Abstract

The XML Events module defined in this specification provides XML languages with the ability to uniformly integrate event listeners and associated event handlers with Document Object Model (DOM) Level 2 event interfaces [DOM2EVENTS][p.25]. The result is to provide an interoperable way of associating behaviors with document-level markup.
Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the [W3C technical reports index at http://www.w3.org/TR/].

This document is a Recommendation of the W3C. It has been reviewed by W3C Members and other interested parties, and has been endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited as a normative reference from another document. W3C’s role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web. A test suite for XML Events has been developed as part of a public [XForms 1.0 Test Suite] along with an [implementation report].

This document has been produced by the W3C HTML Working Group [Members only] as part of the HTML Activity. The goals of the HTML Working Group are discussed in the [HTML Working Group charter]. Patent disclosures relevant to this specification can be found on the Working Group’s patent disclosure page.

Please report errors in this specification to www-html-editor@w3.org [archive]. It is inappropriate to send discussion email to this address. Public discussion may take place on www-html@w3.org [archive].

Contents

1. Introduction
2. Conformance Requirements
   2.1. Document Conformance
   2.2. Host Language Conformance
   2.3. User Agent Conformance
3. The XML Events Module
   3.1. The listener Element
      3.1.1. Examples of listener Usage
   3.2. Attaching Attributes Directly to the Observer Element
      3.2.1. Examples of Using Attributes Attached to an Observer Element
   3.3. Attaching Attributes Directly to the Handler Element
      3.3.1. Examples of Using Attributes Attached to a Handler Element
   3.4. Summary of Observer and Handler Attribute Defaulting
   3.5. Event Handlers
   3.6. The Basic XML Events Profile
4. Naming Event Types
   A. DTD Implementation
      A.1. Qualified Names Module
      A.2. XML Events Module
B. Schema Implementation
   B.1. Attributes Module
   B.2. XML Events Module
C. References
   C.1. Normative References
   C.2. Other References
D. Acknowledgments
1. Introduction

This section is informative.

An event is the representation of some asynchronous occurrence (such as a mouse click on the presentation of the element, or an arithmetical error in the value of an attribute of the element, or any of unthinkably many other possibilities) that gets associated with an element (targeted at it) in an XML document.

In the DOM model of events [DOM2EVENTS[p.25]], the general behavior is that when an event occurs it is dispatched by passing it down the document tree in a phase called capture to the element where the event occurred (called its target), where it then may be passed back up the tree again in the phase called bubbling. In general an event can be responded to at any element in the path (an observer) in either phase by causing an action, and/or by stopping the event, and/or by cancelling the default action for the event. The following diagram illustrates this:

Event flow in DOM2: an event targeted at an element (marked 'target') in the tree passes down the tree from the root to the target in the phase called 'capture'. If the event type allows it, the event then travels back up the tree by the same route in a phase called 'bubbling'. Any node in the route, including the root node and the target, may be an 'observer': that is to say, a handler may be attached to it that is activated when the event passes through in either phase. A handler can only listen for one phase. To listen for both you have to attach two handlers.
An action is some way of responding to an event; a handler is some specification for such an action, for instance using scripting or some other method. A listener is a binding of such a handler to an event targeting some element in a document.

HTML [HTML4][p.25] binds events to an element by encoding the event name in an attribute name, such that the value of the attribute is the action for that event at that element. This method has two main disadvantages: firstly it hardwires the events into the language, so that to add a new event, you have to make a change to the language, and secondly it forces you to mix the content of the document with the specifications of the scripting and event handling, rather than allowing you to separate them out. SVG [SVG][p.25] uses a similar method.

The process of defining a new version of HTML identified the need for an extensible event specification method. The design requirements were the following:

- Syntactically expose the DOM event model to an XML document [XML][p.25].
- Provide for new event types without requiring modification to the DOM or the DTD.
- Allow for integration 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,
- Access to context information for each event, and
- A definition of event flow, as sketched above.

Element listener and its attributes defined in this specification are the method of binding a DOM level 2 event at an element to an event handler. They encapsulate various aspects of the DOM level 2 event interface, thereby providing markup-level specification of the actions to be taken during the various phases of event propagation.

This document neither specifies particular events, nor mandates any particular methods of specifying actions. These definitions are left to any markup language using the facilities described here.
2. Conformance Requirements

This section is normative.

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119[p.25]].

2.1. Document Conformance

XML Events is not a stand-alone document type. It is intended to be integrated into other host languages such as XHTML. A conforming XML Events document is a document that requires only the facilities described as mandatory in this specification and the facilities described as mandatory in its host language. Such a document must meet all the following criteria:

1. The document must conform to the constraints expressed in Appendix B - Schema Implementation[p.21] or Appendix A - DTD Implementation[p.17], combined with the constraints expressed in its host language implementation.

2. The document must contain an xmlns declaration for the XML Events namespace [XMLNAMES[p.25]]. The namespace for XML Events is defined to be http://www.w3.org/2001/xml-events. An example start tag of a root element might look like:

   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
       xmlns:ev="http://www.w3.org/2001/xml-events" >

2.2. Host Language Conformance

When XML Events are included in a host language, all of the facilities required in this specification must be included in the host language. In addition, the elements and attributes defined in this specification must be included in the content model of the host language.

2.3. User Agent Conformance

A conforming user agent must support all of the features required in this specification.
3. The XML Events Module

This section is normative.

This specification defines a module called XML Events. The XML Events module uses the XML namespace [XMLNAMES][p.25] identifier http://www.w3.org/2001/xml-events.

Examples in this document that use the namespace prefix "ev" all assume an xmlns declaration xmlns:ev="http://www.w3.org/2001/xml-events" somewhere suitable in the document involved. All examples are informative.

The remainder of this section describes the elements and attributes in this module, the semantics, and provides an abstract module definition as required in [XHTMLMOD][p.26].

The XML Events Module supports the following element and attributes:

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Minimal Content Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>listener</td>
<td>event (NMTOKEN), observer (IDREF), target (IDREF), handler (URI), phase (&quot;capture&quot;</td>
<td>&quot;default&quot;), propagate (&quot;stop&quot;</td>
</tr>
</tbody>
</table>

Implementations: [DTD][p.19], [XML Schema][p.22]

3.1. The listener Element

Element listener supports a subset of the DOM’s EventListener interface. It is used to declare event listeners and register them with specific nodes in the DOM, and has the following attributes:

event
The required event attribute specifies the event type for which the listener is being registered. As specified by [DOM2EVENTS][p.25], the value of the attribute should be an XML Name [XML][p.25].

observer
The optional observer attribute specifies the id of the element with which the event listener is to be registered. If this attribute is not present, the observer is the element that the event attribute is on (see later under "Attaching Attributes Directly to the Observer Element[p.12"]), or the parent of that element (see later under "Attaching Attributes Directly to the Handler Element[p.12"]).
target
The optional target attribute specifies the id of the target element of the event (i.e., the node that caused the event). If this attribute is present, only events that match both the event and target attributes will be processed by the associated event handler. Clearly because of the way events propagate, the target element should be a descendent node of the observer element, or the observer element itself.

Use of this attribute requires care; for instance, if you specify

```xml
<listener event="click" observer="para1"
    target="link1" handler="#clicker"/>
```

where 'para1' is some ancestor of the following node

```xml
<a id="link1" href="doc.html">The <em>draft</em> document</a>
```

and the user happens to click on the word "draft", the <em> element, and not the <a>, will be the target, and so the handler will not be activated; to catch all mouse clicks on the <a> element and its children, use observer="link1", and no target attribute.

handler
The optional handler attribute specifies the URI reference of a resource that defines the action that should be performed if the event reaches the observer. (This specification does not mandate what form that element should take: see further the section [Event Handlers [p.14] ]). If this attribute is not present, the handler is the element that the event attribute is on (see later under [Attaching Attributes Directly to the Handler Element][p.12] ).

phase
The optional phase attribute specifies when (during which DOM 2 event propagation phase) the listener will be activated by the desired event.

```xml
capture
  Listener is activated during capturing phase.
```

```xml
default
  Listener is activated during bubbling or target phase.
```

The default behavior is phase="default".

Note that not all events bubble, in which case with phase="default" you can only handle the event by making the event's target the observer.

propagate
The optional propagate attribute specifies whether after processing all listeners at the current node, the event is allowed to continue on its path (either in the capture or the bubble phase).

```xml
stop
  event propagation stops
```

```xml
continue
  event propagation continues (unless stopped by other means, such as scripting, or by another listener).
```
The default behavior is propagate="continue".

defaultAction
The optional defaultAction attribute specifies whether after processing of all listeners for the event, the default action for the event (if any) should be performed or not. For instance, in XHTML the default action for a mouse click on an `<a>` element or one of its descendents is to traverse the link.

cancel
if the event type is cancelable, the default action is cancelled

perform
the default action is performed (unless cancelled by other means, such as scripting, or by another listener).

The default value is defaultAction="perform".

Note that not all events are cancelable, in which case this attribute is ignored.

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

Note that observer = "<element-id>" and event = "<event-type>" are similar to the begin = "<element-id>.<event-type>" attribute in SMIL EventTiming [SMIL20][p.25].

3.1.1. Examples of listener Usage

1. This example attaches the handler in the element at "#doit" that will get activated when the event called activate occurs on the element with id="button1", or any of its children. The activation will occur during bubbling, or if the event happened on the observer element itself, when the event reaches the element (phase target).

   `<listener event="activate" observer="button1" handler="#doit"/>

2. This attaches the handler at #overflow-handler that will get activated when the event overflow occurs on the element with id="expr1" and bubbles up to the element with id="progl".

   `<listener event="overflow" observer="progl" target="expr1" handler="#overflow-handler"/>

3. This attaches the handler at #popup that will get activated whenever an activate event occurs at the element with id="embargo" or any of its children. Since it will be activated during the capture phase, and propagation is stopped, this will have the effect (regardless of what the handler does) of preventing any child elements of the embargo element seeing any activate events.
3.2. Attaching Attributes Directly to the Observer Element

All the attributes from the `<listener>` element with the exception of `id` may be used as global attributes, as defined in *Namespaces in XML* [XMLNAMES][p. 25], to attach the attributes to other elements.

Note that this means that the `<listener>` element is strictly speaking redundant, since the following

```
<anylelement ev:event="click" ev:observer="button1" ev:handler="#clicker"/>
```

would have the same effect as

```
<ev:listener ev:event="click" ev:observer="button1" ev:handler="#clicker"/>
```

Nonetheless, for utility the `<listener>` element has been retained.

If the `observer` attribute is omitted (but not the `handler` attribute), then the element that the other attributes are attached to is the observer element.

### 3.2.1. Examples of Using Attributes Attached to an Observer Element

1. This first example will attach the handler identified by "#popper" to the `<a>` element, and cancel the default action for the event.

```
<a href="doc.html" ev:event="activate" ev:handler="#popper"
    ev:defaultAction="cancel">The document</a>
```

2. This will attach the handler at `#handle-overflow` for the event `overflow` to the current element.

```
<div ev:event="overflow" ev:handler="#handle-overflow"> ... </div>
```

3.3. Attaching Attributes Directly to the Handler Element

If, when attaching the global attributes to an element, the `handler` attribute is omitted then the element that the other attributes are attached to is the handler element.

Note that, since the `observer` and `target` attributes are IDREFs, in this case the handler and observer/target elements must be in the same document (while in other cases, since the `handler` attribute is a URI, the handler element may be in another document).
If the observer attribute is also omitted, then the parent of the handler element is the observer element.

### 3.3.1. Examples of Using Attributes Attached to a Handler Element

1. In this case the element is the handler for the submit event on the element with id="form1".

```xml
<script type="application/x-javascript"
    ev:event="submit" ev:observer="form1">
    return docheck(event);
</script>
```

2. In this case the <action> element is the handler for event q-submit, and the observer is the questionnaire element.

```xml
<questionnaire submissionURL="/q/tally">
    <action ev:event="q-submit">
        ...
    </action>
    ...
</questionnaire>
```

3. The <script> element is the handler for event click; the <img> element is the observer.

```xml
<img src="button.gif" alt="OK">
    <script ev:event="activate" type="application/x-javascript">
        doactivate(event);
    </script>
</img>
```

4. The <onevent> element is the handler for event enterforward. The <card> element is the observer.

```xml
<card>
    <onevent ev:event="enterforward">
        <go href="/url"/>
    </onevent>
    <p>
        Hello!
    </p>
</card>
```

5. The <catch> element is the handler for the nomatch event. The observer is the <field> element.

```xml
<form id="launch_missiles">
    <field name="password">
        <prompt>What is the code word?</prompt>
        <grammar>
            <rule id="root" scope="public">rutabaga</rule>
        </grammar>
        <help>It is the name of an obscure vegetable.</help>
        <catch ev:event="nomatch">
```
6. This example shows three handlers for different events. The observer for all three is the `<secret>` element.

```xml
<secret ref="/login/password">
  <caption>Please enter your password</caption>
  <info ev:event="help">
    Mail help@example.com in case of problems
  </info>
  <info ev:event="hint">
    A pet’s name
  </info>
  <info ev:event="alert">
    This field is required
  </info>
</secret>
```

### 3.4. Summary of Observer and Handler Attribute Defaulting

The following table summarizes which elements play the role of observer or handler if the relevant attribute is omitted.

<table>
<thead>
<tr>
<th></th>
<th>Handler present</th>
<th>Handler omitted</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Observer present</strong></td>
<td>(As declared)</td>
<td>Element is handler</td>
</tr>
<tr>
<td><strong>Observer omitted</strong></td>
<td>Element is observer</td>
<td>Parent is observer</td>
</tr>
</tbody>
</table>

### 3.5. Event Handlers

This specification does not require an XML application that uses XML Events to use any particular method for specifying handlers. However, the examples, particularly those in the section on attaching the attributes directly to the handler, are intended to give examples of how they could be specified.

It is however recognized that two methods are likely to occur often: scripting (such as XHTML’s `<script>` element) and declarative markup using XML elements (such as WML’s `<onevent>` element). A companion specification will provide markup to support these methods.
3.6. The Basic XML Events Profile

The Basic XML Events Profile allows restrictions on the usage of the XML Events Module in order to make processing easier on small devices.

The Basic Profile allows the following restrictions on the use of `listener` element and its attributes, and on the use of the attributes from the `listener` element as global attributes.

1. External Event Handlers

The ability to process external event handlers is not required. When the 'handler' attribute on the `listener` element is used, or when the global 'handler' attribute is used, the handler specified in the value of that attribute should be within the current document.

For example, the following is allowed:

```xml
<listener event="click" target="#button1" handler="#clicker"/>
```

while the following is not required to be processed:

```xml
<listener event="click" target="#button1" handler="doc2.html#clicker"/>
```

2. Ordering of Event Bindings

The binding of an event handler to an observer may be required to be lexically before the end of the observer element. In other words, a `<listener>` binding to an observer may not occur after the closing tag of the observer element, and an event handler carrying the attributes to bind it to an observer may also not occur after the closing tag of the observer element.

4. Naming Event Types

This section is informative.

This specification does not normatively specify how language designers should name events (i.e., the values used in the `event` attribute).

However, future versions of DOM Events are likely to allow namespaced event names, so language designers are advised not to use the colon character "::" in event names.

A number of event types are defined in DOM2 [DOM2EVENTS][p.25], to which you should refer for their names and semantics.
A.DTD Implementation

This appendix is normative.

The DTD implementation of XML Events conforms to the requirements defined in [XHTMLMOD] [p.26]. Consequently, it provides a Qualified Names sub-module, and a module file for the XML Events module defined in this Proposed Recommendation.

A.1. Qualified Names Module

Note that this module defines the parameter entity %xml-events-attrs.qname; This entity is intended to be used in the attribute lists of elements in any host language that permits the use of event attributes on elements in its own namespace. In this case the Host Language driver should set a parameter entity %XML-EVENTS.prefixed; to INCLUDE and a parameter entity %XML-EVENTS.prefix; to a value that is the prefix for the XML Events attributes.

<!-- .......................... ............................................. -->
<!-- XML Events Qname Module ........................................... -->
<!-- file: xml-events-qname-1.mod

This is XML Events - the Events Module for XML, a definition of access to the DOM events model.

Copyright 2000-2003 W3C (MIT, ERCIM, Keio), All Rights Reserved.

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

PUBLIC "-//W3C//ENTITIES XML Events Qnames 1.0//EN"
SYSTEM "http://www.w3.org/MarkUp/DTD/xml-events-qname-1.mod"

Revisions:
none

.......................... ............................................. -->

<!-- XML Events 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 XML Events and extensions.

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

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

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

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

<!-- 1. Declare a %XML-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 behavior follows the overall DTD prefixing scheme. -->

<!ENTITY % NS.prefixed "IGNORE">
<!ENTITY % XML-EVENTS.prefixed "%NS.prefixed;">

<!-- 2. Declare a parameter entity (eg., %XML-EVENTS.xmlns;) containing the URI reference used to identify the XML Events namespace -->

<!ENTITY % XML-EVENTS.xmlns "http://www.w3.org/2001/xml-events">

<!-- 3. Declare parameter entities (eg., %XML.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.

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

<!ENTITY % XML-EVENTS.pfx "" >

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

<!-- declare qualified name extensions here ............ -->

<!ENTITY % xml-events-qname-extra.mod "" >
%xml-events-qname-extra.mod;

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

<!ENTITY % xml-events.listener.qname "%XML-EVENTS.pfx;listener" >
A.2.XML Events Module

This is XML Events - the Events Module for XML.

Copyright 2000-2003 W3C (MIT, ERCIM, Keio), All Rights Reserved.

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

PUBLIC "-//W3C//ENTITIES XML Events 1.0//EN"
SYSTEM "http://www.w3.org/MarkUp/DTD/xml-events-1.mod"

Revisions:
(none)

-->

XML Events defines the listener element and its attributes -->

<!ENTITY % xml-events.listener.content "EMPTY" >

<!ELEMENT %xml-events.listener.qname; %xml-events.listener.content;>
<!ATTLIST %xml-events.listener.qname;
  id               ID           #IMPLIED
  event            NMTOKEN      #REQUIRED
  observer         IDREF        #IMPLIED
  target           IDREF        #IMPLIED
  handler          %anyURI.datatype;    #IMPLIED
  phase            (capture|default) #IMPLIED
  propagate        (stop|continue) #IMPLIED
  defaultAction    (cancel|perform) #IMPLIED
>

-- end of xml-events-1.mod -->

- 19 -
B. Schema Implementation

This appendix is normative.

The schema implementation of XML Events conforms to the requirements defined in [XHTMLSCHEMAMOD][p.26]. It is divided into an attributes module and an element module for the XML Events module defined in this Proposed Recommendation.

B.1. Attributes Module

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
   targetNamespace="http://www.w3.org/2001/xml-events"
   xmlns:ev="http://www.w3.org/2001/xml-events"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.w3.org/2001/XMLSchema
   http://www.w3.org/2001/XMLSchema.xsd"
   elementFormDefault="unqualified"
   blockDefault="#all"
   finalDefault="#all"
   attributeFormDefault="unqualified">
  <xs:annotation>
    <xs:documentation>
      This is the XML Schema for XML Events global attributes
      URI: http://www.w3.org/MarkUp/SCHEMA/xml-events-attribs-1.xsd
      $Id: xml-events-attribs-1.xsd,v 1.5 2003/08/02 09:36:54 mimasa Exp $
    </xs:documentation>
  </xs:annotation>
  <xs:annotation>
    <xs:documentation>
      XML Event Attributes
      These "global" event attributes are defined in "Attaching Attributes Directly to the Observer Element" of the XML Events specification.
    </xs:documentation>
  </xs:annotation>
  <xs:attribute name="event" type="xs:NMTOKEN"/>
  <xs:attribute name="observer" type="xs:IDREF"/>
  <xs:attribute name="target" type="xs:IDREF"/>
  <xs:attribute name="handler" type="xs:anyURI"/>
  <xs:attribute name="phase" default="default">
    <xs:simpleType>
      <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="capture"/>
        <xs:enumeration value="default"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:schema>
```
<xs:annotation>
  <xs:documentation>
  This is the XML Schema for XML Events
  
  URI: http://www.w3.org/MarkUp/SCHEMA/xml-events-1.xsd
  $Id: xml-events-1.xsd,v 1.6 2003/08/02 09:36:54 mimasa Exp$
  </xs:documentation>
</xs:annotation>

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
targetNamespace="http://www.w3.org/2001/xml-events"
xmlns="http://www.w3.org/2001/xml-events"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2001/XMLSchema
http://www.w3.org/2001/XMLSchema.xsd"
elementFormDefault="unqualified"
blockDefault="#all"
finalDefault="#all"
attributeFormDefault="unqualified">

<xs:annotation>
  <xs:documentation>
  This is the XML Schema for XML Events
  
  URI: http://www.w3.org/MarkUp/SCHEMA/xml-events-1.xsd
  $Id: xml-events-1.xsd,v 1.6 2003/08/02 09:36:54 mimasa Exp$
  </xs:documentation>
</xs:annotation>

<xs:annotation>
XML Events element listener

This module defines the listener element for XML Events. This element can be used to define event listeners. This module relies upon the XmlEvents.attlist attribute group defined in xml-events-attribs-1.xsd.

```
<xs:documentation>
  XML Events element listener

  This module defines the listener element for XML Events. This element can be used to define event listeners. This module relies upon the XmlEvents.attlist attribute group defined in xml-events-attribs-1.xsd.
</xs:documentation>
</xs:annotation>

<xs:attributeGroup name="listener.attlist">
  <xs:attribute name="event" use="required" type="xs:NMTOKEN"/>
  <xs:attribute name="observer" type="xs:IDREF"/>
  <xs:attribute name="target" type="xs:IDREF"/>
  <xs:attribute name="handler" type="xs:anyURI"/>
  <xs:attribute name="phase" default="default">
    <xs:simpleType>
      <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="capture"/>
        <xs:enumeration value="default"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="propagate" default="continue">
    <xs:simpleType>
      <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="stop"/>
        <xs:enumeration value="continue"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="defaultAction" default="perform">
    <xs:simpleType>
      <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="cancel"/>
        <xs:enumeration value="perform"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="id" type="xs:ID"/>
</xs:attributeGroup>

<xs:complexType name="listener.type">
  <xs:attributeGroup ref="listener.attlist"/>
</xs:complexType>

<xs:element name="listener" type="listener.type"/>
```
C. References

This appendix is normative.

C.1. Normative References

[DOM2EVENTS]
Available at: http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113
The latest version is available at: http://www.w3.org/TR/DOM-Level-2-Events

/XML]
"Extensible Markup Language (XML) 1.0 (Second Edition)", W3C Recommendation, T. Bray et al., eds., 6 October 2000.
Available at: http://www.w3.org/TR/2000/REC-xml-20001006
The latest version is available at: http://www.w3.org/TR/REC-xml

[SCHEMA]
Available at: http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/
The latest version is available at: http://www.w3.org/TR/xmlschema-2/

[RFC2119]
"Key words for use in RFCs to indicate requirement levels", RFC 2119, S. Bradner, March 1997.
Available at: http://www.rfc-editor.org/rfc/rfc2119.txt

[XMLNAMES]
Available at: http://www.w3.org/TR/1999/REC-xml-names-19990114
The latest version is available at: http://www.w3.org/TR/REC-xml-names

C.2. Other References

[HTML4]
Available at: http://www.w3.org/TR/1999/REC-html401-19991224
The latest version is available at: http://www.w3.org/TR/html4

[SMIL20]
Available at: http://www.w3.org/TR/2001/REC-smil20-20010807/
The latest version is available at: http://www.w3.org/TR/smil20

[SVG]
Available at: http://www.w3.org/TR/2001/REC-SVG-20010904/
The [latest version] is available at: http://www.w3.org/TR/SVG/

[XHTML]

"XHTML™ 1.0: The Extensible HyperText Markup Language (Second Edition)": S. Pemberton et al., 26 January 2000, revised 1 August 2002.
Available at: http://www.w3.org/TR/2002/REC-xhtml1-20020801
The [latest version] is available at: http://www.w3.org/TR/xhtml1

[XHTMLMOD]

Available at: http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410
The [latest version] is available at: http://www.w3.org/TR/xhtml-modularization

[XHTMLSCHEMAMOD]

Available at: http://www.w3.org/TR/2003/WD-xhtml-m12n-schema-20031003
The [latest version] is available at: http://www.w3.org/TR/xhtml-m12n-schema
D. 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), Patrick Schmitz, and Peter Stark (Ericsson) for their significant contributions to the evolution of this specification.

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

- Steven Pemberton, CWI/W3C (HTML Working Group Chair)
- Daniel Austin, W. W. Grainger, Inc.
- Jim Bigelow, Hewlett-Packard Company
- Mark Birbeck, x-port.net Ltd. (Invited Expert)
- Jonny Axelsson, Opera Software
- Tantek Çelik, Microsoft Corporation
- Beth Epperson, Netscape/AOL
- Masayasu Ishikawa, W3C (Team Contact)
- Shin’ichi Matsui, Panasonic
- Shane McCarron, Applied Testing and Technology (Invited Expert)
- Ann Navarro, WebGeek, Inc. (Invited Expert)
- Subramanian Peruvemba, Oracle Corporation
- Sebastian Schnitzenbaumer, SAP AG