XHTML Mobile Profile

Candidate Version 1.1 – 16 Aug 2004

Open Mobile Alliance

OMA-WAP-XHTMLMP-V1_1-20040816-C

Continues the Technical Activities
Originated in the WAP Forum

Contents

1. Scope

Open Mobile Alliance (OMA) is a consortium of mobile industry companies working to define an industry-wide specification for developing applications that operate over wireless communication networks. The scope for the OMA is to define a set of specifications to be used by service applications. The wireless market is growing very quickly and reaching new customers and services. To enable operators and manufacturers to meet the challenges in advanced services, differentiation and fast/flexible service creation, OMA defines a set of protocols in transport, session and application layers. For additional information on the Wireless Application Environment, refer to [WAE].

This specification defines an XHTML document type based upon the module framework and the modules defined by Modularization of XHTML [XHTMLMod]. This document type is called XHTML Mobile Profile and is designed for resource-constrained Web clients that do not support the full set of XHTML features, such as mobile phones, PDAs, pagers and set-top boxes. It extends XHTML Basic with modules, elements and attributes to provide a richer authoring language. In addition, this specification defines conformance requirements for user agents that process XHTML Mobile Profile documents.

2. References

2.1. Normative References

[CREQ] "Specification of WAP Conformance Requirements", WAP Forum™, WAP-221-CREQ. URL:http//www.wapforum.org/

[DOM2Events] "Document Object Model (DOM) Level 2 Events Specification", Version 1.0, W3C Recommendation, 13 November 2000. http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113

[ESMP] "ECMAScript – Mobile Profile", WAP Forum™, WAP-271-ESMP. URL:http//www.wapforum.org/

[HTML4] "HTML 4.01 Specification", W3C Recommendation 24 December 1999, Dave Raggett et al., editors. URL:http://www.w3.org/TR/html401/

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997. URL:http://www.ietf.org/rfc/rfc2119.txt

[WCSS] "WAP CSS Specification", WAP Forum™, WAP-239-WCSS. URL:http://www.wapforum.org/

[XHTMLBasic] "XHTML™ Basic", W3C Recommendation 19 December 2000, Mark Baker et al., editors. URL:http://www.w3.org/TR/2000/REC-xhtml-basic-20001219

[XHTMLMod] "Modularization of XHTML™", W3C Recommendation 10 April 2001, M. Altheim et al., editors. URL:http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410

2.2. Informative References

[ECMAScript] "ECMAScript Language Specification – Edition 3", ECMA, ECMA-262, December 1999. URL:ftp://ftp.ecma.ch/ecma-st/Ecma-262.pdf

[WAE] "Wireless Application Environment Specification Version 2.1", WAP Forum™, WAP-299-WAE. URL:http://www.wapforum.org/

[WAPARCH] "WAP Architecture", WAP Forum™, WAP-210-WAPArch. URL:http//www.wapforum.org/

[XHTML11] "XHTML™ 1.1 - Module-based XHTML", W3C Recommendation 31 May 2001, Murry Altheim, Shane McCarron, editors. URL:http://www.w3.org/TR/2001/REC-xhtml11-20010531/

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. All examples are informative.

3.2. Definitions

User – person who interacts with a user agent to view, hear or otherwise use a resource

User Agent – any software or device that interprets XHTML Mobile Profile documents and other resources on behalf of the user

3.3. Abbreviations

CSS	Cascading Style Sheets
DTD	Document Type Definition
ESMP	ECMAScript Mobile Profile
PDA	Personal Digital Assistant
WAP	Wireless Application Protocol
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language

4. Introduction

The motivation for the XHTML Mobile Profile document type is to provide an authoring language based upon XHTML that addressed the special requirements of Web clients operating on resource-constrained devices such as mobile phones.

The XHTML Mobile Profile is a strict subset of XHTML 1.1 [XHTML11]. It extends XHTML Basic [XHTMLBasic] to bring enhanced functionality to application authors, including additional presentation elements and support for internal style sheets.

XHTML Mobile Profile 1.1 adds support for a scripting environment. This includes a mechanism for including scripts within a document and a mechanism for binding scripts to events.

5. The XHTML Mobile Profile 1.1 Document Type

The XHTML Mobile Profile 1.1 document type is an XHTML document type based upon the module framework and the modules defined by [XHTMLMod].

The XHTML Mobile Profile 1.1 document type is defined as a strict superset of [XHTMLBasic]. XHTML Mobile Profile 1.1 consists of the XHTML modules specified in this section. All XHTML modules are defined in [XHTMLMod].

The XHTML Mobile Profile 1.1 document type is a strict superset of XHTML Mobile Profile 1.0. It adds the Scripting Module and Intrinsic Events Module from [XHTMLMod].

XHTML Basic

Module	Element
Structure	`body`, `head`, `html`, `title`
Text	`abbr`, `acronym`, `address`, `blockquote`, `br`, `cite`, `code`, `dfn`, `div`, `em`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `kbd`, `p`, `pre`, `q`, `samp`, `span`, `strong`, `var`
Hypertext	`a`
List	`dl`, `dt`, `dd`, `ol`, `ul`, `li`
Basic Forms	`form`, `input`, `label`, `select`, `option`, `textarea`
Basic Tables	`caption`, `table`, `td`, `th`, `tr`
Image	`img`
Object	`object`, `param`
Metainformation	`meta`
Link	`link`
Base	`base`

Additional XHTML Modules

Module	Elements/Attributes
Forms (*partial*)	`fieldset`, `optgroup`
Intrinsic Events (*partial*)	Various attributes (see section 10.2)
Legacy (*partial*)	`start` attribute on `ol`, `value` attribute on `li`
Presentation (*partial*)	`b`, `big`, `hr`, `i`, `small`
Scripting	`noscript`, `script`

Style Sheet	`style` element
Style Attribute	`style` attribute

For modules marked as *partial* above, the module was not included in its entirety (only selected elements were included). The implication of this is that the XHTML Mobile Profile 1.1 document type is not strictly *XHTML Host Language Conforming*, as defined in [XHTMLMod].

An implementation of this document type as an XML 1.0 DTD is available in Appendix C.

The elements, attributes and minimum content models associated with these modules are defined in [XHTMLMod]. The elements are listed here for informative purposes, but the definitions in [XHTMLMod] should be considered normative. In some cases, the minimum content models have been extended. The DTD found in Appendix C should be consulted for more information.

6. Use of XHTML Mobile Profile

The XHTML Mobile Profile document type serves as an authoring language for content targeted at resource-constrained devices. It is expected that it can be used as it is for this purpose.

The XHTML Mobile Profile document type could also serve as a host language, that is, a language containing a mix of vocabularies within one document type. Those considering its use as a host language should consider that it is not strictly XHTML Host Language Conforming, as it only partially includes certain modules.

7. Conformance

7.1. Document Conformance

A conforming XHTML Mobile Profile 1.1 document is a document that requires only the facilities described as mandatory in this specification. A conforming document MUST meet all of the following criteria:

1. The document MUST conform to the constraints expressed in Appendix A.
2. The root element of the document MUST be `html`.
3. The name of the default namespace on the root element MUST be the XHTML namespace name, http://www.w3.org/1999/xhtml.
4. There MUST be a DOCTYPE declaration in the document prior to the root element, with a public identifier. The public identifier included in the DOCTYPE declaration must reference the DTD found in Appendix C using its Formal Public Identifier. The system identifier may be modified appropriately. For example,

```
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.1//EN"
   "http://www.wapforum.org/DTD/xhtml-mobile11.dtd">
```

5. The DTD internal subset MUST NOT be used to override any parameter entities in the DTD.

A document that meets all of these criteria is positively identified as an XHTML Mobile Profile 1.1 document. For user agents that do not validate documents according to the DTD, a document can be identified as XHTML Mobile Profile 1.1 by detecting a properly formatted DOCTYPE declaration as specified above.

7.2. User Agent Conformance

A conforming user agent MUST meet all the user agent conformance requirements defined in [XHTMLMod].

The MIME media type for XHTML Mobile Profile is “application/vnd.wap.xhtml+xml”.

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 “application/xhtml+xml”.

Note that there is no requirement that XHTML Mobile Profile documents be given the media type “application/vnd.wap.xhtml+xml”; the media type “text/html” may be used instead. As there are no conformance rules for documents with type “text/html”, there is no easy way to determine which documents of type “text/html” are XHTML Mobile Profile documents, except that the document may include the DOCTYPE declaration specified in section 7.1. A conforming user agent SHOULD also accept XHTML Mobile Profile documents identified with type “text/html”.

When declaring support for XHTML Mobile Profile, a conforming user agent MUST use an `Accept` header with value `application/xhtml+xml; profile="http://www.wapforum.org/xhtml"` and an `Accept` header with value `application/vnd.wap.xhtml+xml`.

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

8. Use of Style Sheets with XHTML Mobile Profile

Style sheets can be used to style XHTML Mobile Profile documents. If a WAE user agent supports styling of documents with style sheets, it MUST support the style language WAP CSS [WCSS], a subset of CSS2 with WAP-specific extensions. A user agent MAY support other style languages.

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 information
- Inline style information

8.1.1. 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 `type="text/css"`, the user agent MUST process the style sheet according to the style language WAP 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>
```

8.1.2. 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 WAP 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.

8.1.3. 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. The default style language for style information in the `style` attribute is WAP 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.

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.

The scripting language defined for use with XHTML Mobile Profile is ECMAScript Mobile Profile [ESMP]. Support for scripting is RECOMMENDED. If the user agent supports scripting, it MUST support [ESMP], but MAY also support other scripting languages. Exact conformance requirements for user agents may be defined elsewhere.

9.1. Adding Script to XHTML Mobile Profile

An ECMAScript program, as defined in [ECMAScript], is included with an XHTML Mobile Profile document using the `script` element. The `script` element defines a script, which is a series of ECMAScript statements. An XHTML Mobile Profile 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.

9.1.1. 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 Appendix C 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.

9.1.2. The `noscript` Element

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

9.2. Script Execution

Scripts are executed:

- As the document is loaded by the user agent
- When a specific event occurs on a certain element

9.2.1. 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.

9.2.1.1. 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 9.2.1.2. 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.

9.2.1.2. 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.

9.2.1.3. Supported Scripting Languages

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.

The XHTML Mobile Profile 1.1 user agent MUST support the scripting language ECMAScript Mobile Profile. The MIME media type for ECMAScript Mobile Profile is `text/ecmascript`. The user agent MUST also support ECMAScript Mobile Profile scripts identified with media type `text/javascript`.

The user agent MAY support other scripting languages.

9.2.2. 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.

9.3. Examples

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

9.3.1. Script Executed on Document Load

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

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

9.3.2. 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>
 </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 2 Event Model

The event model for XHTML Mobile Profile is a profile of the event model described by W3C DOM Level 2 Events [DOM2Events], called the DOM Event Model. Specifically, the event capture and event bubbling phases of event processing 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 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 are given in section 10.2.

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, label, input, select, textarea	Occurs when a markup element gains focus	O
Blur	onblur	a, 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`

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`

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`, `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`, `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`.

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`, `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`, `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`.

10.3.5. Focus

Event: Focus

Binding: `onfocus`

Support: Optional

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

Target Element: `a`, `label`, `input`, `select`, `textarea`

10.3.6. Blur

Event: Blur

Binding: `onblur`

Support: Optional

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

Target Element: `a`, `label`, `input`, `select`, `textarea`

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`, `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`, `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`.

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`, `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`, `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`.

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`, `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`, `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`.

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`

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`

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 Element: `input`, `textarea`

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 Element: `input`, `select`, `textarea`

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`, `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`.

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 [DOM2Events] for complete details.

The event model for XHTML Mobile Profile uses the DOM Event Model as the reference model, but does not directly support event handler registration using the interfaces of [DOM2Events]. 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, an `Event` object as defined in ECMAScript Mobile Profile [ESMP]. The `Event` object is in scope for all statements of the event handler:

```
 function (Event event)
 {
   value of 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 is supported for a given event type on a given target element.

10.5. Event Cancellation

The event model for XHTML Mobile Profile provides a mechanism for cancelling an event. Cancelling an event means to prevent the default action for the event. The default action is the action taken by the user agent to handle that event in the case that no script event handler is bound to the event. For example, the default action for the Click event on a hyperlink is to activate the hyperlink, that is, to load the resource indicated by the URI specified by the `href` attribute. Cancelling an event means that the script will completely determine the handling of the event.

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 cancelling the default action for an event – the method `preventDefault()` on the `Event` object:

```
interface Event {
 ...
 void preventDefault();
 ...
}
```

See [DOM2Events] for complete details of the DOM Level 2 Event object.

The event model for XHTML Mobile Profile uses the DOM Event Model as a reference, but does not directly support cancelling events using the `preventDefault` interface of [DOM2Events]. Instead, an XHTML Mobile Profile event handler cancels the default action for an event by returning the value `false` from the anonymous event handler function. If the value `true` is returned, or no value is returned, the event is not cancelled and the default action occurs. See [ESMP] for details of the ECMAScript Mobile Profile Event object.

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

- Click
- 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.

10.6. Examples

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

10.6.1. Event Handler Function

The following shows an example of using a function as an 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 using a function as an event handler, and passing arguments to that function:

```
<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 using a function as an event handler, and passing the Event object to that function:

```
<html>
 <head>
  <title>Script Example: 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>
```

Appendix A. Static Conformance Requirements (Normative)

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

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-005	XHTML Basic 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	

Other XHTML Modules

Item	Function	Reference	Status	Requirement
XHTMLMP-XHTMLMOD-C-012	XHTML Style Sheet module	5	M	
XHTMLMP-XHTMLMOD-C-013	XHTML Style Attribute module	5	M	
XHTMLMP-XHTMLMOD-C-023	XHTML Intrinsic Events module	5	M	
XHTMLMP-XHTMLMOD-C-024	XHTML Scripting module	5	M	

Other XHTML Elements and Attributes

Item	Function	Reference	Status	Requirement
XHTMLMP-XHTMLMOD-C-014	`fieldset` element in Forms module	5	M	
XHTMLMP-XHTMLMOD-C-015	`optgroup` element in Forms module	5	M	
XHTMLMP-XHTMLMOD-C-016	`start` attribute on `ol`	5	M	

Item	Function	Reference	Status	Requirement
XHTMLMP-XHTMLMOD-C-017	`value` attribute on `li`	5	M	
XHTMLMP-XHTMLMOD-C-018	`b` element in Presentation module	5	M	
XHTMLMP-XHTMLMOD-C-019	`big` element in Presentation module	5	M	
XHTMLMP-XHTMLMOD-C-020	`hr` element in Presentation module	5	M	
XHTMLMP-XHTMLMOD-C-021	`i` element in Presentation module	5	M	
XHTMLMP-XHTMLMOD-C-022	`small` element in Presentation module	5	M	

XHTML User Agent Conformance

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

Document Types

Item	Function	Reference	Status	Requirement
XHTMLMP-DOC-C-001	Accept XHTML Mobile Profile documents	7.2	M	
XHTMLMP-DOC-C-002	Advertise support for XHTML Mobile Profile documents	7.2	M	

Style Sheets

Item	Function	Reference	Status	Requirement
XHTMLMP-STYLE-C-001	Support for WAP CSS	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	8.1.1	O	
XHTMLMP-STYLE-C-003	Handling of type "text/css" for internal style sheet	8.1.2	O	
XHTMLMP-STYLE-C-004	Default type "text/css" for inline style rules	8.1.3	O	

Scripting

Item	Function	Reference	Status	Requirement

Item	Function	Reference	Status	Requirement
XHTMLMP-SCRIPT-C-001	Support for scripting	0	O	XHTMLMP-SCRIPT-C-002 AND XHTMLMP-EVENT-C-001
XHTMLMP-SCRIPT-C-002	Support for ESMP	0	O	ESMP:MCF AND XHTMLMP-SCRIPT-C-003 AND XHTMLMP-SCRIPT-C-004 AND XHTMLMP-SCRIPT-C-005
XHTMLMP-SCRIPT-C-003	Script reference processing model	9.2.1	O	
XHTMLMP-SCRIPT-C-004	Accepts ESMP with MIME media type text/ecmascript	9.2.1.3	O	
XHTMLMP-SCRIPT-C-005	Accepts ESMP with MIME media type text/javascript	9.2.1.3	O	

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
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	

Item	Function	Reference	Status	Requirement
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	

Item	Function	Reference	Status	Requirement
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-EVENT-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	
XHTMLMP-EVENT-C-043	Support for single event handler per element per event	10.4	O	
XHTMLMP-EVENT-C-044	Support for cancellable events	10.5	M	

Appendix B. Change History (Informative)

B.1 Approved Version History

Reference	Date	Description
n/a	n/a	No prior version

B.2 Draft/Candidate Version 1.1 History

Document Identifier	Date	Sections	Description
Candidate Versions	04 Sep 2002	n/a	
OMA-WAP-XHTMLMP-V1_1	16 Aug 2003	10.5	Bug fix – Added cancellable events description.

Appendix C. XHTML Mobile Profile 1.1 Document Type Definition (Normative)

See http://www.wapforum.org/DTD/xhtml-mobile11.zip for the ZIP archive of the DTD for XHTML Mobile Profile 1.1. It includes the DTD driver file (xhtml-mobile11.dtd) and a "flat" version of the DTD (xhtml-mobile11-flat.dtd).

See http://www.wapforum.org/DTD/xhtml-mobile11-flat.dtd for the "flat" version of the DTD for XHTML Mobile Profile 1.1.

See http://www.wapforum.org/DTD/xhtml-mobile11.dtd for the driver file for the XHTML Mobile Profile 1.1 DTD. This file depends on the XHTML module implementations defined in [XHTMLMod].