XML Events 2

An Events Syntax for XML

W3C Working Group Note 16 December 2010

This version:
    http://www.w3.org/TR/2010/NOTE-xml-events2-20101216
Latest version:
    http://www.w3.org/TR/xml-events2
Previous version:
    http://www.w3.org/TR/2007/WD-xml-events-20070216
Diff from previous version:
    xml-events-diff.html
Previous recommendation:
    http://www.w3.org/TR/2003/REC-xml-events-20031014
Diff from previous recommendation:
    xml-events-rec-diff.html

Editors:
    Shane McCarron, Applied Testing and Technology, Inc.
    Mark Birbeck, Sidewinder Labs
    Roland Merrick, IBM
Version 1 Editors:
    Shane McCarron, Applied Testing and Technology, Inc.
    Steven Pemberton, CWI/W3C®
    T. V. Raman, IBM Corporation

This document is also available in these non-normative formats: PostScript version, PDF version, ZIP archive, and Gzip'd TAR archive.

The English version of this specification is the only normative version. Non-normative translations may also be available.

Abstract

This specification defines three modules: XML Events to define events and their characteristics; XML Handlers to define mappings between events and actions; and XML Scripting to assist in defining functions to support the handlers. These modules work together to provide XML languages with the ability to uniformly integrate event listeners and associated event handlers with Document Object Model (DOM) Level 3 event interfaces [DOM3EVENTS [p.41]]. 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 Working Group Note. The XHTML2 Working Group's charter expired before it could complete work on this document. Related work is being done by the Forms Working Group.

Publication as a Working Group Note does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This document has been produced by the W3C XHTML 2 Working Group as part of the HTML Activity. The goals of the XHTML 2 Working Group are discussed in the XHTML 2 Working Group charter.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Public discussion of HTML takes place on www-html@w3.org (archive). To subscribe send an email to www-html-request@w3.org with the word *subscribe* in the subject line.

Please report errors in this document to www-html-editor@w3.org (archive).

Contents

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 [DOM3EVENTS [p.41]], 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 DOM3: 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’. When it arrives at the target it is in the ’target’ (or ’at-target’) phase. 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.41]] 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.42]] uses a similar method.

The process of defining XHTML and related markup languages 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.41]].
- 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.

The elements and attributes defined in this specification are the method of binding a DOM level 3 event at an element to an event handler. They encapsulate various aspects of the DOM level 3 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.41]].

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.33] or Appendix A - DTD Implementation [p.27] , combined with the constraints expressed in its Host Language implementation.

2. If the Host Language does not incorporate XML Events elements and attributes into its own namespace, the document MUST contain an `xmlns` declaration for the XML Events namespace [XMLNAMES [p.41]]. 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 mandatory elements and attributes defined in this specification MUST be included in the content model of the Host Language.

2.3. Host Language Processor Conformance

A conforming Host Language Processor 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.41]] 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.41]].

The XML Events Module supports the following element and attributes:

Element	Attributes	Minimal Content Model
listener [p.9]	event (QNames), observer (IDREFS), eventTarget (IDREFS), function (CDATA), handler (URI), phase ("bubble" | "capture" | "default"* | "target"), propagate ("stop" | "continue"*), defaultAction ("cancel" | "perform"*), xml:id ([XMLID [p.41]])	EMPTY

Implementations: DTD [p.29] , XML Schema [p.34]

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 one, or more, event types for which the listener is being registered.

observer
: The optional `observer` attribute specifies the `id` of one, or more, elements 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.13] ").

eventTarget

The optional `eventTarget` attribute specifies the `id` of one, or more, target elements of the event (i.e., the node that caused the event). If this attribute is present, only events that match both the `event` and `eventTarget` 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

```
<listener event="click" observer="para1" eventTarget="link1" handler="#clicker"/>
...
<p id="para1">
    Here is some content in a paragraph that includes a link to
    <a id="link1" href="Overview.html">the
        <em>draft</em>
    document</a>.
</p>
```

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 `eventTarget` attribute.

function

The `function` attribute identifies an optional function name that will be called when the event reaches the observer.

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. 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.13] ").

If both the `function` and `handler` attributes are defined, the `handler` element MUST be ignored.

phase

The optional `phase` attribute specifies when (during which DOM 3 event propagation phase) the listener will be activated by the desired event.

`bubble`
: Listener is activated during the bubbling phase.

`capture`
: Listener is activated during the capture phase.

`target`
: Listener is activated during the target phase.

`default`
: Listener is activated during target or bubbling phases.

If no `phase` attribute is specified it is equivalent to having specified `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. The use of `phase="default"` is equivalent to invoking the DOM `addEventListener` method with a `useCapture` that is `false`.

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, target, or bubble phase).

  `stop`
  : event propagation stops

  `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.

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="DOMActivate" 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="prog1"`.

```
<listener event="overflow" observer="prog1" eventTarget="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.

   ```
   <listener event="DOMActivate" observer="embargo" handler="#popup"
           phase="capture" propagate="stop"/>
   ```

4. This attaches a handler from another document.

   ```
   <listener event="DOMActivate" observer="image1"
       handler="/handlers/events.xml#activate"/>
   ```

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.41]], to attach the attributes to other elements. *In the schema implementations, this is done by defining the attribute collection `XMLEvents`.*

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

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

would have the same effect as

```
<ev:listener event="click" observer="button1" 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="DOMActivate" 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 `eventTarget` 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"`.

```
<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.

```
<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.

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

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

```
<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.

```
<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">
      <prompt>Security violation!</prompt>
      <submit next="apprehend_felon" namelist="user_id"/>
    </catch>
  </field>
  <block>
    <goto next="#get_city"/>
  </block>
</form>
```

6. This example shows three handlers for different events. The observer for all three is the `secret` element.

```
<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.

The effect of omitted observer and handler attributes

	Handler present	**Handler omitted**
Observer present	(As declared)	Element is handler
Observer omitted	Element is observer	Element is handler Parent is observer

3.5. Event Scope

In some environments (e.g., User Agents) it is possible to have multiple documents available simultaneously - even to the extent that one document is *embedded* within another (see the `iframe` element in [XHTMLMOD [p.41]]). When multiple documents are available, each of these documents is in a separate *context*. Consequently, it is not possible for DOM events to move among these separate documents. This means, for example, that if one document is embedded within another, an event fired in the context of that document WILL NOT be available to the event handlers in enclosing document. See [DOM3EVENTS [p.41]] for more information.

3.6. Listener Registration

Dynamic content MUST be evaluated by a user agent to ensure that any new listeners or handlers are correctly registered with the event system.

An implementation MUST behave as if any event handlers are registered prior to the DOM3 `load` event being fired.

4. The XML Handlers Module

This section is normative.

The purpose of this module is to provide a declarative way to map specific events to a series of one or more actions. Those actions are declared either by the Host Language (e.g., [XFORMS [p.42]]) or within the document using the XML Events module above. Implementing actual handlers for the events remains the purview of the Host Language, supported scripting languages, etc.

The XML Handlers Module supports the following elements and attributes:

Element	Attributes	Minimal Content Model
action [p.18]	event (QNames), eventTarget (IDREFS), declare ("declare"), if (ConditionalExpression [p.25]), while (ConditionalExpression [p.25]), xml:id ([XMLID [p.41]])	(action | dispatchEvent | addEventListener | removeEventListener | stopPropagation | preventDefault)+
dispatchEvent [p.19]	eventType (QName), targetid (IDREFS), bubbles ("bubbles"), cancelable ("cancelable"), xml:id ([XMLID [p.41]])	EMPTY
addEventListener [p.19]	event* (QName), handler* (IDREF), phase ("bubble" | "capture" | "default"* | "target"), xml:id ([XMLID [p.41]])	EMPTY
removeEventListener [p.19]	event* (QName), handler* (IDREF), phase ("bubble" | "capture" | "default"* | "target"), xml:id ([XMLID [p.41]])	EMPTY
stopPropagation [p.19]	event* (QName), xml:id ([XMLID [p.41]])	EMPTY
preventDefault [p.19]	event* (QName), xml:id ([XMLID [p.41]])	EMPTY

When this module is selected, the XML Events module MUST also be selected.

Implementations: DTD [p.30] , XML Schema [p.36]

4.1. The action Element

The `action` element is used to group event handler elements (including other `action` elements) that will act *in sequence* as handlers for an event. The `action` element takes the following attributes:

event
: The required `event` attribute specifies one, or more, event types this action will handle.

eventTarget
: The optional `eventTarget` attribute specifies the `id` of one, or more, target elements of the event (i.e., the node that caused the event). If this attribute is present, only events that match both the `event` and `eventTarget` attributes will be processed. Clearly because of the way events propagate, the target element should be a descendant node of the observer element, or the observer element itself.

declare
: When present, this boolean attribute makes the current element (and any elements it may contain) a declaration only.

if
: The optional if attribute allows a condition to be specified. This condition must be met in order for the event handler to be activated. The condition is specified using a Conditional Expression [p.25] . There is no default value.

  This attribute allows event handlers to be specified that respond not just to named events, but to more specific conditions, such as a mouse click with the control key pressed:

```
<action event="click" if="event('ctrlKey') = true()">
 . . .
</action>
```

  The event function is described in Event Function [p.25] .

while
: The optional while attribute allows a condition to be specified. This condition must be met in order for the event handler to be activated. The condition is specified using an Conditional Expression [p.25] . There is no default value.

  This attribute allows event handlers to be specified that perform their action whilst some condition remains true.

  EDITORS' NOTE: Can't think of an example that only makes use of what we have in this spec, i.e., the event() function. We may need to do something like delete a list in XForms.

4.2. The dispatchEvent Element

The `dispatchEvent` element triggers the event identified by the `eventType` attribute, delivering it to the element identified by the `to` attribute.

The `dispatchEvent` element also defines two additional attributes:

bubbles
: Optional boolean indicating if this event bubbles as defined in [DOM3EVENTS [p.41]]. The default value will depend on the actual event being dispatched.

cancelable
: Optional boolean indicating if this event is cancelable as defined in [DOM3EVENTS [p.41]]. The default value will depend on the actual event being dispatched.

targetid
: This attribute specifies the `id` of one or more elements to which the event is dispatched. If this is not specified, it defaults to the parent of the `dispatchEvent` element

eventType
: The QName of the event to be triggered.

4.3. The addEventListener element

This element allows the registration of a listener on a specific event, as defined in [DOM3EVENTS [p.41]].

4.4. The removeEventListener element

The `removeEventListener` element de-registers the handler identified by the required `handler` attribute for the event identified by the required `event` attribute.

4.5. The stopPropagation element

The `stopPropagation` element is used to prevent further propagation of an event during event flow. If this is called by any EventListener the event will cease propagating through the tree. The event will complete dispatch to all listeners on the current EventTarget before event flow stops. This action may be used during any stage of event flow.

4.6. The preventDefault element

If an event is cancelable, the `preventDefault` action is used to signify that the event is to be canceled, meaning any default action normally taken by the implementation as a result of the event will not occur. If, during any stage of event flow, the `preventDefault` action is called the event is canceled. Any default action associated with the event will not occur. Calling this action for a non-cancelable event has no effect. Once preventDefault has been called it will remain in effect throughout the remainder of the event’s propagation. This action may be used during any stage of event flow.

5. The XML Scripting Module

This section is normative.

The XML Script Module defines an element for defining event handlers via a scripting language. This module can be used as a replacement for the XHTML Scripting Module defined in XHTML Modularization [XHTMLMOD [p.41]].

The XML Script Module supports the following element:

Element	Attributes	Minimal Content Model
script [p.21]	charset (Charset), defer ("defer"), implements (`URIorSafeCURIEs`), src (URI), type (ContentType), xml:id ([XMLID [p.41]])	PCDATA

When the Scripting module is included AND the XML Handlers Module is included, the `script` element is added to the content model of the XML Handlers Module `action` element.

Implementations: DTD [p.32] , XML Schema [p.38]

5.1. The script Element

The `script` element contains or references *scripts* that may register one or more event handlers for a document through a *scripting language* that is supported by the implementation.

charset
: The optional `charset` attribute specifies the encoding of the remote resource identified by the `src` attribute.

defer
: This attribute provides a hint to the user agent that the contents of the `script` element will not effect the generation of the document.

implements
: The optional `implements` attribute indicates that the script provides an implementation of the feature or features identified via this attribute. The script SHOULD only be loaded and used if the user agent does not have an implementation of the specified feature.

src
: The `src` attribute identifies an external resource that provides the script implementation.

type
: Defines the programming language in which the script is implemented by using its media type.

xml:id
: The optional xml:id attribute assigns an identifier to an element. The value of this attribute must be unique within a document.

The event handler(s) may be defined within the contents of the `script` element or in an external file. If the `src` attribute is not set, user agents MUST interpret the contents of the element as the handler. If the `src` has a URI value, user agents MUST ignore the element’s contents and retrieve the handler via the URI.

6. 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 DOM 3 Events [DOM3EVENTS [p.41]], to which you should refer for their names and semantics.

7. Conditional Expressions

XML Events uses XPath-like expressions to specify conditionals (if [p.18] , while [p.18]). As described in section 1 of [XPATH [p.42]], each expression is evaluated within a context, which is made up of:

- a node (the context node)
- a pair of non-zero positive integers (the context position and the context size)
- a set of variable bindings
- a function library
- the set of namespace declarations in scope for the expression

By default, XML Events expressions have no context node, and so the context position is 0 and the context size is 0. There are no variable bindings, and the function library contains the functions described below. It is not necessary to provide namespace declarations.

Host languages that import XML Events may provide a richer context, and MUST specify whether their context is the same as that provided here, or more.

7.1. Function Library

The function library consists of the following functions:

- event

7.1.1. Event Function

```
object event(string)
```

Function `event` returns the value of a property of the current event object, as determined by the string argument. The value returned will be typed, depending on the property. For example, the MouseEvent interface [DOM3EVENTS [p.41]] has the attribute `shiftKey`, which is a `boolean`. This can be accessed by passing the string value '`shiftKey`' to the event function. The result will be an XPath-like boolean.

A. DTD Implementation

This appendix is *normative*.

The DTD implementation of XML Events conforms to the requirements defined in [XHTMLMOD [p.41]]. 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-2.mod

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

     Copyright 2000-2007 W3C (MIT, ERCIM, Keio), All Rights Reserved.

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

       PUBLIC "-//W3C//ENTITIES XML Events Qnames 2.0//EN"
       SYSTEM "http://www.w3.org/MarkUp/DTD/xml-events-qname-2.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-2.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.
-->
<!ENTITY % XML-EVENTS.prefix  "" >

<!-- 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.
-->
<![%XML-EVENTS.prefixed;[
<!ENTITY % XML-EVENTS.pfx  "%XML-EVENTS.prefix;:" >
]]>
<!ENTITY % XML-EVENTS.pfx  "" >

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

<!-- 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.
-->
<!ENTITY % XML-EVENTS.xmlns.extra.attrib "" >


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

<!-- 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" >


<!ENTITY % xml-handlers.action.qname  "%XML-EVENTS.pfx;action" >
<!ENTITY % xml-script.script.qname  "%XML-EVENTS.pfx;script" >
<!ENTITY % xml-handlers.dispatchEvent.qname  "%XML-EVENTS.pfx;dispatchEvent" >
<!ENTITY % xml-handlers.addEventListener.qname  "%XML-EVENTS.pfx;addEventListener" >
<!ENTITY % xml-handlers.removeEventListener.qname  "%XML-EVENTS.pfx;removeEventListener" >
```

```
<!ENTITY % xml-handlers.stopPropagation.qname  "%XML-EVENTS.pfx;stopPropagation" >
<!ENTITY % xml-handlers.preventDefault.qname  "%XML-EVENTS.pfx;preventDefault" >


<!-- The following defines a PE for use in the attribute sets of elements in
     other namespaces that want to incorporate the XML Event attributes. Note
     that in this case the XML-EVENTS.pfx should always be defined. -->

<!ENTITY % xml-events.attrs.qname
   "%XML-EVENTS.pfx;event            NMTOKEN      #IMPLIED
    %XML-EVENTS.pfx;observer         IDREFS       #IMPLIED
    %XML-EVENTS.pfx;eventTarget      IDREFS       #IMPLIED
    %XML-EVENTS.pfx;handler          %URI.datatype;        #IMPLIED
    %XML-EVENTS.pfx;phase            (capture|default) #IMPLIED
    %XML-EVENTS.pfx;propagate        (stop|continue) #IMPLIED
    %XML-EVENTS.pfx;defaultAction    (cancel|perform) #IMPLIED
    %XML-EVENTS.pfx;condition        CDATA        #IMPLIED"
    >

<!-- end of xml-events-qname-2.mod -->
```

A.2. XML Events Module

```
<!-- ...................................................................... -->
<!-- XML Events Module .................................................... -->
<!-- file: xml-events-2.mod

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

     Copyright 2000-2007 W3C (MIT, ERCIM, Keio), All Rights Reserved.

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

       PUBLIC "-//W3C//ENTITIES XML Events 2.0//EN"
       SYSTEM "http://www.w3.org/MarkUp/DTD/xml-events-2.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;
    xml:id            ID            #IMPLIED
    event             NMTOKEN       #REQUIRED
    observer          IDREF         #IMPLIED
    eventTarget       IDREF         #IMPLIED
    handler           %anyURI.datatype;        #IMPLIED
    phase             (capture|default) #IMPLIED
    propagate         (stop|continue) #IMPLIED
    defaultAction    (cancel|perform)  #IMPLIED
>

<!-- end of xml-events-2.mod -->
```

A.3. XML Handlers Module

```
<!-- ................................................................... -->
<!-- XML Handlers Module ............................................... -->
<!-- file: xml-handlers-1.mod

     This is XML Handlers - the Handlers Module for XML.
     a redefinition of support for handlers of the DOM event model.

     Copyright 2007-2008 W3C (MIT, ERCIM, Keio), All Rights Reserved.

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

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

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


<!-- XML Handlers defines the various element and attributes -->

<!ENTITY % xml-handlers.action.content
    "( %xml-handlers.action.qname; |
       %xml-handlers.dispatchevent.qname; |
       %xml-handlers.addEventListener.qname; |
       %xml-handlers.removeEventListener.qname; |
       %xml-handlers.stopPropagation.qname; |
       %xml-handlers.preventDefault.qname; |
       %xml-handlers.action.extras; )+ "
>

<!ELEMENT %xml-handlers.action.qname; %xml-handlers.action.content;>
<!ATTLIST %xml-handlers.action.qname;
    xml:id           ID                    #IMPLIED
    event            %QName.datatype;      #IMPLIED
    eventTarget      IDREF                 #IMPLIED
    declare          ( declare )           #IMPLIED
    if               CDATA                 #IMPLIED
    while            CDATA                 #IMPLIED
>

<!ENTITY % xml-handlers.dispatchEvent.content "NONE" >
<!ELEMENT %xml-handlers.dispatchEvent.qname;
          %xml-handlers.dispatchEvent.content >

<ENTITY % xml-handlers.dispatchEvent.attlist "INCLUDE" >
<![%xml-handlers.dispatchEvent.attlist;[
<!ATTLIST %xml-handlers.dispatchEvent.qname;
      %XML-EVENTS.xmlns.attrib;
      xml:id       ID                        #IMPLIED
      targetid     IDREFS                    #IMPLIED
      eventType    %QName.datatype;          #IMPLIED
      bubbles      ( bubbles )               #IMPLIED
      cancelable   ( cancelable )            #IMPLIED
>
<!-- end of xml-handlers.dispatchEvent.attlist -->]]>

<!ENTITY % xml-handlers.addEventListener.content "NONE" >
<!ELEMENT %xml-handlers.addEventListener.qname;
          %xml-handlers.addEventListener.content >

<ENTITY % xml-handlers.addEventListener.attlist "INCLUDE" >
<![%xml-handlers.addEventListener.attlist;[
<!ATTLIST %xml-handlers.addEventListener.qname;
      %XML-EVENTS.xmlns.attrib;
      xml:id       ID                        #IMPLIED
      event        %QName.datatype;          #REQUIRED
      handler      IDREF                     #REQUIRED
      phase        (capture|default)         #IMPLIED
>
<!-- end of xml-handlers.addEventListener.attlist -->]]>

<!ENTITY % xml-handlers.removeEventListener.content "NONE" >
<!ELEMENT %xml-handlers.removeEventListener.qname;
          %xml-handlers.removeEventListener.content >

<ENTITY % xml-handlers.removeEventListener.attlist "INCLUDE" >
<![%xml-handlers.removeEventListener.attlist;[
<!ATTLIST %xml-handlers.removeEventListener.qname;
      %XML-EVENTS.xmlns.attrib;
      xml:id       ID                        #IMPLIED
      event        %QName.datatype;          #REQUIRED
      handler      IDREF                     #REQUIRED
      phase        (capture|default)         #IMPLIED
>
<!-- end of xml-handlers.addEventListener.attlist -->]]>

<!ENTITY % xml-handlers.stopPropagation.content "NONE" >
<!ELEMENT %xml-handlers.stopPropagation.qname;
          %xml-handlers.stopPropagation.content >

<ENTITY % xml-handlers.stopPropagation.attlist "INCLUDE" >
<![%xml-handlers.stopPropagation.attlist;[
<!ATTLIST %xml-handlers.stopPropagation.qname;
      %XML-EVENTS.xmlns.attrib;
      xml:id       ID                        #IMPLIED
      event        %QName.datatype;          #REQUIRED
>
<!-- end of xml-handlers.stopPropagation.attlist -->]]>

<!ENTITY % xml-handlers.preventDefault.content "NONE" >
<!ELEMENT %xml-handlers.preventDefault.qname;
          %xml-handlers.preventDefault.content >

<ENTITY % xml-handlers.preventDefault.attlist "INCLUDE" >
<![%xml-handlers.preventDefault.attlist;[
<!ATTLIST %xml-handlers.preventDefault.qname;
      %XML-EVENTS.xmlns.attrib;
      xml:id       ID                        #IMPLIED
      event        %QName.datatype;          #REQUIRED
>
<!-- end of xml-handlers.preventDefault.attlist -->]]>

<!-- end of xml-handlers-1.mod -->
```

A.4. XML Scripting Module

```
<!-- ...................................................................... -->
<!-- XML Scripting Module .................................................. -->
<!-- file: xml-script-1.mod

     This is XML Scripting - the Scripting Module for XML.

     Copyright 2008 W3C (MIT, ERCIM, Keio), All Rights Reserved.

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

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

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


<!-- XML Scripting defines the following element -->

<!ENTITY % xml-script.script.content "( #PCDATA )" >
<!ELEMENT %xml-script.script.qname; %xml-handlers.script.content; >
<!ENTITY % xml-script.script.attlist  "INCLUDE" >
<![%xml-script.script.attlist;[
<!ATTLIST %xml-script.script.qname;
      %XML-EVENTS.xmlns.attrib;
      xml:id       ID                        #IMPLIED
      encoding     %Charset.datatype;        #IMPLIED
      type         %ContentType.datatype;    #REQUIRED
      src          %URI.datatype;            #IMPLIED
      implements   %URIorSafeCURIEs.datatype;  #IMPLIED
>
<!-- end of xml-script.script.attlist -->]]>

<!-- end of xml-script-1.mod -->
```

B. Schema Implementation

This appendix is *normative*.

The schema implementation of XML Events conforms to the requirements defined in [XHTMLMOD [p.41]]. 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 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-2.xsd
      $Id: xml-events-attribs-2.xsd,v 1.4 2008/10/20 15:13:53 ahby Exp $
    </xs:documentation>
    <xs:documentation source="xml-events-copyright-2.xsd"/>
  </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="eventTarget" 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="bubble"/>
        <xs:enumeration value="capture"/>
        <xs:enumeration value="default"/>
        <xs:enumeration value="target"/>
      </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:attributeGroup name="XmlEvents.attlist">
    <xs:attribute ref="ev:event"/>
    <xs:attribute ref="ev:observer"/>
    <xs:attribute ref="ev:eventTarget"/>
    <xs:attribute ref="ev:handler"/>
    <xs:attribute ref="ev:phase"/>
    <xs:attribute ref="ev:propagate"/>
    <xs:attribute ref="ev:defaultAction"/>
  </xs:attributeGroup>

</xs:schema>
```

B.2. XML Events Module

```
<?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-2.xsd
      $Id: xml-events-2.xsd,v 1.4 2008/10/20 15:13:53 ahby Exp $
    </xs:documentation>
    <xs:documentation source="xml-events-copyright-2.xsd"/>
  </xs:annotation>
  <xs:annotation>
    <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-2.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="eventTarget" 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"/>

</xs:schema>
```

B.3. XML Handlers Module

```
<?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:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
    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 Handlers

      URI: http://www.w3.org/MarkUp/SCHEMA/xml-handlers-1.xsd
      $Id: xml-handlers-1.xsd,v 1.2 2008/10/20 15:13:53 ahby Exp $
    </xs:documentation>
    <xs:documentation source="xml-events-copyright-2.xsd"/>
  </xs:annotation>

  <xs:attributeGroup name="action.attlist">
    <xs:attribute name="event" use="required" type="xs:QName"/>
    <xs:attribute name="eventTarget" type="xs:IDREF"/>
    <xs:attribute name="declare">
        <xs:simpleType>
            <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="declare"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="if" type="xs:normalizedString"/>
    <xs:attribute name="while" type="xs:normalizedString"/>
    <xs:attribute name="id" type="xs:ID"/>
  </xs:attributeGroup>

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

  <xs:element name="action" type="action.type"/>

  <xs:attributeGroup name="dispatchEvent.attlist">
    <xs:attribute name="eventType" type="xs:QName"/>
    <xs:attribute name="targetid">
        <xs:simpleType>
            <xs:list itemType="xs:IDREF"/>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="bubbles">
        <xs:simpleType>
```

```
            <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="bubbles"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="cancelable">
        <xs:simpleType>
            <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="cancelable"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="id" type="xs:ID"/>
  </xs:attributeGroup>

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

  <xs:element name="dispatchEvent" type="dispatchEvent.type"/>

  <xs:attributeGroup name="addEventListener.attlist">
    <xs:attribute name="event" use="required" type="xs:QName"/>
    <xs:attribute name="handler" use="required" type="xs:IDREF"/>
    <xs:attribute name="phase" default="default">
        <xs:simpleType>
            <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="bubble"/>
                <xs:enumeration value="capture"/>
                <xs:enumeration value="default"/>
                <xs:enumeration value="target"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="id" type="xs:ID"/>
  </xs:attributeGroup>

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

  <xs:element name="addEventListener" type="addEventListener.type"/>

  <xs:attributeGroup name="removeEventListener.attlist">
    <xs:attribute name="event" use="required" type="xs:QName"/>
    <xs:attribute name="handler" use="required" type="xs:IDREF"/>
    <xs:attribute name="phase" default="default">
        <xs:simpleType>
            <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="bubble"/>
                <xs:enumeration value="capture"/>
                <xs:enumeration value="default"/>
                <xs:enumeration value="target"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="id" type="xs:ID"/>
  </xs:attributeGroup>
  <xs:complexType name="removeEventListener.type">
    <xs:attributeGroup ref="removeEventListener.attlist"/>
  </xs:complexType>

  <xs:element name="removeEventListener" type="removeEventListener.type"/>

  <xs:attributeGroup name="stopPropagation.attlist">
    <xs:attribute name="event" use="required" type="xs:QName"/>
    <xs:attribute name="id" type="xs:ID"/>
  </xs:attributeGroup>

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

  <xs:element name="stopPropagation" type="stopPropagation.type"/>

  <xs:attributeGroup name="preventDefault.attlist">
    <xs:attribute name="event" use="required" type="xs:QName"/>
    <xs:attribute name="id" type="xs:ID"/>
  </xs:attributeGroup>

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

  <xs:element name="preventDefault" type="stopPropagation.type"/>
</xs:schema>
```

B.4. XML Scripting Module

```
<?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:xh11d="http://www.w3.org/1999/xhtml/datatypes/"
    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 Scripting

      URI: http://www.w3.org/MarkUp/SCHEMA/xml-script-1.xsd
      $Id: xml-script-1.xsd,v 1.1 2008/06/25 14:31:55 ahby Exp $
    </xs:documentation>
    <xs:documentation source="xml-events-copyright-2.xsd"/>
  </xs:annotation>
  <xs:attributeGroup name="script.attlist">
    <xs:attribute name="encoding" type="xh11d:Charset"/>
    <xs:attribute name="implements" type="xh11d:URIorSafeCURIEs"/>
    <xs:attribute name="src" type="xs:anyURI"/>
    <xs:attribute name="type" type="xh11d:ContentTypes"/>
    <xs:attribute name="id" type="xs:ID"/>
  </xs:attributeGroup>

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

  <xs:element name="script" type="script.type"/>
</xs:schema>
```

C. References

This appendix is *normative*.

C.1. Normative References

[DOM3EVENTS]
"*Document Object Model (DOM) Level 3 Events Specification*", W3C Recommendation, BjÃ¶rn HÃ¶hrmann, *ed.*, 21 December 2007.
Available at: http://www.w3.org/TR/DOM-Level-3-Events/
The latest version is available at: http://www.w3.org/TR/DOM-Level-3-Events

[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

[RFC2616]
"*Hypertext Transfer Protocol - HTTP/1.1*", RFC 2119, H. Fielding *et al.*, *eds.*, June 1999.
Available at: http://www.rfc-editor.org/rfc/rfc2616.txt

[XHTMLMOD]
"*Modularization of XHTML™ 1.1*", W3C Recommendation, D. Austin *et al.*, *eds.*, 8 October 2009.
Available at: http://www.w3.org/TR/2008/REC-xhtml-modularization-20081008
The latest version is available at: http://www.w3.org/TR/xhtml-modularization

[XML]
"*Extensible Markup Language (XML) 1.0 (Fourth Edition)*", W3C Recommendation, T. Bray *et al.*, *eds.*, 16 August 2006.
Available at: http://www.w3.org/TR/2006/REC-xml-20060816
The latest version is available at: http://www.w3.org/TR/REC-xml

[XMLID]
"*xml:id Version 1.0*", W3C Recommendation, J. Marsh, D. Veillard, N. Walsh, *eds.*, 9 September 2005.
Available at: http://www.w3.org/TR/2005/REC-xml-id-20050909/

[XMLNAMES]
"*Namespaces in XML*", W3C Recommendation, T. Bray *et al.*, *eds.*, 14 January 1999.
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]
"*HTML 4.01 Specification*", W3C Recommendation, D. Raggett *et al.*, *eds.*, 24 December 1999.
Available at: http://www.w3.org/TR/1999/REC-html401-19991224
The latest version is available at: http://www.w3.org/TR/html4

[MIMETYPES]
"*MIME Media Types*", IANA.
Available at: http://www.iana.org/assignments/media-types/

[SVG]
"*Scalable Vector Graphics (SVG) 1.0 Specification*", W3C Recommendation, J. Ferraiolo, *ed.*, 4 September 2001.
Available at: http://www.w3.org/TR/2001/REC-SVG-20010904/
The latest version is available at: http://www.w3.org/TR/SVG/

[XFORMS]
"*XForms 1.1*", W3C Working Draft, M. Dubinko *et al.*, *eds.*, 15 November 2004.
Available at: http://www.w3.org/TR/2004/WD-xforms11-20041115/

[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

[XPATH]
"*XML Path Language*", W3C Recommendation, James Clark, *et al.*, *eds.*, 16 November 1999.
Available at: http://www.w3.org/TR/1999/REC-xpath-19991116
The latest version is available at: http://www.w3.org/TR/xpath

D. Change History

This section is informative.

In developing version 2 of XML Events, the following major changes have been made:

- Changed from using DOM 2 Events to DOM 3 Events.
- Added XML Handlers Module.
- Added XML Scripting Module.
- Added XML Schema implementation of modules.
- Permitted incorporation of elements and attributes into other namespaces.

E. Acknowledgments

This section is informative.

This document was prepared by the W3C XHTML2 Working Group. The members at the time of publication of the Note were:

- Markus Gylling, DAISY Consortium (XHTML 2 Working Group Co-Chair)
- Steven Pemberton, CWI (XHTML 2 Working Group Co-Chair)
- Mark Birbeck, Sidewinder Labs (Invited Expert)
- Susan Borgrink, Progeny Systems
- Christina Bottomley, Society for Technical Communication (STC)
- Alessio Cartocci, International Webmasters Association / HTML Writers Guild (IWA-HWG)
- Alexander Graf, University of Innsbruck
- Tina Holmboe, Greytower Technologies (Invited Expert)
- John Kugelman, Progeny Systems
- Luca Mascaro, International Webmasters Association / HTML Writers Guild (IWA-HWG)
- Shane McCarron, Applied Testing and Technology, Inc. (Invited Expert)
- Michael Rawling, IVIS Group Limited
- Gregory Rosmaita, Invited Expert
- Sebastian Schnitzenbaumer, Dreamlab Technologies AG
- Richard Schwerdtfeger, IBM
- Elias Torres, IBM
- Masataka Yakura, Mitsue-Links Co., Ltd.
- Toshihiko Yamakami, ACCESS Co., Ltd.

At publication of the first edition, the acknowledgements included was:

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