Cascading Style Sheets (CSS) Snapshot 2010

W3C Working Group Note 12 May 2011

This version:
http://www.w3.org/TR/2011/NOTE-css-2010-20110512/
Latest version:
http://www.w3.org/TR/css-2010/
Previous versions:
http://www.w3.org/TR/2010/WD-css-2010-20101202/
Editor:
Elika J. Etemad

Copyright © 2011 W3C® (MIT, ERCIM, Keio), All Rights Reserved. W3C liability, trademark and document use rules apply.

Abstract

This document collects together into one definition all the specs that together form the current state of Cascading Style Sheets (CSS) as of 2010. The primary audience is CSS implementors, not CSS authors, as this definition includes modules by specification stability, not Web browser adoption rate.

Status of this document

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

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

The document was produced by the CSS Working Group (part of the Style Activity).

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

The (archived) public mailing list www-style@w3.org (see instructions) is preferred for discussion of this document. When sending e-mail, please put the text “css-2010” in the subject, preferably like this: “[css-2010] *…summary of comment…*”

This document represents the state of CSS as of 2010. The CSS Working Group does not expect any further changes to this document: new snapshots will be published at http://www.w3.org/TR/CSS/ as CSS advances.

Table of contents

1. Introduction

When the first CSS specification was published, all of CSS was contained in one document that defined CSS Level 1. CSS Level 2 was defined also by a single, multi-chapter document. However for CSS beyond Level 2, the CSS Working Group chose to adopt a modular approach, where each module defines a part of CSS, rather than to define a single monolithic specification. This breaks the specification into more manageable chunks and allows more immediate, incremental improvement to CSS.

Since different CSS modules are at different levels of stability, the CSS Working Group has chosen to publish this profile to define the current scope and state of Cascading Style Sheets as of late 2010. This profile includes only specifications that we consider stable *and* for which we have enough implementation experience that we are sure of that stability.

Note that this is not intended to be a CSS Desktop Browser Profile: inclusion in this profile is based on feature stability only and not on expected use or Web browser adoption. This profile defines CSS in its most complete form.

Note also that although we don't anticipate significant changes to the specifications that form this snapshot, their inclusion does are not mean they are frozen. The Working Group will continue to address problems as they are found in these specs. Implementers should monitor www-style and/or the CSS Working Group Blog for any resulting changes, corrections, or clarifications.

1.1. The W3C Process and CSS

This section is non-normative.

In the W3C Process, a Recommendation-track document passes through five levels of stability, summarized below:

Working Draft (WD)
: Published during the process of drafting the specification, the purpose of a public Working Draft is to create a snapshot of the specification's current state and to solicit input from the W3C and the public. The document is known to be unstable, and is often incomplete.

Last Call Working Draft (LC or LCWD)

By publishing a Last Call Working Draft, a working group is expressing that they consider the spec to be complete and all issues to be resolved. Publishing a Last Call Working Draft announces that this specification will move toward Candidate Recommendation unless significant issues are brought up. The Last Call period is a last chance for others to submit issues before the transition to CR.

Candidate Recommendation (CR)

By publishing a Candidate Recommendation, a working group is expressing that have resolved all known issues and they believe the spec is ready for implementation.

Proposed Recommendation (PR)

To exit CR and enter this stage, the spec needs a comprehensive test suite and implementation reports proving that every feature in the spec is interoperably implemented in at least two shipping implementations. Entering the Proposed Recommendation stage signals to the W3C that these requirements have been met. Once the W3C officially approves the specification, it becomes a Recommendation.

Recommendation (REC)

This is the final stage. At this point there should need to be no more changes.

In the CSSWG's experience, the recommendation track is not linear. The wider review triggered by an LCWD often results in at least another working draft, possibly several. More significantly, our experience is that many specs enter CR twice, because implementation testing often uncovers significant problems in the spec and thus pushes it back to working draft. Additionally, fixing even minor problems forces a CR to re-enter the Working Draft stage. As a result, although the CSSWG has a clear idea of the stability of the CSS specs, it is very difficult for someone outside the working group to come to that same understanding based on a specification's official status. The CSS Working Group's motivation for creating this document is thus to communicate to others our understanding of the state of CSS.

2. CSS Levels

Cascading Style Sheets does not have versions in the traditional sense; instead it has ***levels***. Each level of CSS builds on the previous, refining definitions and adding features. The feature set of each higher level is a superset of any lower level, and the behavior allowed for a given feature in a higher level is a subset of that allowed in the lower levels. A user agent conforming to a higher level of CSS is thus also conformant to all lower levels.

2.1. CSS Level 1

The CSS Working Group considers the CSS1 specification to be obsolete. ***CSS Level 1*** is defined as all the features defined in the CSS1 specification (properties, values, at-rules, etc), but using the syntax and definitions in the CSS2.1 specification. CSS Style Attributes defines its inclusion in element-specific style attributes.

2.2. CSS Level 2

Although the CSS2 specification is technically a W3C Recommendation, it passed into the Recommendation stage before the W3C had defined the Candidate Recommendation stage. Over time implementation experience and further review has brought to light many problems in the CSS2 specification, so instead of expanding an already unwieldy errata list, the CSS Working Group chose to define *CSS Level 2 Revision 1* (CSS2.1). In case of any conflict between the two specs CSS2.1 contains the definitive definition.

Once CSS2.1 became Candidate Recommendation—effectively though not officially the same level of stability as CSS2—obsoleted the CSS2 Recommendation. Features in CSS2 that were dropped from CSS2.1 should be considered to be at the Candidate Recommendation stage, but note that many of these have been or will be pulled into a CSS Level 3 working draft, in which case that specification will, once it reaches CR, obsolete the definitions in CSS2.

The CSS2.1 specification defines ***CSS Level 2*** and the CSS Style Attributes specification defines its inclusion in element-specific style attributes.

2.3. CSS Level 3

This section is non-normative.

CSS Level 3 builds on CSS Level 2 module by module, using the CSS2.1 specification as its core. Each module adds functionality and/or replaces part of the CSS2.1 specification. The CSS Working Group intends that the new CSS modules will not contradict the CSS2.1 specification: only that they will add functionality and refine definitions. As each module is completed, it will be plugged in to the existing system of CSS2.1 plus previously-completed modules.

From this level on modules are levelled independently: for example Selectors Level 4 may well be defined before CSS Line Module Level 3.

3. Cascading Style Sheets Definition

As of 2010, ***Cascading Style Sheets (CSS)*** is defined by the following specifications.

1. CSS Level 2 Revision 1 (including errata)
2. CSS Style Attributes
3. Media Queries Level 3
4. CSS Namespaces
5. Selectors Level 3
6. CSS Color Level 3

3.1. Partial Implementations

So that authors can exploit the forward-compatible parsing rules to assign fallback values, CSS renderers **must** treat as invalid (and ignore as appropriate) any at-rules, properties, property values, keywords, and other syntactic constructs for which they have no usable level of support. In particular, user agents **must not** selectively ignore unsupported property values and honor supported values in a single multi-value property declaration: if any value is considered invalid (as unsupported values must be), CSS requires that the entire declaration be ignored.

3.2. CSS Profiles

Not all implementations will implement all functionality defined in CSS. For example, an implementation may choose to implement only the functionality required by a CSS Profile. Profiles define a subset of CSS considered fundamental for a specific class of CSS implementations. The W3C CSS Working Group defines the following CSS profiles:

- CSS Mobile Profile 2.0
- CSS Print Profile 1.0
- CSS TV Profile 1.0

3.3. Experimental Implementations

To avoid clashes with future CSS features, the CSS2.1 specification reserves a prefixed syntax for proprietary and experimental extensions to CSS.

Prior to a specification reaching the Candidate Recommendation stage in the W3C process, all implementations of a CSS feature are considered experimental. The CSS Working Group recommends that implementations use a vendor-prefixed syntax for such features, including those in W3C Working Drafts. This avoids incompatibilities with future changes in the draft.

► For legacy reasons, certain experimental CSS properties do not follow this prefixing convention. Two common examples are the ‘word-wrap’ and ‘text-overflow’ properties, which were introduced unprefixed by Microsoft Internet Explorer prior to the introduction of the vendor prefixing policy in CSS2.1 and were subsequently implemented unprefixed by other browsers, creating a dependency on the unprefixed names despite the lack of a W3C spec. Any other legacy exceptions should be made in consultation with the CSS Working Group.

3.4. Non-Experimental Implementations

Once a specification reaches the Candidate Recommendation stage, non-experimental implementations are possible, and implementors should release an unprefixed implementation of any CR-level feature they can demonstrate to be correctly implemented according to spec.

To establish and maintain the interoperability of CSS across implementations, the CSS Working Group requests that non-experimental CSS renderers submit an implementation report (and, if necessary, the testcases used for that implementation report) to the W3C before releasing an unprefixed implementation of any CSS features. Testcases submitted to W3C are subject to review and correction by the CSS Working Group.

Further information on submitting testcases and implementation reports can be found from on the CSS Working Group's website at http://www.w3.org/Style/CSS/Test/. Questions should be directed to the public-css-testsuite@w3.org mailing list.

CSS2.1 implementations are encouraged, but not required, to submit an implementation report.

4. Indices

These sections are non-normative.

4.1. Property Index

Name	Values	Initial value	Applies to	Inherited?	Percentages	Media
background-attachment	scroll | fixed | inherit	scroll		no		visual
background-color	<color> | inherit	transparent		no		visual
background-image	<uri> | none | inherit	none		no		visual
background-position	[[<percentage> | <length> | left | center | right] [<percentage> | <length> | top | center | bottom]?] | [[left | center | right] || [top | center | bottom]] | inherit	0% 0%		no	refer to the size of the box itself	visual
background-repeat	repeat | repeat-x | repeat-y | no-repeat | inherit	repeat		no		visual
background	['background-color' || 'background-image' || 'background-repeat' || 'background-attachment' || background-position] | inherit	see individual properties		no	allowed on 'background-position'	visual
border-collapse	collapse | separate | inherit	separate	'table' and 'inline-table' elements	yes		visual
border-color	[<color>]{1,4} | inherit	see individual properties		no		visual
border-spacing	<length> <length>? | inherit	0	'table' and 'inline-table' elements	yes		visual
border-style	<border-style>{1,4} | inherit	see individual properties		no		visual
border-top **border-right** **border-bottom** **border-left**	[<border-width> || <border-style> || border-top-color] | inherit	see individual properties		no		visual
border-top-color	<color> | inherit	the value of		no		visual

Name	Values	Initial value	Applies to	Inherited?	Percentages	Media
border-right-color border-bottom-color border-left-color		the 'color' property				
border-top-style border-right-style border-bottom-style border-left-style	<border-style> | inherit	none		no		visual
border-top-width border-right-width border-bottom-width border-left-width	<border-width> | inherit	medium		no		visual
border-width	<border-width>{1,4} | inherit	see individual properties		no		visual
border	[<border-width> || <border-style> || border-top-color] | inherit	see individual properties		no		visual
bottom	<length> | <percentage> | auto | inherit	auto	positioned elements	no	refer to height of containing block	visual
caption-side	top | bottom | inherit	top	'table-caption' elements	yes		visual
clear	none | left | right | both | inherit	none	block-level elements	no		visual
clip	<shape> | auto | inherit	auto	absolutely positioned elements	no		visual
color	<color> | inherit	depends on user agent		yes		visual
content	normal | none | [<string> | <uri> | <counter> | attr(<identifier>) | open-quote | close-quote | no-open-quote | no-close-quote]+ | inherit	normal	:before and :after pseudo-elements	no		all
counter-increment	[<identifier> <integer>?]+ | none | inherit	none		no		all

Name	Values	Initial value	Applies to	Inherited?	Percentages	Media
counter-reset	[<identifier> <integer>?]+ | none | inherit	none		no		all
cursor	[[<uri> ,]* [auto | crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize | s-resize | w-resize | text | wait | help | progress]] | inherit	auto		yes		visual, interactive
direction	ltr | rtl | inherit	ltr	all elements, but see prose	yes		visual
display	inline | block | list-item | inline-block | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | none | inherit	inline		no		all
empty-cells	show | hide | inherit	show	'table-cell' elements	yes		visual
float	left | right | none | inherit	none	all, but see 9.7	no		visual
font-family	[[<family-name> | <generic-family>] [, <family-name>| <generic-family>]*] | inherit	depends on user agent		yes		visual
font-size	<absolute-size> | <relative-size> | <length> | <percentage> | inherit	medium		yes	refer to inherited font size	visual
font-style	normal | italic | oblique | inherit	normal		yes		visual
font-variant	normal | small-caps | inherit	normal		yes		visual
font-weight	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit	normal		yes		visual

Name	Values	Initial value	Applies to	Inherited?	Percentages	Media
font	[[‘font-style’ || ‘font-variant’ || ‘font-weight’]? ‘font-size’ [/ ‘line-height’]? font-family] | caption | icon | menu | message-box | small-caption | status-bar | inherit	see individual properties		yes	see individual properties	visual
height	<length> | <percentage> | auto | inherit	auto	all elements but non-replaced inline elements, table columns, and column groups	no	see prose	visual
left	<length> | <percentage> | auto | inherit	auto	positioned elements	no	refer to width of containing block	visual
letter-spacing	normal | <length> | inherit	normal		yes		visual
line-height	normal | <number> | <length> | <percentage> | inherit	normal		yes	refer to the font size of the element itself	visual
list-style-image	<uri> | none | inherit	none	elements with ‘display: list-item’	yes		visual
list-style-position	inside | outside | inherit	outside	elements with ‘display: list-item’	yes		visual
list-style-type	disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-latin | upper-latin | armenian | georgian | lower-alpha | upper-alpha | none | inherit	disc	elements with ‘display: list-item’	yes		visual
list-style	[‘list-style-type’ || ‘list-style-position’ || list-style-image] | inherit	see individual properties	elements with ‘display: list-item’	yes		visual
margin-right margin-left	<margin-width> | inherit	0	all elements except elements with table display types other than table-caption,	no	refer to width of containing block	visual

Name	Values	Initial value	Applies to	Inherited?	Percentages	Media
			table and inline-table			
margin-top margin-bottom	<margin-width> | inherit	0	all elements except elements with table display types other than table-caption, table and inline-table	no	refer to width of containing block	visual
margin	<margin-width>{1,4} | inherit	see individual properties	all elements except elements with table display types other than table-caption, table and inline-table	no	refer to width of containing block	visual
max-height	<length> | <percentage> | none | inherit	none	all elements but non-replaced inline elements, table columns, and column groups	no	see prose	visual
max-width	<length> | <percentage> | none | inherit	none	all elements but non-replaced inline elements, table rows, and row groups	no	refer to width of containing block	visual
min-height	<length> | <percentage> | inherit	0	all elements but non-replaced inline elements, table columns, and column groups	no	see prose	visual
min-width	<length> | <percentage> | inherit	0	all elements but non-replaced inline elements, table rows, and row groups	no	refer to width of containing block	visual
opacity	<number> | inherit	1	all	no		visual
orphans	<integer> | inherit	2	block container elements	yes		visual, paged

Name	Values	Initial value	Applies to	Inherited?	Percentages	Media
outline-color	<color> | invert | inherit	invert		no		visual, interactive
outline-style	<border-style> | inherit	none		no		visual, interactive
outline-width	<border-width> | inherit	medium		no		visual, interactive
outline	[‘outline-color’ || ‘outline-style’ || outline-width] | inherit	see individual properties		no		visual, interactive
overflow	visible | hidden | scroll | auto | inherit	visible	block containers	no		visual
padding-top padding-right padding-bottom padding-left	<padding-width> | inherit	0	all elements except table-row-group, table-header-group, table-footer-group, table-row, table-column-group and table-column	no	refer to width of containing block	visual
padding	<padding-width>{1,4} | inherit	see individual properties	all elements except table-row-group, table-header-group, table-footer-group, table-row, table-column-group and table-column	no	refer to width of containing block	visual
page-break-after	auto | always | avoid | left | right | inherit	auto	block-level elements (but see text)	no		visual, paged
page-break-before	auto | always | avoid | left | right | inherit	auto	block-level elements (but see text)	no		visual, paged
page-break-inside	avoid | auto | inherit	auto	block-level elements (but see text)	no		visual, paged
position	static | relative | absolute | fixed | inherit	static		no		visual

Name	Values	Initial value	Applies to	Inherited?	Percentages	Media
quotes	[<string> <string>]+ | none | inherit	depends on user agent		yes		visual
right	<length> | <percentage> | auto | inherit	auto	positioned elements	no	refer to width of containing block	visual
table-layout	auto | fixed | inherit	auto	'table' and 'inline-table' elements	no		visual
text-align	left | right | center | justify | inherit	a nameless value that acts as 'left' if 'direction' is 'ltr', 'right' if 'direction' is 'rtl'	block containers	yes		visual
text-decoration	none | [underline || overline || line-through || blink] | inherit	none		no (see prose)		visual
text-indent	<length> | <percentage> | inherit	0	block containers	yes	refer to width of containing block	visual
text-transform	capitalize | uppercase | lowercase | none | inherit	none		yes		visual
top	<length> | <percentage> | auto | inherit	auto	positioned elements	no	refer to height of containing block	visual
unicode-bidi	normal | embed | bidi-override | inherit	normal	all elements, but see prose	no		visual
vertical-align	baseline | sub | super | top | text-top | middle | bottom | text-bottom | <percentage> | <length> | inherit	baseline	inline-level and 'table-cell' elements	no	refer to the 'line-height' of the element itself	visual
visibility	visible | hidden | collapse | inherit	visible		yes		visual
white-space	normal | pre | nowrap | pre-wrap | pre-line | inherit	normal		yes		visual

Name	Values	Initial value	Applies to	Inherited?	Percentages	Media
widows	<integer> | inherit	2	block container elements	yes		visual, paged
width	<length> | <percentage> | auto | inherit	auto	all elements but non-replaced inline elements, table rows, and row groups	no	refer to width of containing block	visual
word-spacing	normal | <length> | inherit	normal		yes		visual
z-index	auto | <integer> | inherit	auto	positioned elements	no		visual

4.2. Selector Index

Pattern	Meaning	Described in section	First defined in level
*****	any element	Universal selector	2
E	an element of type E	Type selector	1
E[foo]	an E element with a "foo" attribute	Attribute selectors	2
E[foo="bar"]	an E element whose "foo" attribute value is exactly equal to "bar"	Attribute selectors	2
E[foo~="bar"]	an E element whose "foo" attribute value is a list of whitespace-separated values, one of which is exactly equal to "bar"	Attribute selectors	2
E[foo^="bar"]	an E element whose "foo" attribute value begins exactly with the string "bar"	Attribute selectors	3
E[foo$="bar"]	an E element whose "foo" attribute value ends exactly with the string "bar"	Attribute selectors	3
E[foo*="bar"]	an E element whose "foo" attribute value contains the substring "bar"	Attribute selectors	3
E[foo|="en"]	an E element whose "foo" attribute has a hyphen-separated list of values beginning (from the left) with "en"	Attribute selectors	2
E:root	an E element, root of the document	Structural pseudo-classes	3
E:nth-child(n)	an E element, the n-th child of its parent	Structural pseudo-classes	3

Pattern	Meaning	Described in section	First defined in level
E:nth-last-child(n)	an E element, the n-th child of its parent, counting from the last one	Structural pseudo-classes	3
E:nth-of-type(n)	an E element, the n-th sibling of its type	Structural pseudo-classes	3
E:nth-last-of-type(n)	an E element, the n-th sibling of its type, counting from the last one	Structural pseudo-classes	3
E:first-child	an E element, first child of its parent	Structural pseudo-classes	2
E:last-child	an E element, last child of its parent	Structural pseudo-classes	3
E:first-of-type	an E element, first sibling of its type	Structural pseudo-classes	3
E:last-of-type	an E element, last sibling of its type	Structural pseudo-classes	3
E:only-child	an E element, only child of its parent	Structural pseudo-classes	3
E:only-of-type	an E element, only sibling of its type	Structural pseudo-classes	3
E:empty	an E element that has no children (including text nodes)	Structural pseudo-classes	3
E:link **E:visited**	an E element being the source anchor of a hyperlink of which the target is not yet visited (:link) or already visited (:visited)	The link pseudo-classes	1
E:active **E:hover** **E:focus**	an E element during certain user actions	The user action pseudo-classes	1 and 2
E:target	an E element being the target of the referring URI	The target pseudo-class	3
E:lang(fr)	an element of type E in language "fr" (the document language specifies how language is determined)	The :lang() pseudo-class	2
E:enabled **E:disabled**	a user interface element E which is enabled or disabled	The UI element states pseudo-classes	3
E:checked	a user interface element E which is checked (for instance a radio-button or checkbox)	The UI element states pseudo-classes	3
E::first-line	the first formatted line of an E element	The ::first-line pseudo-element	1

Pattern	Meaning	Described in section	First defined in level
E::first-letter	the first formatted letter of an E element	The ::first-letter pseudo-element	1
E::before	generated content before an E element	The ::before pseudo-element	2
E::after	generated content after an E element	The ::after pseudo-element	2
E.warning	an E element whose class is "warning" (the document language specifies how class is determined).	Class selectors	1
E#myid	an E element with ID equal to "myid".	ID selectors	1
E:not(s)	an E element that does not match simple selector s	Negation pseudo-class	3
E F	an F element descendant of an E element	Descendant combinator	1
E > F	an F element child of an E element	Child combinator	2
E + F	an F element immediately preceded by an E element	Adjacent sibling combinator	2
E ~ F	an F element preceded by an E element	General sibling combinator	3

4.3. At-Rule Index

- @charset
- @import, with the media list replaced by a media query list
- @media, with the media list replaced by a media query list
- @page