed "%NS.prefixed;" >

<!-- 2. Declare a parameter entity (eg., %XHTML.xmlns;) containing
        the URI reference used to identify the XHTML namespace:
        We hope that this namespace will change to something unique to XHTML
        in the near future.
-->
<!ENTITY % XHTML-EVENTS.xmlns  "http://www.w3.org/1999/xhtml" >

<!-- 3. Declare parameter entities (eg., %XHTML.prefix;) containing
        the default namespace prefix string(s) to use when prefixing
        is enabled. This may be overridden in the DTD driver or the
        internal subset of an document instance. If no default prefix
        is desired, this may be declared as an empty string.

     NOTE: As specified in [XMLNAMES], the namespace prefix serves
     as a proxy for the URI reference, and is not in itself significant.
-->
<!ENTITY % XHTML-EVENTS.prefix  "" >

<!-- 4. Declare parameter entities (eg., %XHTML.pfx;) containing the
        colonized prefix(es) (eg., '%XHTML.prefix;:') used when
        prefixing is active, an empty string when it is not.
-->
<![%XHTML-EVENTS.prefixed;[
<!ENTITY % XHTML-EVENTS.pfx  "%XHTML-EVENTS.prefix;:" >
]]>
<!ENTITY % XHTML-EVENTS.pfx  "" >

<!-- declare qualified name extensions here ........... -->
<!ENTITY % xhtml-qname-extra.mod "" >
%xhtml-qname-extra.mod;

<!-- 5. The parameter entity %XHTML.xmlns.extra.attrib; may be
        redeclared to contain any non-XHTML namespace declaration
        attributes for namespaces embedded in XHTML. The default
        is an empty string.  XLink should be included here if used
        in the DTD.
-->
<!ENTITY % XHTML-EVENTS.xmlns.extra.attrib "" >


<!-- Section B: XHTML Qualified Names ::::::::::::::::::::::::::::: -->

<!-- 6. This section declares parameter entities used to provide
        namespace-qualified names for all XHTML element types.
-->

<!-- module:  xhtml-events-basic-1.mod -->

<!ENTITY % xhtml-events.onevent.qname  "%XHTML-EVENTS.pfx;onevent" >
<!ENTITY % xhtml-events.action.qname  "%XHTML-EVENTS.pfx;action" >
```

```
<!ENTITY % xhtml-events.do.qname  "%XHTML-EVENTS.pfx;do" >
<!ENTITY % xhtml-events.stopevent.qname  "%XHTML-EVENTS.pfx;stopevent" >

<!-- end of xhtml-events-qname-1.mod -->
```

A.2. XHTML Events Module

```
<!-- ...................................................................... -->
<!-- XHTML Events Module .................................................. -->
<!-- file: xhtml-events-1.mod

     This is XHTML Events - the Events Module for XHTML and Friends,
     a redefinition of access to the DOM events model.

     Copyright 2000-2001 W3C (MIT, INRIA, Keio), All Rights Reserved.

     This DTD module is identified by the PUBLIC and SYSTEM identifiers:

       PUBLIC "-//W3C//ENTITIES XHTML Events 1.0//EN"
       SYSTEM "http://www.w3.org/TR/xhtml-events/DTD/xhtml-events-1.mod"

     Revisions:
     (none)
     ....................................................................... -->


<!-- XHTML Events-basic defines the essential components of this module -->

<!ENTITY % xhtml-events-basic.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Events Basic 1.0//EN"
            "http://www.w3.org/TR/xhtml-events/DTD/xhtml-events-basic-1.mod" >
%xhtml-events-basic.mod;

<!-- Extend the onevent element with additional attributes -->

<!ATTLIST %xhtml-events.onevent.qname;
    eventsource      IDREF        #REQUIRED
    registerwith     IDREF        #IMPLIED
>

<!-- end of xhtml-events-1.mod -->
```

A.3. Basic XHTML Events Module

```
<!-- ...................................................................... -->
<!-- Basic XHTML Events Module  ........................................... -->
<!-- file: xhtml-events-basic-1.mod

     This is Basic XHTML Events - the Basic Events Module for XHTML and Friends,
     a redefinition of access to the DOM events model.

     Copyright 2000-2001 W3C (MIT, INRIA, Keio), All Rights Reserved.

     This DTD module is identified by the PUBLIC and SYSTEM identifiers:

       PUBLIC "-//W3C//ENTITIES XHTML Basic XHTML Events 1.0//EN"
       SYSTEM "http://www.w3.org/TR/xhtml-events/DTD/xhtml-events-basic-1.mod"
```

```
     Revisions:
     (none)
     ....................................................................... -->

<!ENTITY % xhtml-events.onevent.content "((%xhtml-events.action.qname;,%xhtml-events.stopevent.qname;?)|
                                    (%xhtml-events.do.qname;,%xhtml-events.stopevent.qname;?)|
                        (script.qname;,%xhtml-events.stopevent.qname;?)|
                     %xhtml-events.stopevent.qname;)" >

<!ELEMENT %xhtml-events.onevent.qname; %xhtml-events.onevent.content;>
<!ATTLIST %xhtml-events.onevent.qname;
    id               ID           #IMPLIED
    onphase          (capturing|bubbling|target) #IMPLIED
    type             NMTOKEN      #REQUIRED
>

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

<!ENTITY % xhtml-events.do.content ANY>
<!ELEMENT action %xhtml-events.do.content;>
<!ATTLIST %xhtml-events.do.qname;
    id       ID             #IMPLIED
>

<!ENTITY % xhtml-events.stopevent.content EMPTY>
<!ELEMENT %xhtml-events.stopevent.qname; %xhtml-events.stopevent.content;>
<!ATTLIST %xhtml-events.stopevent.qname;
    id       ID             #IMPLIED
>

<!-- end of xhtml-xhtml-events-1.mod -->
```

B. References

This appendix is *normative*.

B.1. Normative References

[XML]
"Extensible Markup Language (XML) 1.0". W3C Recommendation. See http://www.w3.org/TR/2000/REC-xml-20001006

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

B.2. Other References

[DOM2]
"Document Object Model (DOM) Level 2 Core Specification", Wood L., et.al. W3C Recommendation. See http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113.

[XHTML]
"XHTML™ 1.0: The Extensible HyperText Markup Language". Pemberton S., et.al. W3C Recommendation. See http://www.w3.org/TR/2000/REC-xhtml1-20000126

[XHTMLMOD]
"Modularization of XHTML™", Altheim M., et.al. W3C Recommendation. See http://www.w3.org/TR/xhtml-modularization

C. Acknowledgments

This section is informative.

This document was originally edited by Ted Wugofski, OpenWave.

Special acknowledgments 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.

At the time of publication, the members of the W3C HTML Working Group were:

List will be inserted when this document becomes a Recommendation.