XHTML Mobile Profile

Approved Version 1.3 – 29 Mar 2011

Open Mobile Alliance

OMA-TS-XHTMLMP-V1_3-20110329-A

Use of this document is subject to all of the terms and conditions of the Use Agreement located at http://www.openmobilealliance.org/UseAgreement.html.

Unless this document is clearly designated as an approved specification, this document is a work in process, is not an approved Open Mobile Alliance™ specification, and is subject to revision or removal without notice.

You may use this document or any part of the document for internal or educational purposes only, provided you do not modify, edit or take out of context the information in this document in any manner. Information contained in this document may be used, at your sole risk, for any purposes. You may not use this document in any other manner without the prior written permission of the Open Mobile Alliance. The Open Mobile Alliance authorizes you to copy this document, provided that you retain all copyright and other proprietary notices contained in the original materials on any copies of the materials and that you comply strictly with these terms. This copyright permission does not constitute an endorsement of the products or services. The Open Mobile Alliance assumes no responsibility for errors or omissions in this document.

Each Open Mobile Alliance member has agreed to use reasonable endeavors to inform the Open Mobile Alliance in a timely manner of Essential IPR as it becomes aware that the Essential IPR is related to the prepared or published specification. However, the members do not have an obligation to conduct IPR searches. The declared Essential IPR is publicly available to members and non-members of the Open Mobile Alliance and may be found on the "OMA IPR Declarations" list at http://www.openmobilealliance.org/ipr.html. The Open Mobile Alliance has not conducted an independent IPR review of this document and the information contained herein, and makes no representations or warranties regarding third party IPR, including without limitation patents, copyrights or trade secret rights. This document may contain inventions for which you must obtain licenses from third parties before making, using or selling the inventions. Defined terms above are set forth in the schedule to the Open Mobile Alliance Application Form.

NO REPRESENTATIONS OR WARRANTIES (WHETHER EXPRESS OR IMPLIED) ARE MADE BY THE OPEN MOBILE ALLIANCE OR ANY OPEN MOBILE ALLIANCE MEMBER OR ITS AFFILIATES REGARDING ANY OF THE IPR'S REPRESENTED ON THE "OMA IPR DECLARATIONS" LIST, INCLUDING, BUT NOT LIMITED TO THE ACCURACY, COMPLETENESS, VALIDITY OR RELEVANCE OF THE INFORMATION OR WHETHER OR NOT SUCH RIGHTS ARE ESSENTIAL OR NON-ESSENTIAL.

THE OPEN MOBILE ALLIANCE IS NOT LIABLE FOR AND HEREBY DISCLAIMS ANY DIRECT, INDIRECT, PUNITIVE, SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR EXEMPLARY DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OF DOCUMENTS AND THE INFORMATION CONTAINED IN THE DOCUMENTS.

Contents

1. Scope

This specification defines the markup language XHTML Mobile Profile version 1.3 (XHTML Mobile Profile 1.3 or XHTML MP 1.3), a language designed for resource-constrained Web clients. XHTML Mobile Profile 1.3 is targeted at devices such as mobile phones, PDAs, pagers and set-top boxes that do not support the full set of XHTML features. Syntax for XHTML MP 1.3 has converged with syntax for XHMTL Basic 1.1. This specification further clarifies the execution environment in which XHTML MP 1.3 is used and extends the execution environment conformance requirements to foster interoperability.

For the Document Type Definition for XHTML Mobile Profile 1.3 this specification uses the XHTML Basic 1.1 Document Type Defintion. In addition the XHTML Mobile Profile 1.3 specification defines conformance requirements for user agents that process XHTML Mobile Profile 1.3 documents.

2. References

2.1 Normative References

[DOM3Events]	"Document Object Model (DOM) Level 3 Events Specification", Version 1.0, W3C, W3C Working Draft 13 April 2006, URL: http://www.w3.org/TR/2006/WD-DOM-Level-3-Events-20060413/
[ESMP11]	"ECMAScript Mobile Profile", Version 1.1, Open Mobile Alliance™, OMA-TS-ESMP-V1_1, URL:http://www.openmobilealliance.org/
[HTML4]	"HTML 4.01 Specification", W3C, W3C Recommendation 24 December 1999, URL:http://www.w3.org/TR/html401/
[SCRRULES]	"SCR Rules and Procedures", Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL:http://www.openmobilealliance.org/
[OMNA]	"Open Mobile Naming Authority", Open Mobile Alliance™, URL:http://www.openmobilealliance.org/tech/omna/
[RFC2119]	"Key words for use in RFCs to Indicate Requirement Levels", IETF, S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt
[RFC3986]	[RFC3986] "Uniform Resource Identifiers (URI): Generic Syntax", T. Berners-Lee, R. Fielding, L. Masinter, January 2005, URL: <http://www.ietf.org/rfc/rfc3986.txt>
[RFC4329]	"Scripting Media Types", IETF, B. Hoehrman, April 2006, URL:http://www.ietf.org/rfc/rfc4329.txt
[WCSS]	"Wireless CSS", Version 1.2, Open Mobile Alliance™, OMA-TS-WCSS-V1_2, URL:http://www.openmobilealliance.org/
[XHTMLBasic 1.1]	"XHTML™ Basic 1.1", W3C Candidate Recommendation 13th July 2007, Mark Baker et. al, editors, URL: http://www.w3.org/MarkUp/Group/2007/CR-xhtml-basic-20070713/
[XHTMLMod]	"Modularization of XHTML™", W3C, W3C Recommendation 10 April 2001, Murray Altheim et. al, editors, URL:http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/
[XHTML11]	"XHTML™ 1.1 - Module-based XHTML", W3C Recommendation 31 May 2001, Murray Altheim, et. al, editors, URL:http://www.w3.org/TR/2001/REC-xhtml11-20010531/

2.2 Informative References

[JSR118]	"Mobile Information Device Profile 2.0", Java Community Process, JSR-118, URL:http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html
[MAE]	"Mobile Application Environment", Version 2.4, Open Mobile Alliance™, OMA-TS-MAESpec-V2_4, URL:http://www.openmobilealliance.org/
[PushOTA]	"Push Over The Air", Version 2.2, Open Mobile Alliance™., OMA -TS-PushOTA-V2_2. URL:http://www.openmobilealliance.org/
[RME]	"Rich Media Environment", Version 1.0, Open Mobile Alliance™ , with OMA-TS-RME-V1_0, URL:http://www.openmobilealliance.org/
[XHTMLMP12]	"XHTML Mobile Profile", Version 1.2, Open Mobile Alliance ™, OMA-TS-XHTMLMP-V1_2-20070227-C, URL:http://www.openmobilealliance.org/

3. Terminology and Conventions

3.1 Conventions

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

All sections and appendixes, except "Scope" and "Introduction", are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

User	Person who interacts with a user agent to hear, view or otherwise use a resource
User Agent	Any software or device that interprets XHTML Mobile Profile documents and other related resources on behalf of the user

3.3 Abbreviations

CJK	Chinese, Japanese and Korean
CSS	Cascading Style Sheets
DTD	Document Type Definition
ESMP	ECMAScript Mobile Profile
OMA	Open Mobile Alliance
OMNA	Open Mobile Naming Authority
PDA	Personal Digital Assistant
RME	Rich Media Environment
URI	Uniform Resource Identifier
URN	Uniform Resource Name
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language

4. Introduction

This specification defines XHTML Mobile Profile version 1.3 (XHTML Mobile Profile 1.3 or XHTML MP 1.3), a markup language designed for resource-constrained Internet clients. The motivation for XHTML Mobile Profile is to provide an authoring language based upon XHTML that addresses the special requirements of clients operating on resource-constrained devices such as mobile phones, PDAs, pagers and set-top boxes. The specific motivation for XHTML MP 1.3 is to converge XHTML MP with the XHTML Basic 1.1 specification from the W3C whilst adding conformance requirements for user agents that process XHTML MP 1.3 documents.

XHTML Mobile Profile 1.3 is an XHTML document type based upon the module framework and the modules defined by Modularization of XHTML XHTMLMod. XHTML Mobile Profile 1.3 is also an extended subset of XHTML 1.1 **Error! Reference source not found.** based upon XHTML Basic 1.1 [XHTMLBasic 1.1].

XHTML Mobile Profile 1.3 builds upon XHTML Mobile Profile 1.2 [XHTMLMP12]. Work has been going on within OMA and W3C to harmonize XHTML Mobile Profile and XHTML Basic. As the result of this harmonization W3C has issued a new version of XHTML Basic, version 1.1, and XHTML Mobile Profile 1.3 is aligned with XHTML Basic 1.1.

The main differences between XHTML Mobile Profile 1.3 and XHTML Mobile Profile 1.2 are:

- The DTD for XHTML MP 1.3 is identical to the [XHTMLBasic 1.1] document type definition.

- The Target Module from [XHTMLMod] is added.

- Events have been updated to DOM Level 3 [DOM3Events].

This specification is intended to be used to implement Internet clients that support XHTML Mobile Profile 1.3. It can also be used as a guide for authoring content using XHTML Mobile Profile 1.3, but this is not its primary purpose.

5. The XHTML Mobile Profile 1.3 Document Type

The XHTML Mobile Profile 1.3 document type is an XHTML document type based upon the module framework and the modules defined by XHTMLMod.

The XHTML Mobile Profile 1.3 document type is the same as the the XHTML Basic 1.1 document type. See [XHTMLBasic 1.1], section 3. XHTML Mobile Profile 1.3 User Agents SHALL interpret XHTML Mobile Profile 1.3 documents according to the XHTML Basic 1.1 document type [XHTMLBasic 1.1].

The XHTML Mobile Profile 1.3 document type is a strict superset of XHTML Mobile Profile 1.2. It adds the Target Module from [XHTMLMod], through which the "`target`" attribute is added to the area and "link defining elements" (e.g. <a>).

Note: The XHTML Basic 1.1 document type contains the Presentation Module. However, it is recommended that style sheets be used to create a presentation that is appropriate for the device. The Presentation Module is kept for backwards compatibility reasons.

6. Use of XHTML Mobile Profile

This section is informative.

The XHTML Mobile Profile 1.3 document type serves as an authoring language for content targeted at resource-constrained devices. It is expected that it can be used for this purpose without further modification.

The XHTML Mobile Profile 1.3 document type could also serve as a host language, that is, a language containing a mix of XML vocabularies within one document type. See XHTMLMod, section 3.1, "XHTML Host Language Document Type Conformance" and [XHTMLBasic 1.1], section 4, "How to Use XHTML Basic" for more information.

7. Conformance

7.1   Document Conformance

A conforming XHTML Mobile Profile 1.3 document is a document that requires only the facilities described as mandatory in this specification and in the XHTML Basic 1.1 specification [XHTMLBasic 1.1].

A conforming document MUST meet all of the criteria defined in [XHTMLBasic 1.1], section 2.1, "Document Conformance".

7.2   User Agent Conformance

A conforming user agent MUST meet all the user agent conformance requirements defined in **Error! Reference source not found.**, section 2.2.

XHTML Mobile Profile 1.3 documents should be labelled with the Internet Media Type "application/xhtml+xml"according to [XHTMLBasic 1.1], section 2.1, "Document Conformance", last paragraph.

A conforming user agent MUST accept XHTML Mobile Profile 1.3 documents identified as "application/xhtml+xml".

For backwards compatibility with earlier versions of XHTML Mobile Profile the following also applies:

A conforming user agent MUST accept XHTML Mobile Profile documents identified as "application/vnd.wap.xhtml+xml".

A conforming user agent SHOULD accept XHTML Mobile Profile documents identified as "text/html".

Note that the Internet Media Type label does not give enough information to determine which document type it is. The media type "text/html" may be used instead of an XHTML media type. As there are no conformance rules for documents with type "text/html", there is no easy way for the user agent to determine which documents of type "text/html" are XHTML Mobile Profile 1.3 documents, except that the document may include the DOCTYPE declaration specified in section 7.1.

In addition, when the media type states "xhtml" the user agent can use the DOCTYPE declaration in the document to determine if it is an XHTML Mobile Profile 1.3 document or another XHTML family document. Furthermore, if it is an XHTML Mobile Profile document the XHTML Mobile Profile version can be determined. This may be useful as the DTD differs between different XHTML profiles and XHTML Mobile Profile versions. Specifically, the XHTML inputmode Attribute Module,is not included in XHTMLMod but instead explicitly defined in [XHTMLBasic 1.1], which is the document type that is used for XHTML Mobile Profile 1.3 documents.

When declaring support for XHTML Mobile Profile 1.3, a conforming user agent MUST use the following HTTP header fields:

```
Accept: application/xhtml+xml;
profile="http://www.w3.org/TR/xhtml-basic/xhtml-basic11.dtd"

Accept: application/xhtml+xml; profile="http://www.wapforum.org/xhtml"

Accept: application/vnd.wap.xhtml+xml
```

Note that devices are encouraged to use the application/xhtml+xml accept header.

Additional user agent conformance requirements are defined in [MAE]. To fully understand and implement a conforming MAE user agent, this specification must be considered in conjunction with [MAE].

Note: [MAE] requires support for both UTF-8 and UTF-16 independent of MIME type declaration.

8. Use of Style with XHTML Mobile Profile

XHTML Mobile Profile 1.3 supports the use of style to provide authors control of presentation. XHTML Mobile Profile 1.3 user agents SHOULD support style (see [MAE]) and where supported the style language SHALL be [WCSS].

8.1   Adding Style to XHTML Mobile Profile Documents

Style information can be associated with a document in three ways:

- External style sheet

- Internal style sheet

- Inline style information

8.2   External Style Sheets

An external style sheet can be associated with a document using a special XML processing instruction or the `link` element.

The use of the XML processing instruction is specified in [WCSS]. In the following example, the XML processing instruction is used to associate the external style sheet "mobile.css":

```
<?xml-stylesheet href="mobile.css" media="handheld" type="text/css" ?>
```

The use of the `link` element is specified by XHTMLMod. To link an external style sheet to a document using the `link` element, certain values for the `rel` attribute are specified: `rel="stylesheet"` or `rel="alternate stylesheet"`. In either case, the `type` attribute specifies the style sheet language. For example:

```
<link href="mobile.css" type="text/css" rel="stylesheet"/>
```

For `type="text/css"`, the user agent MUST process the style sheet according to the style language Wireless CSS [WCSS].

In the following example, the `link` element is used to associate the external style sheet "mystyle.css":

```
<html>
  <head>
    <link href="mystyle.css" type="text/css" rel="stylesheet"/>
    ...
  </head>
  ...
</html>
```

8.3   Internal Style Sheets

Style information can be located within the document using the `style` element. This element, like `link`, must be located in the document header. The `style` element has the `type` attribute that specifies the style sheet language.

The following shows an example of an internal style sheet:

```
<html>
  <head>
    <style type="text/css">
      p { text-align: center; }
    </style>
    ...
  </head>
  ...
</html>
```

For `type="text/css"`, the user agent MUST process the style sheet according to the style language Wireless CSS [WCSS].

User agents that don't support style sheets, or don't support the specific style sheet language used by a `style` element, MUST hide the content of the `style` element.

8.4   Inline Style

An author can specify style information for a single element using the `style` attribute. This is called inline style. The `style` attribute is part of the Core attribute set and is therefore available on every element in XHTML Mobile Profile 1.3. The default style language for style information in the `style` attribute is Wireless CSS [WCSS].

In the following example, inline styling information is applied to a specific paragraph element:

```
<p style="text-align: center">...</p>
```

Note that not all styling rules apply to all elements, and some elements are completely unaffected by styling rules. See [WCSS] for details.

Note that the Style Attribute Module is deprecated according to [XHTMLBasic 1.1], section 3.

9. Use of Script with XHTML Mobile Profile

The use of script with XHTML Mobile Profile is through the Scripting Module defined by XHTMLMod. The Scripting Module defines elements and attributes used to contain information pertaining to executable scripts. XHTMLMod normatively references [HTML4] for the semantics of these elements and attributes. To promote interoperability, this specification attempts to clarify the processing of the elements and attributes of the Scripting Module, and to tighten requirements. The semantics for these elements is therefore defined by [HTML4] and this specification, with this specification taking precedence over [HTML4] as necessary.

9.1   Script languages in XHTML Mobile Profile

Support for scripting is RECOMMENDED. The exact conformance requirements for the support of script in user agents supporting XHTML Mobile Profile 1.3 may be defined outside this specification, such as the specification mandating ECMAScript Mobile Profile.

The scripting language is identified by the `type` attribute of the `script` element. The value of the `type` attribute specifies the content type of the scripting language as a MIME media type.

When scripting is supported, the XHTML Mobile Profile 1.3 user agent MUST support the scripting language ECMAScript Mobile Profile [ESMP11], the MIME media types for ECMAScript Mobile Profile being `application/ecmascript` or `text/ecmascript`. The user agent MUST also support ECMAScript Mobile Profile scripts identified with media type `text/javascript`.

The use of `application/ecmascript` is the preferred MIME media type, if the script meets the language, character-set, character encoding, and security, requirements defined in [RFC4329]. Scripts identified as `application/ecmascript` or `application/javascript` SHOULD be processed according to the stricter processing rules found in [RFC4329]. The use of `text/ecmascript` or `text/javascript` MAY be deprecated in a future release.

The XHTML Mobile Profile 1.3 user agent MAY support other scripting languages.

9.2   Adding Script to XHTML Mobile Profile Documents

An ECMAScript program, as defined in [ESMP11], is included with an XHTML Mobile Profile 1.3 document using the `script` element. The `script` element defines a script, which is a series of ECMAScript statements. An XHTML Mobile Profile 1.3 document can contain any number of `script` elements, and therefore any number of scripts, but all the scripts are part of the same ECMAScript program. All global function and variable definitions within any one script are available to all scripts in the document.

9.2.1   The `script` Element

A script is inserted into the document using the `script` element. This element can be located in the document header or document body. If placed in the document body, it can be placed anywhere Block or Inline content is allowed. See the DTD found in **Error! Reference source not found.** for details. The `script` element has the `type` attribute that specifies the script language.

Scripts may be defined as the content of the `script` element or in an external document referenced by the `script` element. The location of the script content is determined by the `src` attribute. If the `src` attribute is present, it is used to locate the external document containing the script. If the `src` attribute is not specified, the script is the content of the `script` element.

9.2.2   The `noscript` Element

The `noscript` element is used to describe an alternate presentation for use when the specified scripting language is not supported.

For example, in circumstances when a script language other than [ESMP11] is provided and script is required for the user agent processing XHTML Mobile Profile 1.3 the alternative presentation afforded by the `noscript` element is used.

Note: The exact conformance requirements for the support of script in user agents supporting XHTML Mobile Profile 1.3 may be defined outside this specification, such as the specification mandating ECMAScript Mobile Profile

9.3   Script Execution

Scripts are executed:

- As the document is loaded by the user agent

- When a specific event occurs on a certain element

9.3.1   Script Reference Processing Model

Scripts are specified by the `script` element. If the `src` attribute is specified, it names an external document containing the script. If the `src` attribute is not specified, the script is the content of the `script` element. The processing of the `script` element is specified by XHTMLMod, which references [HTML4] for the details.

9.3.1.1   Processing the script Element

If the user agent supports scripting it MUST process `script` elements as described in this section.

While loading the document, the user agent MUST process all `script` elements encountered in the document. For each `script` element, the user agent MUST determine if the script language specified by the element (using the `type` attribute) is a supported language:

- If the script language is supported, the user agent MUST execute the script specified by the `script` element. It MUST execute all script statements in the order in which they appear in the document containing the script. Function definitions should be treated as a single statement and should only establish the visibility of the function; statements within the function MUST NOT be executed.

- If the script language is not supported, the user agent MUST ignore the `script` element and MUST begin processing `noscript` elements as specified in section 9.3.1.2. The user agent MUST continue to process `script` elements.

The user agent MUST process all `script` elements (i.e. scripts) in the order in which they are found within the document.

The user agent MUST NOT process a `script` element for which it does not recognize the language specified by the `type` attribute.

9.3.1.2   Processing the noscript Element

If the user agent supports scripting and is configured to evaluate scripts:

- If the user agent encounters a `script` element that specifies a scripting language that is not supported, it MUST begin processing all `noscript` elements encountered in the document. The user agent MUST continue to process `script` elements even after it has found one that specifies an unsupported language. The user agent MUST NOT process `noscript` elements until it has encountered an unsupported script.

If the user agent does not support scripting or is configured to not evaluate scripts:

- The user agent MUST ignore `script` elements and MUST render the content of all `noscript` elements.

9.3.2   Event Reference Processing Model

In addition to being executed upon loading of the document, scripts can also be executed as a result of the occurrence of an event. The user agent processes an event by invoking an event handler, if one has been registered for that event. An event handler is a set of statements within a script. In XHTML Mobile Profile, an event handler is bound to an event (registered) using an event handler attribute, e.g. `onclick`. See section 10 for details.

9.4 Examples

This section contains examples of the use of the `script` and `noscript` elements in XHTML Mobile Profile.

9.4.1 Script Executed on Document Loading

The following shows an example of a script executed as part of loading the document:

```
<html>
 <head>
  <title>Script Example: Global Script</title>
  <script type="text/ecmascript">
   function initApp(p1, p2) { ... }
   var g1 = "img1";
   var g2 = "img2";
   initApp(g1, g2);
  </script>
 </head>
 <body>
  ...
 </body>
</html>
```

9.4.2 Script Executed on Form Submission

The following shows an example of a script executed during form submission that is used to validate form input:

```
<html>
 <head>
  <title>Script Example: Form Validation</title>
  <script type="text/ecmascript">
   function validateForm() { ... }
  </script>
 </head>
 <body>
  <form action="..." onsubmit="validateForm();">
   ...
   <input type="submit" value="Submit"/>
  </form>
 </body>
</html>
```

9.4.3 Use of `noscript` Element

The following shows an example of the use of the `noscript` element:

```
<html>
 <head>
  <title>Script Example: Use of noscript Element</title>
  <script type="text/ecmascript">
   function writeDynamicContent() { ... }
   ...
  </script>
 </head>
 <body>
  <p>
   <script type="text/ecmascript">
    writeDynamicContent();
   </script>
   <noscript>
    static content
```

```
    </noscript>
   </p>
  </body>
</html>
```

10. Events in XHTML Mobile Profile

10.1 DOM Level 3 Event Model

The event model for XHTML Mobile Profile is a profile of the event model described by W3C DOM Level 3 Events [DOM3Events], called the DOM Event Model. Specifically, the event capture and event bubbling phases of event dispatching are excluded. These phases were excluded by removing the mechanisms for attaching an event listener (called a handler elsewhere within this specification) to be triggered during these phases. Said another way, the mechanism in XHTML Mobile Profile 1.3 for binding an event listener does not support registration of capturing listeners and bubbling listeners. The event binding mechanism permits an event listener to only be registered on an element when that element is the target node of the event.

The events defined for XHTML Mobile Profile 1.3 are given in section 10.2.

Editors note: DOM Level 3 Events is still in Working Draft status and as such cannot be considered frozen yet. Currently, the above statement is in line with the 13/04/06 Working Draft of DOM Level 3 Events v1.0. Once this specification reaches candidate status, this statement needs to be reviewed again.

10.2 Events and Event Handlers

The following table defines the supported events and the syntax for defining event handlers for those events.

Event	Event Handler Attribute	Applies to	Semantic Description	Support
Load	`onload`	`body`	Occurs when the requested markup document completes loading	M
Unload	`onunload`	`body`	Occurs just prior to a displayed document being removed from view	O
Click	`onclick`	See section 10.3.3 for details.	Occurs when the primary selection mechanism is activated to select a markup element	M
Double Click	`ondblclick`	See section 10.3.4 for details.	Occurs when the primary selection mechanism is activated twice within a short period of time, to select a markup element	O
Mouse Down	`onmousedown`	See section 10.3.14 for details.	Occurs when the pointing device button is pressed while device is over an element	O
Mouse Up	`onmouseup`	See section 10.3.14 for details.	Occurs when the pointing device button is released while device is over an element	O

Mouse Over	`onmouseover`	See section 10.3.14 for details.	Occurs when the pointing device is moved onto an element	O
Mouse Move	`onmousemove`	See section 10.3.14 for details.	Occurs when the pointing device is moved while it is over an element	O
Mouse Out	`onmouseout`	See section 10.3.14 for details.	Occurs when the pointing device is moved away from an element	O
Focus	`onfocus`	`a`, `button`, `label`, `input`, `select`, `textarea`	Occurs when a markup element gains focus	O
Blur	`onblur`	`a`, `button`, `label`, `input`, `select`, `textarea`	Occurs when a markup element loses focus	O
Key Press	`onkeypress`	See section 10.3.7 for details.	Occurs when any one of a defined set of keys is pressed and released	O
Key Down	`onkeydown`	See section 10.3.8 for details.	Occurs when any one of a defined set of keys is pressed down	O
Key Up	`onkeyup`	See section 10.3.9 for details.	Occurs when any one of a defined set of keys is released	O
Submit	`onsubmit`	`form`	Occurs when a submit form control is activated, just prior to the actual form submission	M
Reset	`onreset`	`form`	Occurs when a reset form control is activated, just prior to the actual form reset	M
Select	`onselect`	`input`, `textarea`	Occurs when some text in a text field is selected	O
Change	`onchange`	`input`, `select`, `textarea`	Occurs when a control loses focus and its value has modified since gaining focus	O

10.3 Event Semantics

This section defines the set of events for use with XHTML Mobile Profile. The user agent MUST support all events indicated to be Mandatory. A user agent MAY support an event indicated to be Optional. If a user agent supports an Optional event, it MUST support the event in a manner consistent with this specification.

Unless otherwise noted, if a user agent supports an event, it MUST support the event on all elements indicated for that event.

10.3.1   Load

Event: Load

Binding: `onload`

Support: Mandatory

Description: The Load event occurs after the user agent has completed the loading and parsing of the document. Specifically, it occurs after all referenced and/or embedded objects have been loaded and processed, and after all scripts have been executed.

Target Element: `body`

Cancellable: No

10.3.2   Unload

Event: Unload

Binding: `onunload`

Support: Optional

Description: The Unload event occurs just prior to a displayed document being removed from view.

Target Element: `body`

Cancellable: No

10.3.3   Click

Event: Click

Binding: `onclick`

Support: Mandatory

Description: The Click event occurs when the primary selection mechanism is activated to select a markup element.

Target Elements:

A user agent MUST support the Click event for the following elements: `a`, `button`, `img`, `input`, `object`, `option`, `textarea`. The user agent MAY support Click events for other elements. If it does, it SHOULD support the complete set of target elements.

The complete set of mandatory and optional elements is: `a`, `abbr`, `acronym`, `address`, `b`, `big`, `blockquote`, `body`, `button`, `caption`, `cite`, `code`, `dd`, `dfn`, `div`, `dl`, `dt`, `em`, `fieldset`, `form`, `h1`-`h6`, `hr`, `i`, `img`, `input`, `kbd`, `label`, `li`, `link`, `noscript`, `object`, `ol`, `optgroup`, `option`, `p`, `pre`, `q`, `samp`, `select`, `small`, `span`, `strong`, `table`, `td`, `textarea`, `th`, `tr`, `ul`, `var`.

Cancellable: Yes

10.3.4   Double Click

Event: Double Click

Binding: `ondblclick`

Support: Optional

Description: The Double Click event occurs when the primary selection mechanism is activated twice within a short period of time, to select a markup element. A user agent that supports the Double Click event SHOULD give users the ability to define the minimum time interval between the first and second click that will result in the event being fired.

Target Elements:

If a user agent supports the Double Click event, it MUST support the event for the following elements: `a`, `button`, `img`, `input`, `object`, `option`, `textarea`. The user agent MAY support Double Click events for other elements. If it does, it SHOULD support the complete set of target elements.

The complete set of mandatory and optional elements is: `a`, `abbr`, `acronym`, `address`, `b`, `big`, `blockquote`, `body`, `button`, `caption`, `cite`, `code`, `dd`, `dfn`, `div`, `dl`, `dt`, `em`, `fieldset`, `form`, `h1-h6`, `hr`, `i`, `img`, `input`, `kbd`, `label`, `li`, `link`, `noscript`, `object`, `ol`, `optgroup`, `option`, `p`, `pre`, `q`, `samp`, `select`, `small`, `span`, `strong`, `table`, `td`, `textarea`, `th`, `tr`, `ul`, `var`.

Cancellable: Yes

10.3.5   Focus

Event: Focus

Binding: `onfocus`

Support: Optional

Description: The Focus event occurs when an element receives input focus.

Target Elements: `a`, `button`, `label`, `input`, `select`, `textarea`

Cancellable: No

10.3.6   Blur

Event: Blur

Binding: `onblur`

Support: Optional

Description: The Blur event occurs when an element loses input focus.

Target Elements: `a`, `button`, `label`, `input`, `select`, `textarea`

Cancellable: No

10.3.7   Key Press

Event: Key Press

Binding: `onkeypress`

Support: Optional

Description: The Key Press event occurs whenever a key is pressed and released when an element has been selected. The Key Press event applies only to the selected element, that is, the target of the event is the selected element. The set of keys on a device that trigger this event is not defined by this specification.

Target Elements:

If the Key Press event is supported, the user agent MUST support the event for the following elements: `a`, `button`, `img`, `input`, `object`, `option`, `textarea`. The user agent MAY support Key Press events for other elements. If it does, it SHOULD support the complete set of target elements.

The complete set of mandatory and optional elements is: a, abbr, acronym, address, b, big, `blockquote`, `body`, `button`, `caption`, `cite`, `code`, dd, dfn, div, dl, dt, em, `fieldset`, `form`, h1-h6, hr, i, `img`, `input`, `kbd`, `label`, li, `link`, `noscript`, `object`, ol, `optgroup`, `option`, p, pre, q, `samp`, `select`, small, span, `strong`, `table`, td, `textarea`, th, tr, ul, var.

Cancellable: Yes

10.3.8   Key Down

Event: Key Down

Binding: `onkeydown`

Support: Optional

Description: The Key Down event occurs whenever a key is pressed down when an element has been selected. The Key Down event applies only to the selected element, that is, the target of the event is the selected element. The set of keys on a device that trigger this event is not defined by this specification.

Target Elements:

If the Key Down event is supported, the user agent MUST support the event for the following elements: `a`, `button`, `img`, `input`, `object`, `option`, `textarea`. The user agent MAY support Key Down events for other elements. If it does, it SHOULD support the complete set of target elements.

The complete set of mandatory and optional elements is: a, abbr, acronym, address, b, big, `blockquote`, `body`, `button`, `caption`, `cite`, `code`, dd, dfn, div, dl, dt, em, `fieldset`, `form`, h1-h6, hr, i, `img`, `input`, `kbd`, `label`, li, `link`, `noscript`, `object`, ol, `optgroup`, `option`, p, pre, q, `samp`, `select`, small, span, `strong`, `table`, td, `textarea`, th, tr, ul, var.

Cancellable: Yes

10.3.9   Key Up

Event: Key Up

Binding: `onkeyup`

Support: Optional

Description: The Key Up event occurs whenever a key is released when an element has been selected. The Key Up event applies only to the selected element, that is, the target of the event is the selected element. The set of keys on a device that trigger this event is not defined by this specification.

Target Elements:

A user agent that supports the Key Up event MUST support the event for the following elements: a, `button`, `img`, `input`, `object`, `option`, `textarea`. The user agent MAY support Key Up events for other elements. If it does, it SHOULD support the complete set of target elements.

The complete set of mandatory and optional elements is: a, abbr, acronym, address, b, big, `blockquote`, `body`, `button`, `caption`, `cite`, `code`, dd, dfn, div, dl, dt, em, `fieldset`, `form`, h1-h6, hr, i, `img`, `input`, `kbd`, `label`, li, `link`, `noscript`, `object`, ol, `optgroup`, `option`, p, pre, q, `samp`, `select`, small, span, `strong`, `table`, td, `textarea`, th, tr, ul, var.

Cancellable: Yes

10.3.10 Submit

Event: Submit

Binding: `onsubmit`

Support: Mandatory

Description: The Submit event occurs when a submit form control has been activated, just prior to the actual submission of the form data.

Target Element: `form`

Cancellable: Yes

10.3.11 Reset

Event: Reset

Binding: `onreset`

Support: Mandatory

Description: The Reset event occurs when a reset form control is activated, just prior to the actual resetting of the form control data.

Target Element: `form`

Cancellable: Yes

10.3.12 Select

Event: Select

Binding: `onselect`

Support: Optional

Description: The Select event occurs when the user selects text in a text input form control.

Target Elements: `input`, `textarea`

Cancellable: No

10.3.13 Change

Event: Change

Binding: `onchange`

Support: Optional

Description: The Change event occurs when a form control loses input focus and its value has been modified since gaining focus.

Target Elements: `input`, `select`, `textarea`

Cancellable: No

10.3.14 Mouse Events

Event: Mouse Down, Mouse Up, Mouse Over, Mouse Move, Mouse Out

Binding: `onmousedown, onmouseup, onmouseover, onmousemove, onmouseout`

Support: Optional

Description: Use of the five mouse-related events should be limited to those devices that support mouse input or some form of pointing input device.

Target Elements:

> The set of elements to which the mouse events apply is: `a, abbr, acronym, address, b, big, blockquote, body, button, caption, cite, code, dd, dfn, div, dl, dt, em, fieldset, form, h1-h6, hr, i, img, input, kbd, label, li, link, noscript, object, ol, optgroup, option, p, pre, q, samp, select, small, span, strong, table, td, textarea, th, tr, ul, var.`

Cancellable: Yes

10.4 Event Handler Registration

As mentioned in section 10.1, the event model for XHTML Mobile Profile is a profile of the DOM Event Model. The DOM Event Model specifies a mechanism for event handler registration using interfaces and methods invoked on those interfaces. Specifically, each node in the document supports the `EventTarget` interface, allowing registration of any event handler on that node. To register an event handler, a program or script invokes the `addEventListener()` method from the `EventTarget` interface on a given element.

```
interface EventTarget {
  void addEventListener(in DOMString type, in EventListener listener,
   in boolean useCapture);
  ...
}
```

See [DOM3Events] for complete details.

The event model for XHTML Mobile Profile 1.3 uses the DOM Event Model as the reference model, but does not directly support event handler registration using the interfaces of [DOM3Events]. Instead, an event handler is registered for an event by assigning a value to an *event handler attribute*. The set of all event handler attributes is given by the table in section 10.2. The user agent MUST map the event handler attribute and its value into the DOM Event Model as follows:

- The event handler attribute specifies the type of the event.

- The event handler attribute value specifies the handler. The handler is a series of zero or more script statements. The user agent MUST operate as if the value of the event handler attribute is the body of an anonymous function that is of type `EventListener`. The prototype of the function contains a single argument, `event`, which is of type `Event` object as defined in ECMAScript Mobile Profile **Error! Reference source not found.**[ESMP11]. The `Event` object is in scope for all statements of the event handler:

```
function (Event event)
{
  value of the event handler attribute
}
```

- The event's target element is the element to which the event handler attribute is attached.

To complete the registration, the user agent MUST perform an operation that is equivalent to invoking the registration method `addEventListener()` on the element (the `EventTarget`) to which the attribute is attached, passing the event type as determined above as the argument `type`, the anonymous function reference as the argument `listener`, with the argument `useCapture` specified as `false`.

Any attempt to modify the value of an event handler attribute via the DOM interfaces MUST result in the deregistration of any existing event handler for that event type on the given element (i.e. invocation of the method `removeEventListener()`). Only a single event handler SHALL be supported for a given event type on a given target element.

Editors note: This section currently conforms to the Working Draft of DOM Level 3 Events v1.0. This section needs to be reviewed when DOM Level 3 Events becomes a candidate recommendation in the W3C.

10.5 Event Cancellation

Only certain events are cancellable. The list of cancellable events is:

- DOMActivate
- textInput
- Click
- mousedown
- mouseup

- mouseover

- mousemove

- mouseout

- keydown

- keyup

- Reset

- Submit

For a cancellable event, if an event handler cancels the event by returning `false`, the user agent MUST NOT execute the default action for that event. If the event handler returns `true` or does not return a value, the user agent MUST execute the default action for that event.

For a non-cancellable event, the user agent MUST ignore any return value by the event handler.

Editors note: This section currently conforms to the Working Draft of DOM Level 3 Events v1.0. This section needs to be reviewed when DOM Level 3 Events becomes a candidate recommendation in the W3C.

10.6 Examples

This section contains examples of the use of event handlers in XHTML Mobile Profile 1.3.

10.6.1 Event Handler Function

The following shows an example of an inline event handler:

```
<html>
 <head>
  <title>Script Example: Event Handler Function</title>
  <script type="text/ecmascript">
   function myHandler() {
    // ECMAScript code goes here
   }
  </script>
 </head>
 <body>
  <p><a onclick="myHandler();">Custom hyperlink</a></p>
 </body>
</html>
```

10.6.2 Event Handler Function with Arguments

The following shows an example of an inline event handler:

```
<html>
 <head>
  <title>Script Example: Event Handler Function with Arguments</title>
  <script type="text/ecmascript">
   function myHandler(x) {
    // ECMAScript code goes here
   }
  </script>
 </head>
 <body>
  <p><a onclick="var x=5; myHandler(x);">Custom hyperlink</a></p>
 </body>
</html>
```

10.6.3 Event Handler Using the Event Object

The following shows an example of an inline event handler:

```
<html>
 <head>
  <title>Script Example: Use of Event Object</title>
  <script type="text/ecmascript">
   function myHandler(evt) {
    // ECMAScript code goes here
   }
  </script>
 </head>
 <body>
  <p><a onclick="myHandler(event);">Custom hyperlink</a></p>
 </body>
</html>
```

10.6.4   Inline Event Handler

The following shows an example of an inline event handler:

```
<html>
 <head>
  <title>Script Example: Inline Event Handler</title>
 </head>
 <body>
  <p><a onclick="history.go(1);">Forward</a></p>
 </body>
</html>
```

11. Media Object Inclusion: the `object` Element

11.1 The `object` Element

The `object` element is defined in XHTML as a generic inclusion element for any type of content. When the `object` element is used to include an object into the XHTML page, two sets of rules are applied:

>   i.   generic inclusion rules, defined in the HTML specification [HTML4], Section 13.3, and

>   ii.   XHTML Mobile Profile 1.3 specific rules, defined in this document.

The user agent MUST conform to the generic rendering rules for the `object` element, as defined in [HTML4], Section 13.3. The generic rules cover such things as the user agent fallback behavior, when the `object` element includes an unsupported or unknown content type, and use of attributes.

According to the HTML specification an `object` element refers to either:

- The location (URI) of the object's data, for example, a GIF image, indicated by the `data` attribute, OR

- The location (URI) of an object's implementation, i.e. the location of the object's executable code, indicated by the `classid` attribute.

This specification covers the two following ways to use the `data` and the `classid` attributes:

- The `data` attribute is used by the content author to specify the location (URI) of the object's data, for example, a GIF image.

- The `classid` attribute is used by the content author to specify the location of a local application in the device that handles media objects. Examples of local applications are a media player application and a (Java) Application Management System.

If both the `data` attribute and the `classid` attribute are present, `classid` MUST take precedence over `data`.

Other ways to use the `data` and the `classid` attributes are implementation dependent.

11.2 Referring to an Object's Data

When the `data` attribute is present the user agent MUST use the value (URL) of this attribute as the location of the object's data. The local application for handling the object's data is determined by the MIME media type (e.g. "image/gif") specified by the `type` attribute. If this attribute is missing, the user agent may use any heuristics to determine the local application (e.g. look at the object's data when it is downloaded or use the file extension).

When initial parameters are defined with the `param` element the user agent MUST pass these parameters along to the local application that renders the object's data.

11.2.1 Examples

11.2.1.1   SVG Animation

In the following example an SVG animation is displayed. A user agent that does not support SVG images renders the GIF image instead.

```
<object type="image/svg+xml" data="anImage.svg">
 <object type="image/gif" data="anImage.gif"/>
</object>
```

11.2.1.2   Audio File

In the following example an audio file is played in the background, while the user is viewing the page:

```
<object data="aSound.mid" type="audio/mid"/>
```

11.2.1.3 RME

In the following example an [RME] scene is displayed:

```
<object data="myRME.svg" Type="image/svg+xml" width="240" height="120"/>
```

Note: In response to the Object call, the client is likely to receive a multipart/related MIME packaged file set.

11.2.1.4 Macromedia Flash

In the following example a Macromedia Flash™ movie is shown. Initial parameters are passed to the Macromedia Flash™ application. The `height` and `width` attributes reserve space for the movie in the page:

```
<object data="myflash.swf" type="application/x-shockwave-flash" width="240"
  height="120">
 <param name="quality" value="high"/>
 <param name="bgcolor" value="#ffffff"/>
</object>
```

11.2.1.5 Java AMS

In the following example the user agent downloads a Java Application Descriptor file that will invoke a (Java) Application Management System (AMS) application to handle the file:

```
<object data="aMIDlet.jad" type="application/vnd.sun.jad"/>
```

11.3 Referring to an Object's Implementation

It is also possible to specify the location (URI) of an object's implementation, using the `classid` attribute. A Uniform Resource Name (URN) may be used by an author to identify a local application in the device that handles media objects. For XHTML Mobile Profile 1.3, the PUSH Application ID, see [PushOTA], is used to identify the local application even though this does not predicate (require) the support for PUSH itself. Developers of content handlers (plug-ins) are encouraged to register an application ID for their application as defined by [OMNA].

When the `classid` attribute is present and its value is a URN according to an OMNA [OMNA] registered PUSH Application ID, the user agent MUST use this URN to identify a local application.

When initial parameters are defined with the `param` element the user agent MUST pass these parameters along to the local application.

11.4 Declared Objects

[HTML4], Section 13.3.4, describes how it is possible to separate the declaration of an object from its instantiation. The boolean `declare` attribute can be used to declare an object so that it is not executed when read by the user agent. At the same time, authors must identify the declaration by setting the `id` attribute in the `object` element to a value unique within the current document. Instantiations of the object will refer to this identifier.

An object defined with the `declare` attribute is instantiated every time an element that refers to that object requires it to be rendered. An object is instantiated when that object is the target of a hyperlink navigation. The URI of the hyperlink must be a URI reference as defined in [RFC3986][RFC3986][RFC3986][RFC3986][RFC3986] [RFC3986]. The URI reference may contain an absolute URI or relative URI, but it must contain the '#' character followed by a fragment identifier. The fragment identifier is the `id` attribute value of the object.

To support declared objects, the processing model for a hyperlink is modified as follows:

- If the value of the `href` attribute is a URI reference that includes a fragment identifier and specifies a location within the current document, the user agent searches for a declared object element with the specified ID. A match is declared if the `declare` attribute is specified and the value of the fragment identifier of the URI reference string matches the `id` attribute of the `object` element. The string match is case-sensitive.

- If a matching declared object is found, the user agent MUST create a new page showing only that object and push that page onto the navigation history stack.

11.4.1   Examples

In the following markup, activating the hyperlink with ID "h1" would instantiate the object with ID "obj1":

```
<object id="obj1" declare="declare" data="...">...</object>
<a id="h1" href="#obj1">Instantiate Object 1</a>
```

11.5   Inline vs. External Rendering

According to [HTML4] the `object` element is for *generic object inclusion*. This implies that the intention is that objects included into the page with the `object` element should be rendered inline within the page. However, mobile devices often have limited user interface capabilities and it may not be possible for a mobile browser to render all types of objects inline. For example, for Java MIDlets, inline rendering may be difficult to achieve in a limited mobile device.

The user agent SHOULD render objects included with the `object` element inline in the document flow. However, if the user agent is not able to render an object inline the object may instead be rendered in a separate context.

11.6   Specific Rules for Certain Local Applications

11.6.1   Java Application Management System

This section is informative.

[JSR118], Section 2, describes how Java MIDlet suites can be downloaded, installed, updated, invoked and removed by a software application in the device that is called "Application Management System" (AMS). Another commonly used name for such an application is "Java Application Manager" (JAM).

When the `classid` attribute is "x-oma-application:java-ams" (registered with OMNA) it indicates that the object is an AMS (JAM) application. It is possible to pass parameters to the AMS application, which can use these parameters to determine whether the MIDlet already is installed into the device or not. If the MIDlet is not installed the user could be given the option to download it. This is advantageous compared to directly downloading a JAD or JAR file with a link from the anchor element or with the `object` element using the `data` attribute, as the downloading in these cases is always performed (even if the MIDlet already is installed into the device).

The author must specify the following initial parameters using the `param` element:

- "AMS-Filename": The value is a URL to the MIDlet to launch. If a relative URL is used and the `codebase` attribute is not specified the default codebase value is the base URI of the current document.

- MIDlet attributes, such as "MIDlet-Name", "MIDlet-Version", and "MIDlet-Vendor": Used by the AMS to determine whether the MIDlet already is installed on the device or not. See [JSR118], Section 12. (Note: MIDlet attributes in the JAD or JAR file always override MIDlet attributes specified with the param element. The purpose of the MIDlet attributes specified with the `param` element is only to initially determine whether the MIDlet is installed on the device or not.)

- "AMS-Startup": Makes it possible for the content author to configure the start-up process for the MIDlet. Values are:
  - "auto": The MIDlet is launched directly if it is already installed on the device. If the MIDlet is not installed on the device it is downloaded and installed according to [JSR118].
  - "launch-only": The MIDlet is launched only if it is already installed on the device.
  - "download-confirm": The MIDlet is downloaded and installed, unless it already is installed on the device, and the user is given a chance to choose whether to launch the MIDlet directly or later.

- Start-up parameters to the MIDlet defined by "AMS-Filename".

11.6.2   Examples

The following example shows the use of a hyperlink to activate a Java application:

```
<object id="theJavaApp" declare="declare" classid="x-oma-application:java-
ams">
```

```
 <param name="AMS-Filename" value="aMIDlet.jad"/>
 <param name="MIDlet-Name" value="The MIDlet"/>
 <param name="MIDlet-version" value="2.0"/>
 <param name="MIDlet-Vendor" value="MIDlet Ltd"/>
 <param name="AMS-Startup" value="download-confirm"/>
 Java MIDlets are not supported
</object>
...
<a href="#theJavaApp">Start Java application</a>
```

12. Navigation Optimizations

This section defines requirements on the user agent with regards to accessibility – features that make it easier for the user to access the content.

12.1 Access Keys

An access key binding, defined by the `accesskey` attribute, is used to assign a key to a link or form control – a one-click shortcut. When the user presses the key, the link or form control will be activated. When such alternative ways of accessing a link in the page are available, the user agent SHOULD provide a presentational hint about this to the user.

12.2 Links

A user agent that provides access to the links in the page within a menu SHOULD use the `title` attribute of the anchor (`<a>`) element as the link title, in order to make it easier for the user to access the link.

12.3 Navigation Menu

The user agent SHOULD provide access to linked resources, defined by the `link` element, of the following link types: "start", "next", and "prev". The link types are specified in the `rel` attribute, and defined in [HTML4]. The user accesses the links through a navigation menu or other user interface construct.

The benefits of the navigation menu are:

- It does not take space from the actual content. The links can be presented in a separate menu that is always available when the user scrolls up and down in the page.

- It can be integrated into the general user interface of the user agent, so the user can find navigation options such as Next, Prev, and Home where it is most logical in the user interface.

- It separates structural navigation from other links. The site navigation links are a kind of "meta-link" since they do not really belong to the content; they are used by the user to access the content.

Here is an example that shows how the link types can be used to define links to the start page, the next page, and the previous page in a collection of pages:

```
<head>
 <link rel="start" href="index.htm"/>
 <link rel="next" href="news-item-003.htm"/>
 <link rel="prev" href="news-item-001.htm"/>
</head>
```

The user agent MAY preload linked resources, in order to improve the response time, should the user select the link.

The navigation menu is created by `link` elements specified by the content author inside the XHTML document. The menu is independent of any navigation history that the user agent may have in addition, such as a Back option to navigate to the previously viewed page.

Content authors should not rely solely on the navigation menu to provide access to the linked resources. The links are ignored by user agents that do not support a navigation menu.

13. XHTML `inputmode` Attribute Module

This section describes the XHTML `inputmode` Attribute Module for XHTML Mobile Profile 1.3.

13.1 Overview

Text input modes are hints to the user agent to assist it in selecting the appropriate input mode for the text input expected in a particular text input form control. The XHTML `inputmode` Attribute Module defines the `inputmode` attribute. This attribute enables the content author to specify the default input mode for text input controls, for example, the entry of numeric digits. This default, or initial, input mode acts only as an aid to the user; the user is always free to change the input mode using a browser- or device-specific mechanism.

Note that the XHTML `inputmode` Attribute Module is not included in XHTMLMod but defined in [XHTMLBasic 1.1].

13.2 XHTML inputmode Attribute Module Definition

The XHTML `inputmode` Attribute Module is designed to operate within the framework of XHTMLMod. Text input modes are specified using the `inputmode` attribute on the text input form control elements `input` and `textarea`. The syntax and semantics of the `inputmode` attribute are defined in [XHTMLBasic 1.1], section 5.

13.3 Content Authoring Guidelines

Text input modes are intended to enhance usability by assisting the user agent in selecting the most appropriate input mode for a given text input form control. However, not all modes will be supported by all user agents.

13.3.1 General Guidelines

The following guidelines are provided to content authors:

- Use of multiple input mode tokens in a single XHTML Mobile Profile 1.3 document is discouraged. User agents will not perform any consistency checking or conflict resolution for any mismatches among character set (e.g. charset attribute in Content-Type HTTP header), language (e.g. `xml:lang` attribute) and input mode tokens.

- Use of both the (deprecated) WCSS text input property `-wap-input-format` and the `inputmode` attribute in a single XHTML Mobile Profile 1.3 document is discouraged. Note that, if both are applied to a text input form control, the styling property takes precedence, according to the CSS cascade rules.

13.3.2 CJK Guidelines

The following guideline is provided to authors of CJK content:

- It is common for implementations of XHTML input form control type `password` to provide "peek protection" to ensure the security of sensitive information. In CJK implementations, it may be easier for end users if `digits` mode is used for `type="password"`, because input modes other than numeric digits involve a language preprocessor that makes the peek protection difficult. Authors are cautioned that text input modes other than `digits` for password text fields may be ignored by CJK implementations.

13.4 Examples

13.4.1 Telephone Number

The following example shows how to configure the default input mode for a text field where the user is expected to enter a telephone number:

```
<input type="text" name="tel" inputmode="digits"/>
```

13.4.2   E-mail Address

The following example shows how to configure the default input mode for a text field where the user is expected to enter an e-mail address:

```
<input type="text" name="email" inputmode="latin lowerCase"/>
```

14. Minimum Processing Capacity for XHTML Elements

14.1 Overview

This section is informative.

The XHTML user agent should guarantee the minimum processing capacity for interoperable content authoring, a minimum number of a) option elements per select element, b) input elements, c) textarea elements, d) anchor elements.

The user agent should fail gracefully when the supported file size was exceeded.

14.2 Conformance

This section is normative.

Conformance: In isolation a mobile web browser MUST be able to support at least 30 XHTML Form "Option" elements per "Select" element (See Note) (MAESpec-MINCAP-C-001)

Conformance: In isolation a mobile web browser MUST be able to support at least 30 XHTML Form "Input (Radio and multiple Radio)" elements (See Note) (MAESpec-MINCAP-C-002)

Conformance:] In isolation a mobile web browser MUST be able to support at least 30 XHTML Form "Text area" elements (See Note) (MAESpec-MINCAP-C-003)

Conformance: In isolation a mobile web browser MUST be able to support at least 100 Anchor elements (See Note) (MAESpec-MINCAP-C-004)

Conformance: In isolation the "Text area" of a mobile web browser MUST be able to support at least 512 bytes (MAESpec-MINCAP-C-005)

Conformance: When the size of an applied XHTML file exceeds that of the supported XHTML file size on a mobile web browser then the mobile web browser MUST fail gracefully (e.g. the mobile web browser is recoverable without end-user intervention) (MAESpec-MINCAP-C-006)

NOTE. These requirements represent generic support of the features on a per document basis and are not intended to imply a specific User Experience. For example, the support of 100 Anchors does not imply the support of 100 Anchors in the viewable screen.

15. Using Schemes in XHTML Forms

The use of XHTML forms to build URLs for schemes is well specified for only 3 cases:

- http: (responses between client and server),
- multipart-mime (file upload), and
- plain-text.

Processing rules for the generalized use of forms to build URIs in other cases is ill-defined, and has led to non-interoperable implementations on the web. This section describes a set of processing rules for the construction of url-encoded URIs from the combination of form elements, and gives an illustrative example.

15.1 Processing Rules

This section is normative.

These rules assume that within the target **<form>** element, that **enctype="application/x-www-form-urlencoded"**, which is the default.

Submitting a form (clicking the submit button) MUST generate a URI which is in standard urlencoded form.

URIs, that are to meet standard URL formatting are constructed according to the following rules:

1. Starting with an empty string, append the contents of the **<form> action** attribute value to the string, being sure to treat the string as a url-encoded string (i.e. spaces should be there as %20 encoded.). If there are no **<form> input** or **textarea** statements within the **<form>** then construction is complete.

2. Scan the string for the presence of the "?" (question mark) delimiter. If not present, append it to the string. Note: there is no requirement to validate the syntax of the **action** string.

3. In the order that they appear in a target **<form>**, evaluate **input** and **textarea** elements for appending to the URI string. hname/hvalue pairs are constructed by assigning the **name** attribute as the hname, and **value** as the hvalue, separated by an "=" (equal sign).

4. Append the name=value string to the URL string, separating the existing string from the new substring with either the existing "?" (question mark), or by prepending an "&" (ampersand) to the URL string. Note that values MUST follow existing encoding requirements for x-www-form-urlencoded syntax [HTML4 section 17.13.4]. For example, all spaces will be translated into a "+" (plus sign).

5. Repeat steps #3 and #4 for all **input** and **textarea** elements within the form.

```
<form type="get" enctype="application/x-www-form-urlencoded"
action="U:[H][?][S]" >
[<input type="text" name="N" value="V" />]      // 0 or more input elements
…..
        [<textarea name="TN" value="TV" ></textarea>]  // 0 or more
textareas
</form>
```

The above **<form>** will generate a URL of the form:

$$U:[H][?][S][\&N_1=V_1][\&N_2=V_2][\&N_n=V_n][\&TN_1=TV_1][\&TN_n=TV_n]$$

"?" is required whenever there are string contents that follow.

"&" is required to separate name/value pairs from each other.

No statement is made when there is duplication of names in name/value pairs. Generally this should be avoided.

15.2 Example and Use

This section is informative.

The following is a typical example of using a form to gather information that is to be sent as an email. This example "hardwires" certain inputs (i.e. "cc"), and generates input forms for other inputs (i.e. "bcc"), some with start values.

```
<html>
<head></head>
<body>
<form type="get"
action="mailto:me@you.com?&amp;cc=ccdude@nowhere.com&amp;body=This%20is%20th
e%20body%20of%20a%20hybrid%20message."  enctype="application/x-www-form-
urlencoded">
        To:  <input type="text" name="to" value=" "/><br/>
        Cc: ccdude@nowhere.com        (fixed cc – in action string)<br/>
        Bcc:<input type="text" name="bcc"value="hybrid@where.org"/><br/>
        From:<input type="text" name="from"value="Bad@nowhere.org" ><br/>
        Subject:<input type="text" name="subject" value="This is the subject
too"/><br/>
        Body: This is the body of a hybrid message.    (fixed– in action
string)<br/>
        <input type="submit" value="Submit to Email"/>
</form>
</body>
</html>
```
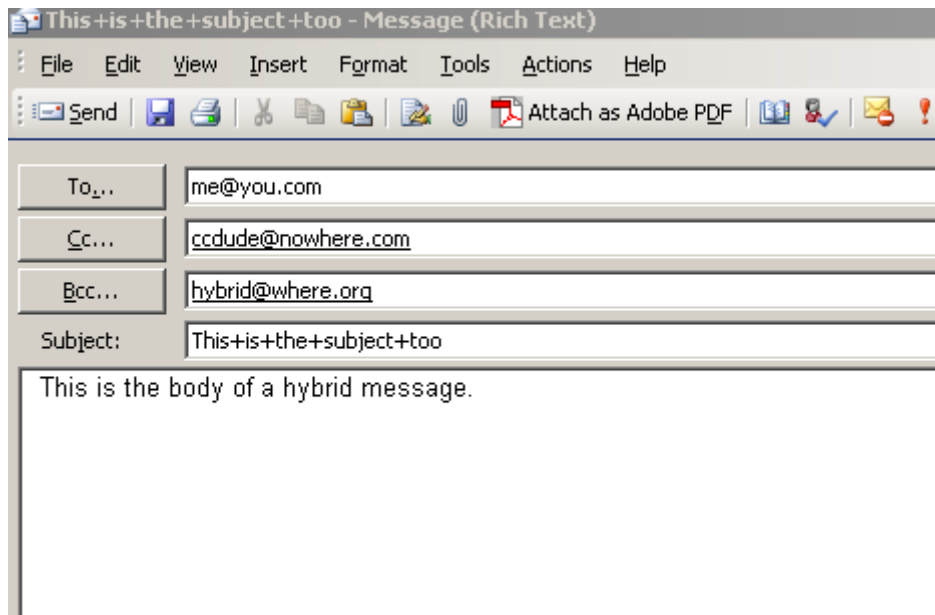
The above XHTML document should generate a presentation similar to what is found in Figure 1.

Figure 1 - XHTML Presentation

When the "Submit to Email" button is pressed the form should generate the URL string:

```
mailto:me@you.com?&cc=ccdude@nowhere.com&body=This%20is%20the%20body%20of%20a%20hybrid%20message.&to=
+&bcc=hybrid@where.org&from=Bad@nowhere.org&subject=This+is+the+subject+too
```

Submitting this URL to an email user agent should, depending upon the agent, generate something similar to 2.

Figure 2 - Sample Email User agent presentation

A number of things should be noted. First, all values submitted through the form are subject to form urlencoding. This means that all spaces are converted to "+" signs (as seen in the subject). Secondly, the action string must be pre-encoded escaping any required characters below 0x7F, most notably the space character. Thirdly, this example correctly ignores the "from" field, even though it is not shown.

Appendix A. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [SCRRULES].

XHTML Basic Modules

Item	Function	Reference	Status	Requirement
XHTMLMP- XHTMLMOD-C-001	XHTML Structure module	5	M	
XHTMLMP- XHTMLMOD-C-002	XHTML Text module	5	M	
XHTMLMP- XHTMLMOD-C-003	XHTML Hypertext module	5	M	
XHTMLMP- XHTMLMOD-C-004	XHTML List module	5	M	
XHTMLMP- XHTMLMOD-C-025	XHTML Forms module	5	M	
XHTMLMP- XHTMLMOD-C-006	XHTML Basic Tables module	5	M	
XHTMLMP- XHTMLMOD-C-007	XHTML Image module	5	M	
XHTMLMP- XHTMLMOD-C-008	XHTML Object module	5	M	
XHTMLMP- XHTMLMOD-C-009	XHTML Metainformation module	5	M	
XHTMLMP- XHTMLMOD-C-010	XHTML Link module	5	M	
XHTMLMP- XHTMLMOD-C-011	XHTML Base module	5	M	
XHTMLMP- XHTMLMOD-C-012	XHTML Intrinsic Events module	5	M	
XHTMLMP- XHTMLMOD-C-012	XHTML Style Sheet module	5	M	
XHTMLMP- XHTMLMOD-C-013	XHTML Style Attribute module	5	M	
XHTMLMP- XHTMLMOD-C-024	XHTML Scripting module	5	M	
XHTMLMP- XHTMLMOD-C-013	XHTML Presentation module	5	M	
XHTMLMP- XHTMLMOD-C-014	XHTML Target module	5	M	
XHTMLMP- XHTMLMOD-C-015	XHTML inputmode Attribute module	5	M	

Other XHTML Basic Elements and Attributes

Item	Function	Reference	Status	Requirement
XHTMLMP- XHTMLMOD-C-016	`value` attribute on the `li` element of the XHTML List module	5	M	

Note: Since the *start* attribute on "ol" is no longer supported, page overflow numbering using the *start* attribute will begin at index 1 on each new page. To accomplish multipage indexing, content developers should use the **value** attribute on the "li" element.

XHTML Document Conformance

Item	Function	Reference	Status	Requirement
XHTMLMP-XHTMLDC-C-001	Document conformant to XHTMLMP Basic 1.1	7.1	M	

XHTML User Agent Conformance

Item	Function	Reference	Status	Requirement
XHTMLMP-XHTMLUA-C-001	User agent meets "Modularization of XHTML" XHTMLMod conformance requirements	7.2	M	

Document Types

Item	Function	Reference	Status	Requirement
XHTMLMP-DOC-C-001	Accept XHTML Mobile Profile documents identified as "application/vnd.wap.xhtml+xml"	7.2	M	
XHTMLMP-DOC-C-002	Accept XHTML Mobile Profile documents identified as "application/xhtml+xml"	7.2	M	
XHTMLMP-DOC-C-003	Accept XHTML Mobile Profile documents identified as "text/html"	7.2	O	
XHTMLMP-DOC-C-003	Declare support for XHTML Mobile Profile documents	7.2	M	

Style Sheets

Item	Function	Reference	Status	Requirement
XHTMLMP-STYLE-C-001	Support for WCSS	8	O	WCSS:MCF AND XHTMLMP-STYLE-C-002 AND XHTMLMP-STYLE-C-003 AND XHTMLMP-STYLE-C-004
XHTMLMP-STYLE-C-002	Handling of type "text/css" for external style sheet	8.2	O	
XHTMLMP-STYLE-C-003	Handling of type "text/css" for internal style sheet	8.3	O	
XHTMLMP-STYLE-C-004	Default type "text/css" for inline style rules	8.4	O	

Note that WCSS support is defined as mandatory in the MAE specification [MAE]. See section 7.2 for specification relationships.

Scripting

Item	Function	Reference	Status	Requirement
XHTMLMP-SCRIPT-C-001	Support for scripting	9	O	XHTMLMP-SCRIPT-C-002 AND XHTMLMP-EVENT-C-001
XHTMLMP-SCRIPT-C-002	Support for ESMP	9.1	O	ESMP:MCF AND XHTMLMP-SCRIPT-C-003 AND XHTMLMP-SCRIPT-C-006 AND XHTMLMP-SCRIPT-C-007
XHTMLMP-SCRIPT-C-003	Script reference processing model	9.3.1	O	XHTMLMP-SCRIPT-C-004 AND XHTMLMP-SCRIPT-C-005
XHTMLMP-SCRIPT-C-004	Processing `script` element	9.3.1.1	O	
XHTMLMP-SCRIPT-C-005	Processing `noscript` element	9.3.1.2	O	
XHTMLMP-SCRIPT-C-006	Accepts ESMP with MIME media type text/ecmascript	9.1	O	
XHTMLMP-SCRIPT-C-007	Accepts ESMP with MIME media type text/javascript	9.1	O	
XHTMLMP-SCRIPT-C-008	Support for other scripting languages	9.1	O	

Note that scripting support is defined as mandatory in the MAE specification [MAE]. See section 7.2 for specification relationships.

Events

Item	Function	Reference	Status	Requirement
XHTMLMP-EVENT-C-001	Support for XHTML Mobile Profile event model	10.1	O	XHTMLMP-EVENT-C-002 AND XHTMLMP-EVENT-C-006 AND XHTMLMP-EVENT-C-035 AND XHTMLMP-EVENT-C-037 AND XHTMLMP-EVENT-C-043 AND XHTMLMP-EVENT-C-044 AND XHTMLMP-EVENT-C-045 AND XHTMLMP-EVENT-C-046

Item	Function	Reference	Status	Requirement
XHTMLMP-EVENT-C-002	Load event	10.3.1	O	XHTMLMP-EVENT-C-003
XHTMLMP-EVENT-C-003	Support for Load event on body element	10.3.1	O	
XHTMLMP-EVENT-C-004	Unload event	10.3.2	O	XHTMLMP-EVENT-C-005
XHTMLMP-EVENT-C-005	Support for Unload event on body element	10.3.2	O	
XHTMLMP-EVENT-C-006	Click event	10.3.3	O	XHTMLMP-EVENT-C-007
XHTMLMP-EVENT-C-007	Support for Click event on mandatory elements	10.3.3	O	
XHTMLMP-EVENT-C-008	Support for Click event on all specified elements	10.3.3	O	
XHTMLMP-EVENT-C-009	Double Click event	10.3.4	O	XHTMLMP-EVENT-C-010
XHTMLMP-EVENT-C-010	Support for Double Click event on mandatory elements	10.3.4	O	
XHTMLMP-EVENT-C-011	Support for Double Click event on all specified elements	10.3.4	O	
XHTMLMP-EVENT-C-012	Mouse Down event	10.3.14	O	XHTMLMP-EVENT-C-013
XHTMLMP-EVENT-C-013	Support for Mouse Down event on all specified elements	10.3.14	O	
XHTMLMP-EVENT-C-014	Mouse Up	10.3.14	O	XHTMLMP-EVENT-C-015
XHTMLMP-EVENT-C-015	Support for Mouse Up event on all specified elements	10.3.14	O	
XHTMLMP-EVENT-C-016	Mouse Over event	10.3.14	O	XHTMLMP-EVENT-C-017
XHTMLMP-EVENT-C-017	Support for Mouse Over event on all specified elements	10.3.14	O	
XHTMLMP-EVENT-C-018	Mouse Move event	10.3.14	O	XHTMLMP-EVENT-C-019
XHTMLMP-EVENT-C-019	Support for Mouse Move event on all specified elements	10.3.14	O	
XHTMLMP-EVENT-C-020	Mouse Out event	10.3.14	O	XHTMLMP-EVENT-C-021
XHTMLMP-EVENT-C-021	Support for Mouse Out event on all specified elements	10.3.14	O	
XHTMLMP-EVENT-C-022	Focus event	10.3.5	O	XHTMLMP-EVENT-C-023
XHTMLMP-EVENT-C-023	Support for Focus event on mandatory elements	10.3.5	O	
XHTMLMP-EVENT-C-024	Blur event	10.3.6	O	XHTMLMP-EVENT-C-025
XHTMLMP-EVENT-C-025	Support for Blur event on mandatory elements	10.3.6	O	
XHTMLMP-EVENT-C-026	Key Press event	10.3.7	O	XHTMLMP-EVENT-C-027
XHTMLMP-EVENT-C-027	Support for Key Press event on mandatory elements	10.3.7	O	
XHTMLMP-EVENT-C-028	Support for Key Press event on all specified elements	10.3.7	O	
XHTMLMP-EVENT-C-029	Key Down event	10.3.8	O	XHTMLMP-EVENT-C-030
XHTMLMP-EVENT-C-030	Support for Key Down event on mandatory elements	10.3.8	O	
XHTMLMP-EVENT-C-031	Support for Key Down event on all specified elements	10.3.8	O	
XHTMLMP-EVENT-C-032	Key Up event	10.3.9	O	XHTMLMP-EVENT-C-033
XHTMLMP-EVENT-C-033	Support for Key Up event on mandatory elements	10.3.9	O	
XHTMLMP-EVENT-C-034	Support for Key Up event on all specified elements	10.3.9	O	
XHTMLMP-EVENT-C-035	Submit event	10.3.10	O	XHTMLMP-EVENT-C-036
XHTMLMP-EVENT-C-036	Support for Submit event on form element	10.3.10	O	
XHTMLMP-EVENT-C-037	Reset event	10.3.11	O	XHTMLMP-EVENT-C-038
XHTMLMP-EVENT-C-038	Support for Reset event on form element	10.3.11	O	
XHTMLMP-EVENT-C-039	Select event	10.3.12	O	XHTMLMP-EVENT-C-040
XHTMLMP-XHVENT-C-040	Support for Select event on input, textarea elements	10.3.12	O	
XHTMLMP-EVENT-C-041	Change event	10.3.13	O	XHTMLMP-EVENT-C-042
XHTMLMP-EVENT-C-042	Support for Change event on input, select, textarea elements	10.3.13	O	

Item	Function	Reference	Status	Requirement
XHTMLMP-EVENT-C-043	Support for single event handler per element per event	10.4	O	
XHTMLMP-EVENT-C-044	Registration	10.4	O	
XHTMLMP-EVENT-C-045	Attempt to modify the value causes deregistration	10.4	O	
XHTMLMP-EVENT-C-046	Support for cancellable events	10.5	O	

Note that event support is mandatory dependant upon support for scripting. See section 7.2 for specification relationships.

The object Element

Item	Function	Reference	Status	Requirement
XHTMLMP-OBJECT-C-001	Support for object element	11.1	M	
XHTMLMP-OBJECT-C-002	Conform to generic rules for rendering the `object` element ([HTML4], section 13.3)	11.1	M	
XHTMLMP-OBJECT-C-003	Support for 'data' attribute as the location of the object's data	11.2	M	
XHTMLMP-OBJECT-C-004	Support for 'classid' attribute as the location of the object's implementation	11.3	M	
XHTMLMP-OBJECT-C-005	The 'classid' attribute takes precedence over 'data' attribute when both are specified	11.1	M	
XHTMLMP-OBJECT-C-006	Pass parameters defined with `param` element to local app	11.2, 11.3	M	
XHTMLMP-OBJECT-C-007	Use of PUSH Application ID to identify the location application	11.3	M	
XHTMLMP-OBJECT-C-008	Support for declared objects ('declare' attribute)	11.4	M	

Navigation Optimizations

Item	Function	Reference	Status	Requirement
XHTMLMP-NAVOPT-C-001	Presentation hint indicating presence of an access key for an element	12.1	O	
XHTMLMP-NAVOPT-C-002	Use of the 'title' attribute of the anchor element as the link title when presenting page's links in a navigation menu	12.2	O	
XHTMLMP-NAVOPT-C-003	Provide user access to linked resources defined by the `link` element, e.g. through a navigation menu	12.3	O	
XHTMLMP-NAVOPT-C-004	Preloading of linked resources	12.3	O	

Text Input Modes

Item	Function	Reference	Status	Requirement
XHTMLMP-INPUTMODE-C-001	Support for XHTML inputmode Attribute module	13	O	XHTMLMP-INPUTMODE-C-002 AND XHTMLMP-INPUTMODE-C-003
XHTMLMP-INPUTMODE-C-002	Support for 'inputmode' attribute on `input` element	13.2	O	
XHTMLMP-INPUTMODE-C-003	Support for 'inputmode' attribute on `textarea` element	13.2	O	

Note: *inputmode* is optional because it is treated as a hint to the user agent.

Minimum Processing Capacity

Item	Function	Reference	Status	Requirement
XHTMLMP-MINCAP-C-001	Option elements	14.2	M	
XHTMLMP-MINCAP-C-002	Input elements	14.2	M	
XHTMLMP-MINCAP-C-003	Textaea elements	14.2	M	
XHTMLMP-MINCAP-C-004	Anchor elements	14.2	M	
XHTMLMP-MINCAP-C-005	Byte size of textarea elements	14.2	M	
XHTMLMP-MINCAP-C-006	Graceful failure behavior	14.2	M	

Using schemes in XHTML Forms

Item	Function	Reference	Status	Requirement
XHTMLMP-URISCHEMEFORM-C-001	Correct processing of URI schemes in XHTML forms	Section 15.1	M	

Appendix B.   Change History   (Informative)

B.1   Approved Version History

Reference	Date	Description
OMA-TS-XHTMLMP-V1_3-20110329-A	29 Mar 2011	Status changed to Approved by TP: OMA-TP-2011-0097-INP_Browsing_V2_4_ERP_for_Final_Approval