Media Queries Level 3

Editor's Draft 13 April 2012

This Version:
http://www.w3.org/TR/2012/ED-css3-mediaqueries-20120413/

Latest Version:
http://www.w3.org/TR/css3-mediaqueries/

Latest Editor Version:
http://dev.w3.org/csswg/css3-mediaqueries/

Previous Version:
http://www.w3.org/TR/2010/CR-css3-mediaqueries-20100727/

Disposition of Comments:
http://www.w3.org/Style/2012/MediaQueriesDisposalOfComments.html

Editors:
Florian Rivoal <florianr @opera.com>

Previous Editors:
Håkon Wium Lie <howcome @opera.com>
Tantek Çelik <tantek @cs.stanford.edu>
Daniel Glazman <daniel.glazman @disruptive-innovations.com>
Anne van Kesteren <annevk@opera.com>

Abstract

HTML4 and CSS2 currently support media-dependent style sheets tailored for different *media types*. For example, a document may use sans-serif fonts when displayed on a screen and serif fonts when printed. '`screen`' and '`print`' are two media types that have been defined. *Media queries* extend the functionality of media types by allowing more precise labeling of style sheets.

A media query consists of a media type and zero or more expressions that check for the conditions of particular *media features*. Among the media features that can be used in media queries are '`width`', '`height`', and '`color`'. By using media queries, presentations can be tailored to a specific range of output devices without changing the content itself.

Status of this document

This is a public copy of the editors' draft. It is provided for discussion only and may change at any moment. Its publication here does not imply endorsement of its contents by W3C. Don't cite this document other than as work in progress.

The (archived) public mailing list www-style@w3.org (see instructions) is preferred for discussion of this specification. When sending e-mail, please put the text "css3-mediaqueries" in the subject, preferably like this: "[css3-mediaqueries] *…summary of comment…*"

This 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.

For this specification to exit the CR stage, the following conditions shall be met:

1. There must be at least two interoperable implementations. For the purposes of this criterion, we define the following terms:

   interoperable
   : passing the respective test case(s) in the CSS test suite, or, if the implementation is not a Web browser, an equivalent test. Every relevant test in the test suite should have an equivalent test created if such a user agent (UA) is to be used to claim interoperability. In addition if such a UA is to be used to claim interoperability, then there must one or more additional UAs which can also pass those equivalent tests in the same way for the purpose of interoperability. The equivalent tests must be made publicly available for the purposes of peer review.

   implementation
   : a user agent which:

    1. implements the specification.
    2. is available (i.e. publicly downloadable or available through some other public point of sale mechanism). This is the "show me" requirement.
    3. is shipped, or is a "nightly build" (i.e., a development version for the next release), but is not experimental (i.e., a version specifically designed to pass the test suite and not intended for daily usage going forward).
2. There must be a Test Suite.
3. A minimum of another four weeks of the CR period must elapse. That is, this specification will not exit CR before 24 August 2010. When the specification exits CR, an implementation report will be published. At this point, no such report exists.

Table of contents

1. Background

(This section is not normative.)

HTML4 [HTML401] and CSS2 [CSS21] currently support media-dependent style sheets tailored for different media types. For example, a document may use different style sheets for screen and print. In HTML4, this can be written as:

```
<link rel="stylesheet" type="text/css" media="screen" href="sans-
serif.css">
<link rel="stylesheet" type="text/css" media="print" href="serif.css">
```

Inside a CSS style sheet, one can declare that sections apply to certain media types:

```
@media screen {
  * { font-family: sans-serif }
}
```

The ‘`print`’ and ‘`screen`’ media types are defined in HTML4. The complete list of media types in HTML4 is: ‘`aural`’, ‘`braille`’, ‘`handheld`’, ‘`print`’, ‘`projection`’, ‘`screen`’, ‘`tty`’, ‘`tv`’. CSS2 defines the same list, deprecates ‘`aural`’ and adds ‘`embossed`’ and ‘`speech`’. Also, ‘`all`’ is used to indicate that the style sheet applies to all media types.

Media-specific style sheets are supported by several user agents. The most commonly used feature is to distinguish between ‘`screen`’ and ‘`print`’.

There have been requests for ways to describe in more detail what type of output devices a style sheet applies to. Fortunately HTML4 foresaw these requests and defined a forward-compatible syntax for media types. Here is a quote from HTML4, section 6.13:

> Future versions of HTML may introduce new values and may allow parameterized values. To facilitate the introduction of these extensions, conforming user agents must be able to parse the `media` attribute value as follows:
>
> 1. The value is a comma-separated list of entries. For example,
>
>    ```
>    media="screen, 3d-glasses, print and resolution > 90dpi"
>    ```
>
>   is mapped to:
>
>    ```
>    "screen"
>    "3d-glasses"
>    "print and resolution > 90dpi"
>    ```
>
> 2. Each entry is truncated just before the first character that isn't a US ASCII letter [a-zA-Z] (Unicode decimal 65-90, 97-122), digit [0-9] (Unicode hex 30-39), or hyphen (45). In the example, this gives:
>
>    ```
>    "screen"
>    "3d-glasses"
>    "print"
>    ```

Media queries, as described in this specification, build on the mechanism outlined in HTML4. The syntax of media queries fit into the media type syntax reserved in HTML4. The `media` attribute of HTML4 also exists in XHTML and generic XML. The same syntax can also be used inside in the '`@media`' and '`@import`' rules of CSS.

However, the parsing rules for media queries are incompatible with those of HTML4 so that they are consistent with those of media queries used in CSS.

> HTML5 [HTML5] (at the moment of writing still work in progress) references the Media Queries specification directly and thus updates the rules for HTML.

2. Media Queries

A media query consists of a media type and zero or more expressions that check for the

conditions of particular media features.

Statements regarding media queries in this section assume the syntax section is followed. Media queries that do not conform to the syntax are discussed in the error handling section. I.e. the syntax takes precedence over requirements in this section.

Here is a simple example written in HTML:

```
<link rel="stylesheet" media="screen and (color)" href="example.css" />
```

This example expresses that a certain style sheet (`example.css`) applies to devices of a certain media type (‘`screen`’) with certain feature (it must be a color screen).

Here the same media query written in an @import-rule in CSS:

```
@import url(color.css) screen and (color);
```

A media query is a logical expression that is either true or false. A media query is true if the media type of the media query matches the media type of the device where the user agent is running (as defined in the "Applies to" line), and all expressions in the media query are true.

A shorthand syntax is offered for media queries that apply to all media types; the keyword ‘`all`’ can be left out (along with the trailing ‘`and`’). I.e. if the media type is not explicitly given it is ‘`all`’.

I.e. these are identical:

```
@media all and (min-width:500px) { … }
@media (min-width:500px) { … }
```

As are these:

```
@media (orientation: portrait) { … }
@media all and (orientation: portrait) { … }
```

Several media queries can be combined in a media query list. A comma-separated list of media queries. If one or more of the media queries in the comma-separated list are true,

the whole list is true, and otherwise false. In the media queries syntax, the comma expresses a logical OR, while the 'and' keyword expresses a logical AND.

Here is an example of several media queries in a comma-separated list using the an @media-rule in CSS:

```
@media screen and (color), projection and (color) { … }
```

If the media query list is empty (i.e. the declaration is the empty string or consists solely of whitespace) it evaluates to true.

I.e. these are equivalent:

```
@media all { … }
@media { … }
```

The logical NOT can be expressed through the 'not' keyword. The presence of the keyword 'not' at the beginning of the media query negates the result. I.e., if the media query had been true without the 'not' keyword it will become false, and vice versa. User agents that only support media types (as described in HTML4) will not recognize the 'not' keyword and the associated style sheet is therefore not applied.

```
<link rel="stylesheet" media="not screen and (color)" href="example.css" />
```

The keyword 'only' can also be used to hide style sheets from older user agents. User agents must process media queries starting with 'only' as if the 'only' keyword was not present.

```
<link rel="stylesheet" media="only screen and (color)" href="example.css"
/>
```

The media queries syntax can be used with HTML, XHTML, XML [XMLSTYLE] and the @import and @media rules of CSS.

Here is the same example written in HTML, XHTML, XML, @import and @media:

```
<link rel="stylesheet" media="screen and (color), projection and (color)"
rel="stylesheet" href="example.css">

<link rel="stylesheet" media="screen and (color), projection and (color)"
rel="stylesheet" href="example.css" />

<?xml-stylesheet media="screen and (color), projection and (color)"
rel="stylesheet" href="example.css" ?>

@import url(example.css) screen and (color), projection and (color);

@media screen and (color), projection and (color) { … }
```

The [XMLSTYLE] specification has not yet been updated to use media queries in the `media` pseudo-attribute.

If a media feature does not apply to the device where the UA is running, expressions involving the media feature will be false.

The media feature '`device-aspect-ratio`' only applies to visual devices. On an aural device, expressions involving '`device-aspect-ratio`' will therefore always be false:

```
<link rel="stylesheet" media="aural and (device-aspect-ratio: 16/9)"
href="example.css" />
```

Expressions will always be false if the unit of measurement does not apply to the device.

The '`px`' unit does not apply to '`speech`' devices so the following media query is always false:

```
<link rel="stylesheet" media="speech and (min-device-width: 800px)"
href="example.css" />
```

Note that the media queries in this example would have been true if the keyword '`not`' had been added to the beginning of the media query.

To avoid circular dependencies, it is never necessary to apply the style sheet in order to evaluate expressions. For example, the aspect ratio of a printed document may be influenced by a style sheet, but expressions involving '`device-aspect-ratio`' will be based on the default aspect ratio of the user agent.

> User agents are expected, but not required, to re-evaluate and re-layout the page in response to changes in the user environment, for example if the device is tilted from landscape to portrait mode.

3. Syntax

The media query syntax is described in terms of the CSS2 grammar. As such, rules not defined here are defined in CSS2. The `media_query_list` production defined below replaces the `media_list` production from CSS2. [CSS21]

```
media_query_list
 : S* [media_query [ ',' S* media_query ]* ]?
 ;
media_query
 : [ONLY | NOT]? S* media_type S* [ AND S* expression ]*
 | expression [ AND S* expression ]*
 ;
media_type
 : IDENT
 ;
expression
 : '(' S* media_feature S* [ ':' S* expr ]? ')' S*
 ;
media_feature
 : IDENT
 ;
```

COMMENT tokens, as defined by CSS2, do not occur in the grammar (to keep it readable), but any number of these tokens may appear anywhere between other tokens. [CSS21]

The following new definitions are introduced:

```
L  l|\\0{0,4}(4c|6c)(\r\n|[ \t\r\n\f])?|\\l
Y  y|\\0{0,4}(59|79)(\r\n|[ \t\r\n\f])?|\\y
```

The following new tokens are introduced:

```
{O}{N}{L}{Y}      {return ONLY;}
{N}{O}{T}         {return NOT;}
{A}{N}{D}         {return AND;}
{num}{D}{P}{I}    {return RESOLUTION;}
{num}{D}{P}{C}{M} {return RESOLUTION;}
```

`RESOLUTION` is to be added to the CSS2 `term` production.

CSS style sheets are generally ASCII case-insensitive, and this is also the case for media queries.

In addition to conforming to the syntax, each media query needs to use media types and media features according to their respective specification in order to be considered conforming.

> Only the first media query is conforming in the example below because the "example" media type does not exist.
>
> ```
> @media all { body { background:lime } }
> @media example { body { background:red } }
> ```

3.1. Error Handling

For media queries that are not conforming user agents need to follow the rules described in this section.

- **Unknown media types.** Unknown media types evaluate to false. Effectively, they are treated identically to known media types that do not match the media type of the device.

  > The media query "`unknown`" will evaluate to false, unless `unknown` is actually a supported media type. Similarly, "`not unknown`" will evaluate to true.

  > Unknown media types are distinct from media types that do not actually match the IDENT production. Those fall under the malformed media query clause.

- **Unknown media features.** User agents are to represent a media query as "`not all`"

when one of the specified media features is not known.

```
<link rel="stylesheet" media="screen and (max-weight: 3kg) and
(color), (color)" href="example.css" />
```

In this example, the first media query will be represented as "`not all`" and evaluate to false and the second media query is evaluated as if the first had not been specified, effectively.

```
@media (min-orientation:portrait) { … }
```

Is represented as "`not all`" because the '`orientation`' feature does not accept the '`min-`' prefix.

- **Unknown media feature values.** As with unknown media features, user agents are to represent a media query as "`not all`" when one of the specified media feature values is not known.

  The media query `(color:20example)` specifies an unknown value for the '`color`' media feature and is therefore represented as "`not all`".

  This media query is represented as "`not all`" because negative lengths are not allowed for the '`width`' media feature:

  ```
  @media (min-width: -100px) { … }
  ```

- **Malformed media query.** User agents are to handle unexpected tokens encountered while parsing a media query by reading until the end of the media query, while observing the rules for matching pairs of (), [], {}, "", and '', and correctly handling escapes. Media queries with unexpected tokens are represented as "`not all`". [CSS21]

```
@media (example, all,), speech { /* only applicable to speech devices
*/ }
@media &test, screen           { /* only applicable to screen devices
*/ }
```

The following is an malformed media query because having no space between '`and`' and the expression is not allowed. (That is reserved for the functional notation syntax.)

```
@media all and(color) { … }
```

Media queries are expected to follow the error handling rules of the host language as well.

```
@media test;,all { body { background:lime } }
```

… will not apply because the semicolon terminates the `@media` rule in CSS.

4. Media features

Syntactically, media features resemble CSS properties: they have names and accept certain values. There are, however, several important differences between properties and media features:

- Properties are used in *declarations* to give information about how to present a document. Media features are used in *expressions* to describe requirements of the output device.
- Most media features accept optional '`min-`' or '`max-`' prefixes to express "greater or equal to" and "smaller or equal to" constraints. This syntax is used to avoid "<" and ">" characters which may conflict with HTML and XML. Those media features that accept prefixes will most often be used with prefixes, but can also be used alone.
- Properties always require a value to form a declaration. Media features, on the other hand, can also be used without a value. For a media feature *feature*, (*`feature`*) will evaluate to true if (*`feature`*`:`*`x`*) will evaluate to true for a value *x* other than zero or zero

followed by a unit identifier (i.e., other than `0`, `0px`, `0em`, etc.). Media features that are prefixed by min/max cannot be used without a value. When a media feature prefixed with min/max is used without a value it makes the media query malformed.

- Properties may accept more complex values, e.g., calculations that involve several other values. Media features only accept single values: one keyword, one number, or a number with a unit identifier. (The only exceptions are the '`aspect-ratio`' and '`device-aspect-ratio`' media features.)

For example, the '`color`' media feature can form expressions without a value ('`(color)`'), or with a value ('`(min-color: 1)`').

This specification defines media features usable with visual and tactile devices. Similarly, media features can be defined for aural media types.

4.1. width

Value: <length>
Applies to: visual and tactile media types
Accepts min/max prefixes: yes

The '`width`' media feature describes the width of the targeted display area of the output device. For continuous media, this is the width of the viewport (as described by CSS2, section 9.1.1 [CSS21]) including the size of a rendered scroll bar (if any). For paged media, this is the width of the page box (as described by CSS2, section 13.2 [CSS21]).

A specified <length> cannot be negative.

For example, this media query expresses that the style sheet is usable on printed output wider than 25cm:

```
<link rel="stylesheet" media="print and (min-width: 25cm)" href="http://…"
/>
```

This media query expresses that the style sheet is usable on devices with viewport (the part of the screen/paper where the document is rendered) widths between 400 and 700 pixels:

```
@media screen and (min-width: 400px) and (max-width: 700px) { … }
```

This media query expresses that style sheet is usable on screen and handheld devices if the width of the viewport is greater than 20em.

```
@media handheld and (min-width: 20em),
  screen and (min-width: 20em) { … }
```

The '`em`' value is relative to the initial value of 'font-size'.

4.2. height

Value: <length>
Applies to: visual and tactile media types
Accepts min/max prefixes: yes

The '`height`' media feature describes the height of the targeted display area of the output device. For continuous media, this is the height of the viewport including the size of a rendered scroll bar (if any). For paged media, this is the height of the page box.

A specified <length> cannot be negative.

4.3. device-width

Value: <length>
Applies to: visual and tactile media types
Accepts min/max prefixes: yes

The '`device-width`' media feature describes the width of the rendering surface of the output device. For continuous media, this is the width of the screen. For paged media, this is the width of the page sheet size.

A specified <length> cannot be negative.

```
@media screen and (device-width: 800px) { … }
```

In the example above, the style sheet will apply only to screens that currently displays exactly 800 horizontal pixels. The ‘px’ unit is of the logical kind, as described in the Units section.

4.4. device-height

Value: <length>
Applies to: visual and tactile media types
Accepts min/max prefixes: yes

The ‘device-height’ media feature describes the height of the rendering surface of the output device. For continuous media, this is the height of the screen. For paged media, this is the height of the page sheet size.

A specified <length> cannot be negative.

```
<link rel="stylesheet" media="screen and (device-height: 600px)" />
```

In the example above, the style sheet will apply only to screens that have exactly 600 vertical pixels. Note that the definition of the ‘px’ unit is the same as in other parts of CSS.

4.5. orientation

Value: portrait | landscape
Applies to: bitmap media types
Accepts min/max prefixes: no

The ‘orientation’ media feature is ‘portrait’ when the value of the ‘height’ media feature is greater than or equal to the value of the ‘width’ media feature. Otherwise ‘orientation’ is ‘landscape’.

```
@media all and (orientation:portrait) { … }
@media all and (orientation:landscape) { … }
```

4.6. aspect-ratio

Value: <ratio>
Applies to: bitmap media types
Accepts min/max prefixes: yes

The '`aspect-ratio`' media feature is defined as the ratio of the value of the '`width`' media feature to the value of the '`height`' media feature.

4.7. device-aspect-ratio

Value: <ratio>
Applies to: bitmap media types
Accepts min/max prefixes: yes

The '`device-aspect-ratio`' media feature is defined as the ratio of the value of the '`device-width`' media feature to the value of the '`device-height`' media feature.

For example, if a screen device with square pixels has 1280 horizontal pixels and 720 vertical pixels (commonly referred to as "16:9"), the following Media Queries will all match the device:

```
@media screen and (device-aspect-ratio: 16/9) { … }
@media screen and (device-aspect-ratio: 32/18) { … }
@media screen and (device-aspect-ratio: 1280/720) { … }
@media screen and (device-aspect-ratio: 2560/1440) { … }
```

4.8. color

Value: <integer>
Applies to: visual media types
Accept min/max prefixes: yes

The ‘color’ media feature describes the number of bits per color component of the output device. If the device is not a color device, the value is zero.

A specified <integer> cannot be negative.

For example, these two media queries express that a style sheet applies to all color devices:

```
@media all and (color) { … }
@media all and (min-color: 1) { … }
```

This media query expresses that a style sheet applies to color devices with 2 or more bits per color component:

```
@media all and (min-color: 2) { … }
```

If different color components are represented by different number of bits, the smallest number is used.

For instance, if an 8-bit color system represents the red component with 3 bits, the green component with 3 bits and the blue component with 2 bits, the ‘color’ media feature will have a value of 2.

In a device with indexed colors, the minimum number of bits per color component in the lookup table is used.

The described functionality is only able to describe color capabilities at a superficial level. If further functionality is required, RFC2531 [RFC2531] provides more specific media features which may be supported at a later stage.

4.9. color-index

Value: <integer>
Applies to: visual media types
Accepts min/max prefixes: yes

The ‘color-index’ media feature describes the number of entries in the color lookup table of the output device. If the device does not use a color lookup table, the value is zero.

A specified <integer> cannot be negative.

For example, here are two ways to express that a style sheet applies to all color index devices:

```
@media all and (color-index) { … }
@media all and (min-color-index: 1) { … }
```

This media query expresses that a style sheet applies to a color index device with 256 or more entries:

```
<?xml-stylesheet media="all and (min-color-index: 256)"
  href="http://www.example.com/…" ?>
```

4.10. monochrome

Value: <integer>
Applies to: visual media types
Accepts min/max prefixes: yes

The ‘monochrome’ media feature describes the number of bits per pixel in a monochrome frame buffer. If the device is not a monochrome device, the output device value will be 0.

A specified <integer> cannot be negative.

For example, here are two ways to express that a style sheet applies to all monochrome devices:

```
@media all and (monochrome) { … }
@media all and (min-monochrome: 1) { … }
```

Express that a style sheet applies to monochrome devices with more than 2 bits per pixels:

```
@media all and (min-monochrome: 2) { … }
```

Express that there is one style sheet for color pages and another for monochrome:

```
<link rel="stylesheet" media="print and (color)" href="http://…" />
<link rel="stylesheet" media="print and (monochrome)" href="http://…" />
```

4.11. resolution

Value: <resolution>
Applies to: bitmap media types
Accepts min/max prefixes: yes

The '`resolution`' media feature describes the resolution of the output device, i.e. the density of the pixels. When querying devices with non-square pixels, in '`min-resolution`' queries the least-dense dimension must be compared to the specified value and in '`max-resolution`' queries the most-dense dimensions must be compared instead. A '`resolution`' (without a "min-" or "max-" prefix) query never matches a device with non-square pixels.

For printers, this corresponds to the screening resolution (the resolution for printing dots of arbitrary color).

For example, this media query expresses that a style sheet is usable on devices with resolution greater than 300 dots per inch:

```
@media print and (min-resolution: 300dpi) { … }
```

This media query expresses that a style sheet is usable on devices with resolution greater than 118 dots per centimeter:

```
@media print and (min-resolution: 118dpcm) { … }
```

4.12. scan

Value: progressive | interlace
Applies to: "tv" media types
Accepts min/max prefixes: no

The '`scan`' media feature describes the scanning process of "tv" output devices.

> For example, this media query expresses that a style sheet is usable on tv devices with progressive scanning:
>
> ```
> @media tv and (scan: progressive) { … }
> ```

4.13. grid

Value: <integer>
Applies to: visual and tactile media types
Accepts min/max prefixes: no

The '`grid`' media feature is used to query whether the output device is grid or bitmap. If the output device is grid-based (e.g., a "tty" terminal, or a phone display with only one fixed font), the value will be 1. Otherwise, the value will be 0.

Only 0 and 1 are valid values. (This includes -0.) Thus everything else creates a malformed media query.

> Here are two examples:
>
> ```
> @media handheld and (grid) and (max-width: 15em) { … }
> @media handheld and (grid) and (device-max-height: 7em) { … }
> ```

5. Values

This specification also introduces two new values.

The <ratio> value is a positive (not zero or negative) <integer> followed by optional whitespace, followed by a solidus ('`/`'), followed by optional whitespace, followed by a

positive <integer>.

The <resolution> value is a positive <number> immediately followed by a unit identifier (‘dpi’ or ‘dpcm’).

Whitespace, <integer>, <number> and other values used by this specification are the same as in other parts of CSS, normatively defined by CSS 2.1. [CSS21]

6. Units

The units used in media queries are the same as in other parts of CSS. For example, the pixel unit represents CSS pixels and not physical pixels.

Relative units in media queries are based on the initial value, which means that units are never based on results of declarations. For example, in HTML, the ‘em’ unit is relative to the initial value of ‘font-size’.

6.1. Resolution

The ‘dpi’ and ‘dpcm’ units describe the resolution of an output device, i.e., the density of device pixels. Resolution unit identifiers are:

dpi
: dots per CSS ‘inch’

dpcm
: dots per CSS ‘centimeter’

In this specification, these units are only used in the ‘resolution’ media feature.

7. Changes

7.1. Changes Since the 27 July 2010 Candidate Recommendation

The following changes were made to this specification since the 27 July 2010 Candidate Recommendation:

- Section 4.11: Clarified the meaning of resolution in the case of printers, for which the meaning of dots was ambiguous.

> For printers, this corresponds to the screening resolution (the resolution for printing dots of arbitrary color).

- Section 6.1: Made it explicit that the '`inch`' and '`cm`' mentioned are the CSS units, not the physical ones.

> **dpi**
> dots per CSS '`inch`'~~inch~~
>
> **dpcm**
> dots per CSS '`centimeter`'~~cm~~

- Section 4.1: Adjust mistaken non normative wording to match correct normative wording from Section 6.

> The '`em`' value is relative to the ~~font size of the root element~~initial value of 'font-size'..

- Section 6: Clarify that units are never based on the results of declarations.

> Relative units in media queries are based on the initial value, which means that units are never based on results of declarations. For example, in HTML, the '`em`' unit is relative to the initial value of '`font-size`'.

Acknowledgments

This specification is the product of the W3C Working Group on Cascading Style Sheets.

Comments from Björn Höhrmann, Christoph Päper, Chris Lilley, Simon Pieters, Rijk van Geijtenbeek, Sigurd Lerstad, Arve Bersvendsen, Susan Lesch, Philipp Hoschka, Roger Gimson, Steven Pemberton, Simon Kissane, Melinda Grant, and L. David Baron improved this specification.

References

Normative references

[CSS21]

Bert Bos; et al. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification.* 7 June 2011. W3C Recommendation. URL: http://www.w3.org/TR/2011/REC-CSS2-20110607

Other references

[HTML401]

Dave Raggett; Arnaud Le Hors; Ian Jacobs. *HTML 4.01 Specification.* 24 December 1999. W3C Recommendation. URL: http://www.w3.org/TR/1999/REC-html401-19991224

[HTML5]

Ian Hickson. *HTML5.* 25 May 2011. W3C Working Draft. (Work in progress.) URL: http://www.w3.org/TR/2011/WD-html5-20110525/

[RFC2531]

G. Klyne; L. McIntyre. *Content Feature Schema for Internet Fax.* March 1999. Internet RFC 2531. URL: http://www.ietf.org/rfc/rfc2531.txt

[XMLSTYLE]

James Clark. *Associating Style Sheets with XML documents.* 29 June 1999. W3C Recommendation. URL: http://www.w3.org/1999/06/REC-xml-stylesheet-19990629