WAP CSS Specification

Version 26-Oct-2001

Wireless Application Protocol
WAP-239-WCSS-20011026-a

A list of errata and updates to this document is available from the WAP Forum™ Web site, http://www.wapforum.org/, in the form of SIN documents, which are subject to revision or removal without notice.

This document is available online in PDF format at http://www.wapforum.org/.

Known problems associated with this document are published at http://www.wapforum.org/.

Comments regarding this document can be submitted to the WAP Forum™ in the manner published at http://www.wapforum.org/.

Document History	
WAP-239-WCSS-20011026-p	Current
WAP-239_100-WCSS-20011026-a	SIN

Contents

1. Scope

This section is informative.

The WAP CSS specifies a subset of CSS2 [CSS2] and WAP specific extensions. It can be used to style XHTML Basic and WML 2.0 documents, as well as any other XML document.

A subset of CSS2 is also specified in the W3C, the “W3C CSS Mobile profile” working draft [CSSM]. Since both CSS2 variants follow CSS user agent semantics, conforming WAP CSS user agents will accept valid W3C CSS Mobile Profile style sheets.

The CSS2 subset contains core CSS functionality such as inheritance, cascading, selectors, and the CSS syntax. It also contains CSS properties for margins, text, fonts, visual effects, colour, and lists.

The following WAP extensions are specified:

- **Marquee** - properties to create simple animation effects. Typically this will be used to create scrolling text.
- **Input** - properties to specify the format of the user input into a form control. It is the same feature as the WML ‘format’ and ‘emptyok’ attributes provide.
- **Accesskey** - property to specify an optional, additional way to activate an element in a document. It is similar to the feature the HTML ‘accesskey’ attribute provides.

2. References

2.1. Normative

[CREQ] "Specification of WAP Conformance Requirements". WAP Forum™. WAP-221-CREQ-20010425-a. URL:*http//www.wapforum.org/*

[CSS2] "Cascading Style Sheets, level 2", CSS2 Specification, W3C Recommendation.
URL: *http://www.w3.org/TR/1998/REC-CSS2-19980512*

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

[RFC2318] "The text/css Media Type", H. Lie, B. Bos, C. Lilley, March 1998.
URL*: http://www.ietf.org/rfc/rfc2318.txt*

[WML2] "WML 2.0", WAP-238-WML.
URL: *http://www.wapforum.org/*

[XMLCSS] "Associating Style Sheets with XML documents", Version 1.0, W3C Recommendation.
URL: *http://www.w3.org/1999/06/REC-xml-stylesheet-19990629*

2.2. Informative

[BASIC] "XHTML Basic", W3C Recommendation
URL: *http://www.w3.org/TR/2000/REC-xhtml-basic-20001219*

[CSSM] "CSS Mobile Profile 1.0", W3C Working Draft
URL: *http://www.w3.org/TR/2001/WD-css-mobile-20010129*

[GSM0230] "Digital cellular telecommunications system (Phase 2+); Man-Machine Interface (MMI) of the Mobile Station (MS)", GSM 02.30 version 7.1.0.
URL: *http://www.3gpp.org/*

[SELECTORS] "CSS3 module: W3C selectors", W3C Working Draft
URL: *http://www.w3.org/TR/2001/WD-css3-selectors-20010126*

[UICSS3] "User Interface for CSS3 ", W3C Working Draft.
URL: *http://www.w3.org/TR/2000/WD-css3-userint-20000216*

3. Definitions and Abbreviations

3.1. Definitions

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].

CSS property names are written as

‘property name’.

CSS property values are written as

property value.

Data types are written as

<data type>

Definitions are written as

Definition: A definition is a term or expression that is used a lot.

Conformance requirements are written as

Conformance requirement: A user agent that conforms to this part of the specification MUST implement this requirement.

Notes are written as

Note: An informational message to the reader.

3.2. Abbreviations

CSS	Cascading Style Sheets
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
WML	Wireless Markup Language
XHTML	Extensible Hyper Text Markup Language
XML	Extensible Markup Language

4. Introduction

This section is informative.

Readers of this specification are assumed to be familiar with Cascading Style Sheets (CSS) as defined in [CSS2].

4.1. How to Read This Specification

The major part of this specification is a table with normative references to the CSS2 specification. The last few sections specify the WAP specific extensions to CSS. Appendix A follows the syntax described in [CREQ] and is the Static Conformance Requirements (SCR) for the whole specification.

It is recommended that the specification is read in the following way:

1. Read Appendix A to get an overall view of what parts of the specification are mandatory and what parts are optional. The SCR is a list of features of WAP CSS.

2. For each feature in the SCR, details on conformance can be found in the referenced section.

Where there are references to CSS look into the CSS specification [CSS2] for details.

Note: The tables in the body of this document are not SCR tables.

4.2. Conformance

Conformance to this specification is defined in the Static Conformance Requirements (SCR) table in Appendix A. A conforming user agent may support additional CSS features.

This specification **does not** specify user interface presentation. It is not specified what a dotted line or a solid line should look like or how colors should be represented on the display. The look and feel depends on the device. This limits to what extent certain features can be tested. It is however expected that conformance testing will be carried out in such a way that it is possible to distinguish between a device that has implemented a particular feature and one that has not. It can be done for example by observing just the difference in effect of two different property values. A device that is unable to support a certain feature cannot claim to conform to that feature.

> **Note**: Conformance to this specification is intended to be stronger than for CSS2, where conformance can be claimed even if parts of the specification have not been implemented, see [CSS2] section 3.1. In WAP CSS conformance can be claimed, in the Implementation Conformance Statement (ICS), only for features that have actually been implemented.

Fortunately, in many cases it is clear how a property value should be presented. Whether a font is “sans serif” or whether the text is indented “five pixels” can be tested. And if that test fails, the device does not conform to that feature.

Some features are optional and some features are mandatory. Here are some of the principles that have been used to determine what shall be mandatory and what shall be optional:

- For interoperability most of the core CSS features are mandatory: syntax, selectors, cascading, inheritance, how values are calculated, etc.

- Since devices have very different capabilities, all properties are optional. In some cases, however, related properties are put in groups and support for one of the properties in the group requires support for all other properties of the same group.

- For consistency, when there are variants of the same property, such as ‘border-left’ and ‘border-right’ of the ‘border’ property, then all variants are required.

- On the properties where the **auto** and **inherit** values are defined, these values are mandatory..

- In general, property values that are keywords, such as **larger**, **left**, and **smaller**, are required for that property.
- When a property takes values of only one type, that type is required.
- Some properties can take many different types of values. A property requires at least one type of values. The *requirement* column in the SCR table is used to express the dependency between a property and the types of values it can take.
- Values expressed as colours always require support for the 16 HTML colour keywords and the RGB specification.
- Shorthand properties are optional.

4.3. The ‘text/css’ Media Type

Since WAP CSS style sheets are either valid CSS2 style sheets [CSS2] or contain WAP extensions expressed in valid CSS syntax, when delivered over the Internet, the ‘text/css’ [RFC2318] MIME media type can be used to represent WAP CSS style sheets.

No WAP specific MIME media type for WAP CSS has been defined.

5. Selectors

This section is normative.

5.1. Pattern Matching

In order to support the WAP CSS Selectors the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
SELECTOR-1	Universal selector	[CSS2] section 5.3	M	
SELECTOR-2	Type selector	[CSS2] section 5.4	M	
SELECTOR-3	Descendant selector	[CSS2] section 5.5	M	
SELECTOR-4	Link pseudo-class: :link	[CSS2] section 5.11.2	M	
SELECTOR-5	Link pseudo-class: :visited	[CSS2] section 5.11.2	O	
SELECTOR-6	Dynamic pseudo-class: :active	[CSS2] section 5.11.3	O	
SELECTOR-7	Dynamic pseudo-class: :focus	[CSS2] section 5.11.3	O	
SELECTOR-8	Class selectors	[CSS2] section 5.8.3	M	
SELECTOR-9	ID selectors	[CSS2] section 5.9	M	
SELECTOR-10	Grouping	[CSS2] section 5.2.1	M	

Note: In order to conform to the W3C CSS Mobile profile the optional “:visited”, “:active”, and “:focus” pseudo-classes must be supported.

6. Syntax and Basic Data Types

This section is normative.

6.1. Syntax and Parsing

In order to support the WAP CSS Syntax and Parsing the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
SYNTAX-1	CSS style sheet syntax	[CSS2] section 4.1	M	
SYNTAX-2	Rules for handling parsing errors	[CSS2] section 4.2	M	
SYNTAX-3	Algorithm for determining the character encoding	[CSS2] section 4.4	M	
SYNTAX-4	The “charset” at-rule in external style sheets	[CSS2] section 4.4	M	

6.2. Data Types

In order to support the WAP CSS Data Types the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
TYPES-1	<number> Integers and real numbers.	[CSS2] section 4.3.1	M	
TYPES-2	<length> The “px”, “em”, and “ex” length units.	[CSS2] section 4.3.2	M	TYPES-1
TYPES-3	<percentage>	[CSS2] section 4.3.3	M	TYPES-1
TYPES-4	<uri>	[CSS2] section 4.3.4	M	
TYPES-5	<color> The 16 HTML 4.0 colours.	[CSS2] section 4.3.6	M	
TYPES-6	<color> Numerical RGB specification.	[CSS2] section 4.3.6	M	

7. Assigning Property Values, Cascading, and Inheritance

This section is normative.

In order to support the WAP CSS Assigning property values, cascading, and inheritance the user agent MUST implement all mandatory items ("M") in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
ASSIGN-1	Assigning values	[CSS2] section 6.1	M	
INHERIT-1	Inheritance	[CSS2] section 6.2	M	
INHERIT-2	The inherit value	[CSS2] section 6.2.1	M	
IMPORT-1	The "import" at-rule	[CSS2] section 6.3	M	
CASCADE-1	The cascade	[CSS2] section 6.4	M	

Note: [CSS2] section 6.4.4, covered by CASCADE-1, specifies how non-CSS presentational hints (e.g. the presentational markup found in WML and XHTML) are used in the CSS cascade. It also specifies the same thing for the style rules in the XHTML "style" attribute.

8. Media Types

This section is normative.

In order to support the WAP CSS Media Types the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
MEDIA-1	The ‘media’ at-rule	[CSS2] section 7.2.1	M	
MEDIA-2	Accept and process style sheets that target the ‘handheld’ and ‘all’ media types	[CSS2] section 7.3	M	

Note: A user agent may support additional media types, such as “aural” if it supports aural style sheets.

9. Associating Style Sheets With XML Document

This section is normative.

In order to support the WAP CSS Associating style sheets with XML documents the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
ASSOC-1	The ‘xml-stylesheet’ Processing Instruction	[XMLCSS]	M	

10. Box Model

This section is normative.

10.1. Margin

In order to support the WAP CSS Margin properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
MARGIN-1	‘margin-top’, ‘margin-right’, ‘margin-bottom’, ‘margin-left’	[CSS2] section 8.3	M	MARGIN-2 OR MARGIN-3 OR MARGIN-4
MARGIN-2	Margin as <length> value	[CSS2] section 8.3	M	TYPES-2
MARGIN-3	Margin as <percentage> value	[CSS2] section 8.3	M	TYPES-3
MARGIN-4	Auto value	[CSS2] section 8.3	M	
MARGIN-5	‘margin’	[CSS2] section 8.3	O	MARGIN-1

10.2. Padding

In order to support the WAP CSS Padding properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
PADDING-1	‘padding-top’, ‘padding-right’, ‘padding-bottom’, ‘padding-left’	[CSS2] section 8.4	M	PADDING-2 OR PADDING-3
PADDING-2	Padding as <length> value	[CSS2] section 8.4	M	TYPES-2
PADDING-3	Padding as <percentage> value	[CSS2] section 8.4	M	TYPES-3
PADDING-4	‘padding’	[CSS2] section 8.4	O	PADDING-1

10.3. Border Width

In order to support the WAP CSS Border Width properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
BORDER-WIDTH-1	‘border-top-width’, ‘border-right-width’, ‘border-bottom-width’, ‘border-left-width’	[CSS2] section 8.5.1	M	BORDER-WIDTH-2 OR BORDER-WIDTH-3
BORDER-WIDTH-2	Border width keywords: thin, medium, thick	[CSS2] section 8.5.1	M	
BORDER-WIDTH-3	Border width as <length> value.	[CSS2] section 8.5.1	O	TYPES-2
BORDER-WIDTH-4	‘border-width’	[CSS2] section 8.5.1	O	BORDER-WIDTH-1

10.4. Border Colour

In order to support the WAP CSS Border Colour properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
BORDER-COLOR-1	‘border-top-color’, ’border-right-color’, ‘border-bottom-color’, ‘border-left-color’, ‘border-color’	[CSS2] section 8.5.2	M	BORDER-COLOR-2
BORDER-COLOR-2	Border colour as <color>	[CSS2] section 8.5.2	M	TYPES-5 AND TYPES-6

10.5. Border Style

In order to support the WAP CSS Border Style properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
BORDER-STYLE-1	‘border-top-style’, ’border-right-style’, ‘border-bottom-style’, ‘border-left-style’	[CSS2] section 8.5.3	M	BORDER-STYLE-2 OR BORDER-STYLE-3 OR BORDER-STYLE-4
BORDER-STYLE-2	Border style as keywords: none, solid	[CSS2] section 8.5.3	M	
BORDER-STYLE-3	Border style as keywords: hidden	[CSS2] section 8.5.3	O	
BORDER-STYLE-4	Border style as keywords: dotted, dashed, double, groove, ridge, inset, outset	[CSS2] section 8.5.3	O	
BORDER-STYLE-5	‘border-style’	[CSS2] section 8.5.3	O	BORDER-STYLE-1

10.6. Border Shorthand Properties

In order to support the WAP CSS Border Shorthand properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
BORDER-SHORT-1	‘border-top’, ’border-right’, ‘border-bottom’, ‘border-left’, ‘border’	[CSS2] section 8.5.4	M	BORDER-STYLE-1 AND BORDER-COLOR-1 AND BORDER-WIDTH-1

11. Colours and Background

This section is normative.

11.1. Foreground Colour

In order to support the WAP CSS Foreground colour properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
FCOLOR-1	‘color’	[CSS2] section 14.1	M	FCOLOR-2
FCOLOR-2	Foreground as <color>	[CSS2] section 14.1	M	TYPES-5 AND TYPES-6

11.2. Background Colour

In order to support the WAP CSS Background Colour properties the user agent MUST implement all mandatory items ("M") in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
BCOLOR-1	'background-color'	[CSS2] section 14.2	M	BCOLOR-2 OR BCOLOR-3
BCOLOR-2	Transparent colour	[CSS2] section 14.2	M	
BCOLOR-3	Background as <color>	[CSS2] section 14.2	M	TYPES-5 AND TYPES-6

11.3. Background Images

In order to support the WAP CSS Background Images properties the user agent MUST implement all mandatory items ("M") in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
BIMAGE-1	'background-image'	[CSS2] section 14.2	M	BIMAGE-2
BIMAGE-2	Background image as <uri>	[CSS2] section 14.2	M	TYPES-4
BIMAGE-3	'background-repeat'	[CSS2] section 14.2	M	BIMAGE-4 AND BIMAGE-1
BIMAGE-4	Repeat background as: repeat, repeat-x, repeat-y, no-repeat	[CSS2] section 14.2	M	
BIMAGE-5	'background-attachment'	[CSS2] section 14.2	M	BIMAGE-6 OR BIMAGE-7
BIMAGE-6	Background image scroll	[CSS2] section 14.2	M	
BIMAGE-7	Background image fixed	[CSS2] section 14.2	M	
BIMAGE-8	'background-position'	[CSS2] section 14.2	M	(BIMAGE-9 AND BIMAGE-10) OR BIMAGE-11
BIMAGE-9	Background image position as <length> value	[CSS2] section 14.2	O	TYPES-2
BIMAGE-10	Background image position as <percentage> value	[CSS2] section 14.2	O	TYPES-3
BIMAGE-11	Background image position as keywords: top, center, bottom, left, right	[CSS2] section 14.2	M	
BIMAGE-12	'background'	[CSS2] section 14.2	M	BIMAGE-1

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
				AND BIMAGE-3 AND BIMAGE-5 AND BIMAGE-8 AND BCOLOR-1

12. Fonts

This section is normative.

12.1. Font Family

In order to support the WAP CSS Font Family properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
FONTS-FAMILY-1	‘font-family’	[CSS2] section 15.2.2	M	FONTS-FAMILY-2 OR FONTS-FAMILY-3
FONTS-FAMILY-2	<generic-family> Generic font family name value	[CSS2] section 15.2.2	M	
FONTS-FAMILY-3	<font-family> Specific font family name value	[CSS2] section 15.2.2	O	

12.2. Font Style

In order to support the WAP CSS Font Style properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
FONTS-STYLE-1	‘font-style’	[CSS2] section 15.2.3	M	FONTS-STYLE-2
FONTS-STYLE-2	Font styles as keywords: normal, italic, oblique	[CSS2] section 15.2.3	M	

12.3. Font Variant

In order to support the WAP CSS Font Variant properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
FONTS-VARIANT-1	‘font-variant’	[CSS2] section 15.2.3	M	FONTS-VARIANT-2
FONTS-VARIANT-2	Font variants as keywords: normal, small-caps	[CSS2] section 15.2.3	M	

12.4. Font Weight

In order to support the WAP CSS Font Weight properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
FONTS-WEIGHT-1	‘font-weight’	[CSS2] section 15.2.3	M	FONTS-WEIGHT-2
FONTS-WEIGHT-2	Font weight as keywords: normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900	[CSS2] section 15.2.3	M	

12.5. Font Size

In order to support the WAP CSS Font Size properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
FONTS-SIZE-1	‘font-size’	[CSS2] section 15.2.4	M	(FONTS-SIZE-2 AND FONTS-SIZE-3) OR (FONTS-SIZE-4 AND FONTS-SIZE-5)
FONTS-SIZE-2	Font size as absolute keywords: xx-small, x-small, small, medium, large, x-large, xx-large	[CSS2] section 15.2.4	M	
FONTS-SIZE-3	Font size as relative keywords: larger, smaller	[CSS2] section 15.2.4	M	
FONTS-SIZE-4	Font size as <length> value	[CSS2] section 15.2.4	O	TYPES-2
FONTS-SIZE-5	Font size as <percentage> value	[CSS2] section 15.2.4	O	TYPES-3

12.6. Shorthand Font Property

In order to support the WAP CSS Font Shorthand properties the user agent MUST implement all mandatory items ("M") in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
FONTS-SHORT-1	'font'	[CSS2] section 15.2.5	M	FONTS-FAMILY-1 AND FONTS-STYLE-1 AND FONTS-VARIANT-1 AND FONTS-WEIGHT-1 AND FONTS-SIZE-1

13. Lists

13.1. This section is normative.Lists

In order to support the WAP CSS Lists properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
LISTS-1	‘list-style-type’	[CSS2] section 12.6.2	M	LISTS-2
LISTS-2	List style type as keywords: disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha, none	[CSS2] section 12.6.2	M	
LISTS-3	‘list-style-position’	[CSS2] section 12.6.2	M	LISTS-4
LISTS-4	List style position as keywords: inside, outside	[CSS2] section 12.6.2	M	
LISTS-5	‘list-style-image’	[CSS2] section 12.6.2	M	LISTS-6
LISTS-6	List style image, located at specified <uri>	[CSS2] section 12.6.2	M	TYPES-4
LISTS-7	‘list-style’	[CSS2] section 12.6.2	O	LISTS-1 AND LISTS-3 AND LISTS-5

14. Text

This section is normative.

14.1. Indentation

In order to support the WAP CSS Text Indentation properties the user agent MUST implement all mandatory items ("M") in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
TEXT-INDENT-1	'text-indent'	[CSS2] section 16.1	M	TEXT-INDENT-2 OR TEXT-INDENT-3
TEXT-INDENT-2	Text indent as <length> value	[CSS2] section 16.1	M	TYPES-2
TEXT-INDENT-3	Text indent as <percentage> value	[CSS2] section 16.1	M	TYPES-3

14.2. Alignment

In order to support the WAP CSS Text Alignment properties the user agent MUST implement all mandatory items ("M") in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
TEXT-ALIGN-1	'text-align'	[CSS2] section 16.2	M	TEXT-ALIGN-2 OR TEXT-ALIGN-3
TEXT-ALIGN-2	Text align as keywords: left, right, center	[CSS2] section 16.2	M	
TEXT-ALIGN-3	Text align as keywords: justify	[CSS2] section 16.2	O	

14.3. Decoration

In order to support the WAP CSS Text Decoration properties the user agent MUST implement all mandatory items ("M") in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
TEXT-DEC-1	'text-decoration'	[CSS2] section 16.3.1	M	TEXT-DEC-2 OR TEXT-DEC-3
TEXT-DEC-2	Text decoration as keyword: none	[CSS2] section 16.3.1	M	
TEXT-DEC-3	Text decoration as keyword: underline	[CSS2] section 16.3.1	O	
TEXT-DEC-4	Text decoration as keyword: blink	[CSS2] section 16.3.1	O	

14.4. Transformation

In order to support the WAP CSS Text Transformation properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
TEXT-TRANS-1	‘text-transform’	[CSS2] section 16.5	M	TEXT-TRANS-2
TEXT-TRANS-2	Text transformation as keywords: capitalize, uppercase, lowercase, none	[CSS2] section 16.5	M	

14.5. White Space

In order to support the WAP CSS Text White space properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
TEXT-WS-1	‘white-space’	[CSS2] section 16.6	M	TEXT-WS-2
TEXT-WS-2	White space as keywords: normal, pre, nowrap	[CSS2] section 16.6	M	

15. Visual Effects

This section is normative.

In order to support the WAP CSS Visual effects properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
VEFFECT-1	‘visibility’	[CSS2] section 11.2	M	VEFFECT-2 OR VEFFECT-3
VEFFECT-2	Visibility as keywords: visible, hidden	[CSS2] section 11.2	M	
VEFFECT-3	Visibility as keywords: collapse	[CSS2] section 11.2	O	

16. Visual Formatting

This section is normative.

16.1. Display Properties

In order to support the WAP CSS Display properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
VFORM-DISP-1	‘display’	[CSS2] section 9.2.5	M	VFORM-DISP-2
VFORM-DISP-2	Display property as keywords: inline, block, list-item, none	[CSS2] section 9.2.5	M	

16.2. Float Positioning

In order to support the WAP CSS Float Positioning properties the user agent MUST implement all mandatory items ("M") in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
VFORM-FLOAT-1	'float'	[CSS2] section 9.5.1	M	VFORM-FLOAT-2
VFORM-FLOAT-2	Positioning of float as keywords: left, right, none	[CSS2] section 9.5.1	M	

16.3. Float Flow Control

In order to support the WAP CSS Float Flow Control properties the user agent MUST implement all mandatory items ("M") in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
VFORM-CLEAR-1	'clear'	[CSS2] section 9.5.2	M	VFORM-CLEAR-2 AND VFORM-FLOAT-1
VFORM-CLEAR-2	Controlling flow next to float as keywords: left, right, both, none	[CSS2] section 9.5.2	M	

16.4. Content Width and Height

In order to support the WAP CSS Content Width and Height properties the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
VFORM-SIZE-1	‘width’	[CSS2] section 10.2	M	VFORM-SIZE-2 OR VFORM-SIZE-3 OR VFORM-SIZE-4
VFORM-SIZE-2	Content width as <length> value	[CSS2] section 10.2	M	TYPES-2
VFORM-SIZE-3	Content width as <percentage> value	[CSS2] section 10.2	M	TYPES-3
VFORM-SIZE-4	Auto value	[CSS2] section 10.2	M	
VFORM-SIZE-5	‘height’	[CSS2] section 10.5	M	VFORM-SIZE-6 OR VFORM-SIZE-7 OR VFORM-SIZE-8
VFORM-SIZE-6	Content height as <length> value	[CSS2] section 10.5	M	TYPES-2
VFORM-SIZE-7	Content height as <percentage> value	[CSS2] section 10.5	M	TYPES-3
VFORM-SIZE-8	Auto value	[CSS2] section 10.5	M	
VFORM-SIZE-9	‘vertical-align’	[CSS2] section 10.8.1	M	VFORM-SIZE-10
VFORM-SIZE-10	Vertical align as keywords: baseline, sub, super, top, middle, bottom	[CSS2] section 10.8.1	M	

17. WAP CSS Extension: Marquee

This section is normative.

In order to support the WAP Marquee CSS extension the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
WAPEXT-MARQUEE-1	Display property keyword: -wap-marquee	Section 17.1	M	VFORM-DISP-1
WAPEXT-MARQUEE-2	‘-wap-marquee-style’	Section 17.2	O	
WAPEXT-MARQUEE-3	‘-wap-marquee-loop’	Section 17.3	O	
WAPEXT-MARQUEE-4	‘-wap-marquee-dir’	Section 17.4	O	
WAPEXT-MARQUEE-5	‘-wap-marquee-speed’	Section 17.5	O	

17.1. The -wap-marquee Property Value

Definition: The *marquee box* inherits and extends the characteristics of the principal block box ([CSS2] section 9.2.1). The content of the *marquee box* is animated according to the marquee properties defined in this section. As principal block boxes, *marquee boxes* participate in a block formatting context ([CSS2] section 9.4.1).

Conformance Requirement: When the 'display' property is set to **-wap-marquee** the user agent MUST generate one *marquee box* with all marquee style properties set to their respective initial values.

Here is an example of how the **-wap-marquee** property value can be used to cause the text and a pictogram inside a paragraph to scroll horizontally:

```
<p style="display: -wap-marquee">
You have a <img wml:localsrc="pict:///core/message/message"
               src="http://www.pict.com/message.wbmp"
               alt="email"/>!
</p>
```

And here is one that causes the content of list items to scroll:

```
<style type="text/css">
.scroll { display: -wap-marquee;}
</style>

...

<ol>
 <li><div class="scroll">Item one</div></li>
 <li><div class="scroll">Item two</div></li>
</ol>
```

17.2. Marquee Style: '-wap-marquee-style'

'-wap-marquee-style'

Value: scroll | slide | alternate

Initial: scroll

Applies to: Marquee elements

Inherited: no

Percentages: N/A

Media: visual

Definition: A *marquee style* is one of the following:

scroll
Start completely off one side, scroll all the way across and completely off, and then start again.

slide
Start completely off one side, scroll in, and stop as soon as the text touches the other margin.

alternate
Bouncing back and forth within the screen.

Conformance Requirement: For each supported marquee style the user agent MUST support plain text and, if supported, pictograms. It MAY in addition support any other content permitted by the content model of the element for which the marquee property is specified.

Reference processing models for the different marquee types are specified in the following sections. The text is normative and takes precedence over the illustrations.

17.2.1. Scroll

Definition: The “scroll" behaviour is defined by the following reference processing model:

1. Set 'overflow' of the principal block to **hidden**.
2. Set 'white-space' of the marquee box to **nowrap**.
3. Set ‘width’ of the marquee box to a value based on the width of the boxes that it encloses.
4. Set the border and padding properties of the marquee box to **0px**.
5. Set the number of marquee loops to the value of ‘-wap-marquee-loop’.
6. For each marquee loop:
   [Repeat until the number of marquee loops is **zero**.]
   a. For right-to-left direction, decrease 'margin-left' of the marquee box property from **100%** to the negative value needed to ensure that the marquee box entirely overflows the principal block box. For left-to-right direction, use 'margin-right' instead.
   b. For right-to-left direction, maintain a constant width for the marquee box by animating 'margin-right' of the marquee box to compensate for changes in the value of 'margin-left'. For left-to-right direction, use 'margin-left' instead.
   c. Decrease the number of marquee loops by one.

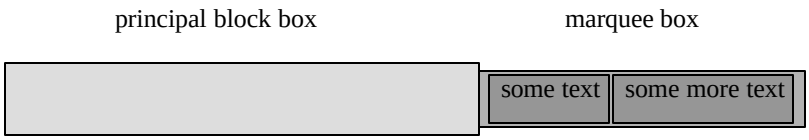
The added complexity of keeping a constant width for the marquee box is necessary in case descendent boxes are laid out with respect to the marquee box's width.

The figure below illustrates the three states for rendering the marquee element:

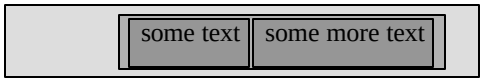
1. Begin loop state
2. Intermediate state
3. End loop state

The begin loop state describes the presentation at the beginning of each cycle. The intermediate state is where the element content is visible in the principal block box (although some of it may be overflowing the principal block box). The end loop state describes the presentation at the end of each cycle.

Begin Loop State (right-to-left direction)



Intermediate State



End Loop State (right-to-left direction)

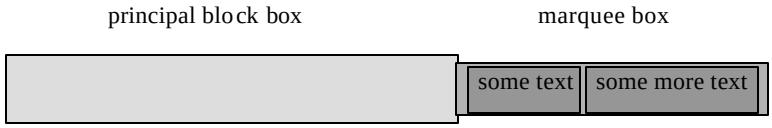
17.2.2. Slide

Definition: The “slide" behaviour is defined by the following reference processing model:

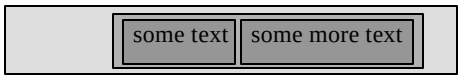
1. Set 'overflow' of the principal block box to **hidden**.
2. Set 'white-space' of the marquee box to **nowrap**.
3. Set ‘width’ of the marquee box to a value based on the width of the boxes that it encloses.
4. Set the border and padding properties of the marquee box to **0px**.
5. Set the number of marquee loops to the value of ‘-wap-marquee-loop’.
6. For each marquee loop:
   [Repeat until the number of marquee loops is **zero**.]
   a. For right-to-left direction, if ‘width’ of the marquee box is shorter than ‘width’ of the principal box, animate the marquee box's 'margin-left' from **100%** to **0%** . For left-to-right direction, use 'margin-right' instead.
   b. For right-to-left direction, if ‘width’ of the marquee box is longer than ‘width’ of the principal box, animate the marquee box's 'margin-left' from **100%** to the negative value needed to make ‘margin-right’ **0%** , and to guarantee that the whole element content can be presented on the screen. For a left-to-right direction, use 'margin-right' and ‘margin-left’, respectively, instead.
   c. For right-left direction, maintain a constant width for the marquee box by animating the marquee box's 'margin-right' to compensate for changes in the value of the 'margin-left'. For left-to-right direction, use 'margin-left' instead.
   d. Decrease the number of marquee loops by one.

The figure below illustrates the three states for rendering the marquee element:

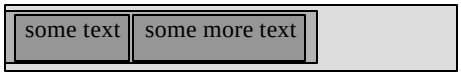
Begin Loop State (right-to-left direction)



Intermediate State



End Loop State (right-to-left direction, if the marquee box’s ‘width’ is shorter than the principal box’s ‘width’)



End Loop State (right-to-left direction, if the marquee box’s ‘width’ is longer than the principal box’s ‘width’)

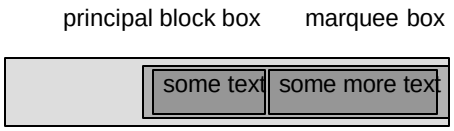
17.2.3. Alternate

Definition: The “alternate" behaviour is defined by the following reference processing model:

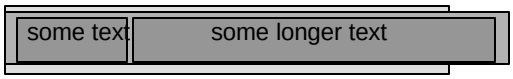
1. Set 'overflow' of the principal block box to **hidden**.
2. Set 'white-space' of the marquee box to **nowrap**.
3. Set the ‘width’ of the marquee box to a value based on the width of the boxes that it encloses.
4. Set the border and padding properties of the marquee box' to **0px**.
5. Set the number of marquee loops to the value of ‘-wap-marquee-loop’.
6. For each marquee loop:
   [Repeat until the number of marquee loops is **zero**.]
   a. For right-to-left direction, if the ‘width’ of the marquee box is shorter than the ‘width’ of the principal box, animate the marquee box's 'margin-right' from **0** to the value needed until the ‘margin-left’ property is animated to **0**. If the decreased number of marquee loops is a value greater than **0**, the marquee box will go back to right until ‘margin-right’ is animated to **0** and the number of marquee loops is decreased. For left-to-right direction, use 'margin-right' and ‘margin-left’ properties, respectively, instead.
   b. For right-to-left direction, if the ‘width’ of the marquee box is longer than the ‘width’ of the principal box, animate the marquee box's 'margin-left' from **0** to the value needed until ‘margin-right’ is animated to **0** to guarantee that the whole element content can be presented on the screen once. If the decreased number of marquee loops is a value greater than **0**, the marquee box will go back to right until ‘margin-left’ is animated to **0**, and the number of marquee loops is decreased. For left-to-right direction, use 'margin-right' and ‘margin-left’ properties, respectively, instead.
   c. Maintain a constant width for the marquee box by animating the marquee box's 'margin-right' property to compensate for changes in the value of the 'margin-left' property (in the case of right-left direction). The 'margin-left' property is similarly animated for a left-to-right direction.
   d. Decrease the number of marquee loops by one.
   e. Change direction for the following loop, so that loops alternate between left-to-right and right-to-left.

The figure below illustrates the three states for rendering the marquee element:

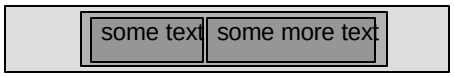
Begin Loop State (right-to-left direction, if the marquee box's 'width' is shorter than the principal box's 'width')



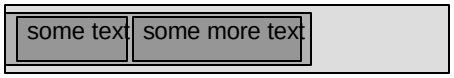
Begin Loop State (right-to-left direction, if the marquee box's 'width' is longer than the principal box's 'width')



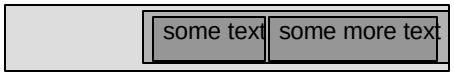
Intermediate State



End Loop State (right-to-left direction, if the number of loops is an odd number and if the marquee box's 'width' is shorter than the principal box's 'width')
)

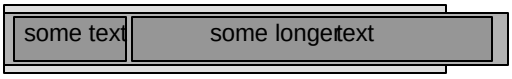


End Loop State (right-to-left direction, if the number of loops is an even number and if the marquee box's 'width' is shorter than the principal box's 'width')
)



End Loop State (right-to-left direction, if the number of loops is an odd number and if the marquee box's 'width' is longer than the principal box's 'width')
)

End Loop State (right-to-left direction, if the number of loops is an even number and if the marquee box's 'width' is longer than the principal box's 'width')
)



17.3. Marquee Loop: '-wap-marquee-loop'

‘-wap-marquee-loop’

Value: <integer> | infinite

Initial: 1

Applies to: Marquee elements

Inherited: no

Percentages: N/A

Media: visual

Definition: One *marquee loop* is the transition from the begin loop state to the end loop state. The initial value is "1" iterations. The maximum value is at least "16" iterations.

Conformance Requirement: After the user agent has looped the specified number of times, the element MUST be rendered as if the 'display' property was set to **block**.

Conformance Requirement: If the value is "0", no looping occurs and the element is displayed as if it had finished looping a specified number of times.

17.4. Marquee Direction: '-wap-marquee-dir'

‘-wap-marquee-dir’

Value: ltr | rtl

Initial: rtl

Applies to: Marquee elements

Inherited: no

Percentages: N/A

Media: visual

Definition: The *marquee direction* is the direction in which the marquee block should move across the principal block box. It is independent of the writing direction of blocks. The following two directions exists:

ltr
Left-to-right direction.

rtl
Right-to-left direction.

17.5. Marquee Speed: '-wap-marquee-speed'

‘-wap-marquee-speed’

Value: slow | normal | fast

Initial: normal

Applies to: Marquee elements

Inherited: no

Percentages: N/A

Media: visual

Definition: The *marquee speed* is how fast the marquee box moves across the principal block box. The **normal**, **slow**, and **fast** values refer to entries in a table of marquee speeds computed and kept by the user agent. The table may be different depending on the content type of the marquee element.

Conformance Requirement: For the marquee speed, the following relationship MUST always be true:

slow < **normal** < **fast**

18. WAP CSS Extension: Access Keys

This section is normative.

In order to support the WAP Access Keys CSS extension the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
WAPEXT-ACCESSKEYS-1	‘-wap-accesskey’	Section 18.1	M	

18.1. Access Keys: '-wap-accesskey'

'-wap-accesskey'

Value: none | <KeyCombination>+ | inherit

Initial: none

Applies to: all elements

Inherited: no

Percentages: N/A

Media: interactive

Definition: An *access key* is an additional, optional way to activate an element using a keypad key or key combination. What it means to *activate* an element depends on the type of element, and is specified for each element. For elements in WML, see the WML specification [WML2].

The actual list of supported keys and characters that may be used as access keys is platform dependent.

The access key is a key, not a function. The "mail" access key, for example, may be used on the device to launch the email application, but if used in the '-wap-accesskey' property, and assigned by the user agent, it will instead be used to activate the corresponding element - not launch the email application.

Definition: A *valid access key combination* matches the following syntax.

```
KeyCombination ::= Key ('-' Key)*

Key ::= 'space' | Char | SpecialKey

SpecialKey = ModifierKey | FunctionKey | NavigationKey | EditKey | MiscKey |
             VolumeControlKey | ApplicationKey | PhoneKey | VendorKey

ModifierKey = 'accesskey' | CmdKey | OptKey | CtrlKey | ShiftKey | AltKey |
             WinKey | MetaKey | 'fn' | 'fcn' | 'caps'

CmdKey ::= 'cmd' | 'rcmd' | 'lcmd'

OptKey ::= 'opt' | 'ropt' | 'lopt'

CtrlKey ::= 'ctrl' | 'rctrl' | 'lctrl'

ShiftKey ::= 'shift' | 'rshift' | 'lshift'

AltKey ::= 'alt' | 'ralt' | 'lalt'

WinKey ::= 'win' | 'rwin' | 'lwin'

MetaKey ::= 'meta' | 'rmeta' | 'lmeta'

FunctionKey ::= 'f1' | 'f2' | 'f3' | 'f4' | 'f5' | 'f6' | 'f7' | 'f8' | 'f9'
             | 'f10' | 'f11' | 'f12' | 'f13' | 'f14' | 'f15'

NavigationKey ::= 'tab' | 'esc' | 'enter' | 'return' | 'menu' | 'help' |
             'namemenu' | 'rcl' | 'snd' | ArrowKey | PageKey
```

```
ArrowKey ::= 'up' | 'down' | 'left' | 'right'

PageKey ::= 'home' | 'end' | 'pgup' | 'pgdn'

EditKey ::= 'bs' | 'del' | 'ins' | 'undo' | 'cut' | 'copy' | 'paste' | 'clr'
            | 'sto'

MiscKey ::= 'prtsc' | 'sysrq' | 'scrlock' | 'pause' | 'brk' | 'numlock' |
            'pwr'

VolumeControlKey ::= 'volumeup' | 'volumedown'

ApplicationKey ::= 'memo' | 'todo' | 'calendar' | 'mail' | 'address'

PhoneKey ::=  'phone-send' | 'phone-end' | 'phone-accept' /*See [GSM0230]*/

VendorKey ::= 'vnd.' Ident

Ident ::= See [CSS2], an identifier

Char ::= See [XML], a single Unicode character
```

Note: Many of the keys have been adopted from the W3C working draft “User Interface for CSS3” [UICSS3]. Keys common on wireless devices have been added.

Conformance Requirement: The user agent MUST ignore invalid access key combinations.

When more than one access key combination is specified they are separated by a space character or a “,” character. See section 18.1.2.

Access key names that start with “vnd.” are vendor specific. It is recommended that vendors should allocate names with the following format: vnd.<*name-of-vendor-specific-accesskey*>.

Since the actual user input for keys is case insensitive, single characters that represent keys must be specified in uppercase or as entities. Special or modifier keys are specified in all lowercase so as to be distinguished from the characters representing keys.

18.1.1. Access Key Availability and Assignment

Definition: An access key is *unavailable* for assignment to an element when it has already been assigned to another element, is used for some other purpose by the user agent, or is not supported by the platform.

Conformance Requirement: For elements whose access key is available, the user agent MUST use the specified access key and assign it to the element.

Conformance Requirement: For elements whose access key is unavailable, the user agent SHOULD select another key and assign it to the element. If no other key is available, the access key MUST be ignored.

Conformance Requirement: User agents SHOULD render the value of an access key in such a way as to emphasise its role and to distinguish it from other characters. For example, by underlining it, enclosing it in brackets, displaying it in reverse video or in a distinctive font.

> **Note**: The author should not refer to the specific access key values in content or in documentation, as the values as specified in the style rule may not be used.

18.1.2. Fallbacks and Multiple Assignments

Conformance Requirement: In a ","separated list of access keys the user agent MUST select the first available access key in the list for assignment, and ignore all other access keys in the list.

In the following example the user agent will first try to assign the "send" key to the element. If that key is unavailable it will try the "*" key, and if that is unavailable it will try the "#" key.

```
a { -wap-accesskey: send, *, # ;}
```

This is the same algorithm that is used for the CSS 'font-family' property.

Conformance Requirement: In a " " (space, as defined in [CSS2]) separated list of access keys the user agent MUST select all available access keys for assignment.

In the following example the user agent will assign both access keys ("send" and "*") if they both are available.

```
a { -wap-accesskey: send *;}
```

The " " (space) and the "," operators may be used together.

In the following example the user aget will first try to assign the "send" and "*" keys if both are available, otherwise it will try to assign the "#" key.

```
a { -wap-accesskey: send *, #;}
```

18.1.3. The XHTML 'accesskey' Attribute

The following conformance requirement is derived from the text in [CSS2] section 6.4.4 that specifies how non-CSS presentational hints are used in the CSS cascade.

Conformance Requirement: If the XHTML 'accesskey' attribute is present on an element it MUST be translated into the '-wap-accesskey' rule with specificity equal to zero. The rule is assumed to be at the start of the author style sheet and may be overridden by subsequent style sheet rules.

This means that the '-wap-accesskey' property will override the 'accesskey' attribute, if both are present on the same element.

19. WAP CSS Extension: Input

This section is normative.

> **Note**: Input formatting and input validation are not features currently implemented as CSS2 [CSS2] properties.

In order to support the WAP Input CSS extensions the user agent MUST implement all mandatory items (“M”) in the following table.

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
WAPEXT-INPUT-GENERAL-1	General concepts	Section 19.1	M	
WAPEXT-INPUT-FORMAT-2	‘-wap-input-format’	Section 19.2	M	
WAPEXT-INPUT-REQUIRED-3	‘-wap-input-required’	Section 19.3	M	

19.1. General Concepts

Definition: An *input element* is an XML element that represents a UI control that accepts text input from the user. Typically the element represents a form control such as a textbox.

> **Note**: In XHTML Basic and WML 2.0 the following elements accept text input from the user: textarea and input.

Conformance Requirement: If a WAP Input property is applied to a non-input element, the element MUST be unaffected by the property.

Definition: The *input type* is the type of data that the input element accepts. The default data type is a character string. The value space is the set of legal finite-length sequences of characters (in the character set of the document).

Definition: The *input value* is the data from the user.

Definition: An input value that is not in the value space of the input element is said to be *invalid*.

Conformance Requirement: The user agent SHOULD reject invalid input values and SHOULD request a new value from the user.

The user agent can use knowledge of the value space to optimise the user interface, for example changing the current input mode according to the specified value space.

19.2. Input Format: '-wap-input-format'

'-wap-input-format'

Value: <format>

Initial: "*M"

Applies to: Input elements

Inherited: no

Percentages: N/A

Media: interactive

Definition: An *input format* is a sequence of characters that denotes a set of strings, L (F). The input format constrains the value space of the input data type. Strings in L (F) are valid input values for the input element to which the property is applied.

The <format> value is a *string* as defined in [CSS2] section 4. This means that input formats must be quoted.

Conformance Requirement: The user agent MUST ignore input masks that do not match the "input mask" syntax defined in [WML2].

Note: The "input mask" syntax is case sensitive.

Here are some examples of input formats:

```
.phonenumber { -wap-input-format: "NNNNN\-3N"; }

.pincode { -wap-input-format: "NNNN"; }
```

19.2.1. The WML 'format' Attribute

The following conformance requirement is derived from the text in [CSS2] section 6.4.4 that specifies how non-CSS presentational hints are used in the CSS cascade.

Conformance Requirement: If the WML 'format' attribute is present on an element it MUST be translated into the '-wap-input-format' rule with specificity equal to zero. The rule is assumed to be at the start of the author style sheet and may be overridden by subsequent style sheet rules.

This means that the '-wap-input-format' property will override the 'format' attribute, if both are present on the same element.

19.3. Required Input: '-wap-input-required'

'-wap-input-required'

Value: true | false

Initial: none

Applies to: Input elements

Inherited: no

Percentages: N/A

Media: interactive

Definition: The value space for *required input* is constrained to non-zero length strings.

Here is an example of using required input:

```
input { -wap-input-required: true; }
```

19.3.1. The WAP CSS '-wap-input-format' Property

The '-wap-input-required' property is combined with '-wap-input-format' property in order to determine the set of valid values of the input element.

Conformance Requirement: When '-wap-input-required' has the value 'false', the zero-length string is valid regardless of what is specified in '-wap-input-format'. When the value is 'true', the zero-length string is invalid regardless of what is specified in '-wap-input-format'. If the author does not explicitly specify a value for the '-wap-input-required' property, then the value "none" is applied, and the '-wap-input-format' property fully defines the set of valid input values. This is illustrated in table 1.

> **Note**: The '-wap-input-required' property cannot be set to the value 'none' in any other way than not specifying the property.

The '-wap-input-required' property takes precedence over the '-wap-input-format' property when both are applied to the same element. If the '-wap-input-format' allows for zero-length string as a valid input value, then, if input is required according to the '-wap-input-required' property, it is anyway an illegal value.

		-wap-input-required		
		true	**false**	**none**
-wap-input-format	**Property allows for a zero-length string (e.g. *M)**	According to the format but zero-length string is not allowed (e.g. M*M).	According to the format (e.g. *M).	According to the format (e.g. *M).
	Property requires a non-zero-length string (e.g. NNN)	According to the format (e.g. NNN).	According to the format but zero-length string also allowed.	According to the format (e.g. NNN).

Table 1Combinations of '-wap-input-format' and '-wap-input-required'.

19.3.2. The WML ‘emptyok’ Attribute

The following conformance requirement is derived from the text in [CSS2] section 6.4.4 that specifies how non-CSS presentational hints are used in the CSS cascade.

Conformance Requirement: If the WML ‘emptyok’ attribute is present on an element it MUST be translated into the ‘-wap-input-required’ rule with specificity equal to zero. The rule is assumed to be at the start of the author style sheet and may be overridden by subsequent style sheet rules.

This means that the ‘-wap-input-required’ property will override the ‘emptyok’ attribute, if both are present on the same element.

20. Associating CSS Style Sheets With XHTML Mobile

This section is informative.

It is recommended that content authors associate WML documents with external WAP CSS style sheets using the XML style sheet processing instruction in the following way:

```
<?xml version=”1.0” ?>
<?xml-stylesheet href=”mobilestyle.css” media=”handheld” type=”text/css” ?>
<!DOCTYPE html PUBLIC “-//WAPFORUM//DTD XHTML MOBILE 1.0//EN” “xhtml-
mobile10.dtd”>
<html>
  <head><title>Welcome to XHTML Mobile</title></head>
  <body>
   ..
  </body>
</html>
```

It is also recommended that authors use the ‘handheld’ media type to indicate that the style sheet is appropriate for handheld devices.

> **Note:** Since WAP CSS is based on CSS2 it does not support XML namespaces. The next version of WAP CSS will support XML namespaces as a part of “CSS3 Module: W3C Selectors” [SELECTORS]. At the time of writing, this specification is still a W3C Working Draft.

Appendix A. Static Conformance Requirements (Normative)

In order to support WAP CSS the user agent MUST implement all mandatory items (“M”) in the following table, and SHOULD implement all optional items (“O”).

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
WCSS-SELECTOR-C-001	Pattern matching	Section 5.1	M	
WCSS-SYNTAX-C-001	Syntax	Section 6.1	M	
WCSS-TYPES-C-001	Data types	Section 6.2	M	
WCSS-CASC-C-001	Assigning property values, cascading, and inheritance	Section 7	M	
WCSS-MEDIA-C-001	Media types	Section 8	M	
WCSS-ASOC-C-001	Associating style sheets with XML documents	Section 9	M	
WCSS-MARGIN-C-001	Margin properties	Section 10.1	O	
WCSS-PADDING-C-001	Padding properties	Section 10.2	O	
WCSS-BORDER-WIDTH-C-001	Border width	Section 10.3	O	
WCSS-BORDER-COLOR-C-001	Border colour	Section 10.4	O	
WCSS-BORDER-STYLE-C-001	Border style	Section 10.5	O	
WCSS-BORDER-SHORT-C-001	Border shorthand property	Section 10.6	O	
WCSS-FCOLOR-C-001	Foreground colour	Section 11.1	O	
WCSS-BCOLOR-C-001	Background colour	Section 11.2	O	
WCSS-BIMAGES-C-001	Background images	Section 11.3	O	
WCSS-FONTS-FAMILY-C-001	Font family	Section 12.1	O	
WCSS-FONTS-STYLE-C-001	Font style	Section 12.2	O	

ITEM	FUNCTION	REFERENCE	STATUS	REQUIREMENT
WCSS-FONTS-VARIANT-C-001	Font variant	Section 12.3	O	
WCSS-FONTS-WEIGHT-C-001	Font weight	Section 12.4	O	
WCSS-FONTS-SIZE-C-001	Font size	Section 12.5	O	
WCSS-FONTS-SHORT-C-001	Shorthand font property	Section 12.6	O	
WCSS-LISTS-C-001	Lists	Section 13.1	O	
WCSS-TEXT-INDENT-C-001	Text indentation	Section 14.1	O	
WCSS-TEXT-ALIGN-C-001	Text alignment	Section 14.2	O	
WCSS-TEXT-DEC-C-001	Text decoration	Section 14.3	O	
WCSS-TEXT-TRANS-C-001	Text transformation	Section 14.4	O	
WCSS-TEXT-WS-C-001	White space	Section 14.5	O	
WCSS-VEFFECT-C-001	Visual effects	Section 15	O	
WCSS-VFORM-DISP-C-001	Display properties	Section 16.1	O	
WCSS-VFORM-FLOAT-C-001	Float Positioning	Section 16.2	O	
WCSS-VFORM-CLEAR-C-001	Float Flow Control	Section 16.3	O	
WCSS-VFORM-SIZE-C-001	Content width and height	Section 16.4	O	
WCSS-WAPEXT-MARQUEE-C-001	WAP CSS Extension: Marquee	Section 17	O	
WCSS-WAPEXT-ACC-C-001	WAP CSS Extension: Access keys	Section 18	O	
WCSS-WAPEXT-INPUT-C-001	WAP CSS Extension: Input	Section 19	O	

Appendix B. Change History (Informative)

Type of Change	Date	Section	Description
Class 0	26-Oct-2001		The initial version of this document.