CSS Conditional Rules Module Level 5

W3C First Public Working Draft, 21 December 2021

▶ **More details about this document**

Copyright © 2021 W3C® (MIT, ERCIM, Keio, Beihang). W3C liability, trademark and permissive document license rules apply.

Abstract

This module contains the features of CSS for conditional processing of parts of style sheets, based on capabilities of the processor or the environment the style sheet is being applied in. It includes and extends the functionality of CSS Conditional 4 [css-conditional-4], adding the generalized conditional rule '@when' and the chained conditional rule '@else', as well as introducing font processing queries to the supports query syntax used in '@supports' rules.

CSS is a language for describing the rendering of structured documents (such as HTML and XML) on screen, on paper, etc.

Status of this document

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

This document was published by the CSS Working Group as a **First Public Working Draft** using the Recommendation track. Publication as a First Public Working Draft does not imply endorsement by W3C and its Members.

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.

Please send feedback by filing issues in GitHub (preferred), including the spec code "css-conditional" in the title, like this: "[css-conditional] ...*summary of comment*...". All issues and comments are

archived. Alternately, feedback can be sent to the (archived) public mailing list www-style@w3.org.

This document is governed by the 2 November 2021 W3C Process Document.

This document was produced by a group operating under the 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.

Table of Contents

§ 1. Introduction

> ISSUE 1 This is currently an early draft of the things that are *new* in level 5. The features in Level 3 and Level 4 are still defined in [css-conditional-3] and [css-conditional-4] and have not yet been copied here.

CSS Conditional Level 5 extends the '@supports' rule and supports query syntax to allow testing for supported font technologies.

It also adds an '@when' rule, which generalizes the concept of a conditional rule. Anything that can be expressed in an existing conditional rule can be expressed in '@when' by wrappng it in an appropriate function to declare what kind of condition it is. This allow authors to easily combine multiple types of queries, such as media queries and supports queries, in a single boolean expression. Without this, authors must rely on nesting separate conditional rules, which is harder to read and write, presupposes the conditions are to be conjoined with the "and" boolean relation (with no easy way to indicate anything else), and restricts their utility in the proposed conditional rule chains.

It also adds '@else' rules, which immediately follow other conditional rules and automatically qualify their conditions as the inverse of the immediately preceding rule's conditions, such that only the first matching rule in a conditional rule chain is applied.

§ 2. Extensions to the '@supports' rule

This level of the specification extends the <supports-feature> syntax as follows:

```
<supports-feature> = <supports-selector-fn> | <supports-font-tech-fn>
                   | <supports-font-format-fn> | <supports-decl>
<supports-font-tech-fn> = font-tech ( <font-tech> )
<font-tech> = [ features-opentype | features-aat | features-graphite
                   | color-colrv0 | color-colrv1 | color-svg | color-sbix | colc
                   | variations | palettes | incremental ]
```

```
<supports-font-format-fn> = font-format ( <font-format> )
<font-format = [ collection | embedded-opentype | opentype
                    | svg | truetype | woff | woff2 ]
```

ISSUE 2 <font-format> and <font-tech> should be imported from css-fonts-4, not defined here.

<supports-font-tech-fn>

The result is true if the UA supports the font tech provided as an argument to the function.

> Note: The allowed values for the 'font-tech()' function are the same as those for the 'tech()' function in the '@font-face' 'src' descriptor.

<supports-font-format-fn>

The result is true if the UA supports the font format provided as an argument to the function.

> Note: The allowed values for the 'font-format()' function are the same as those for the 'format()' function in the '@font-face' 'src' descriptor.

§ 2.1. Extensions to the definition of support

A CSS processor is considered to ***support a font tech*** when it is capable of utilising the specified *CSS Fonts 4* § 11.1 Font tech in layout and rendering.

A CSS processor is considered to ***support a font format*** when it is capable of utilising the specified *CSS Fonts 4* § 11.2 Font formats in layout and rendering.

§ 3. Generalized Conditional Rules: the '@when' rule

The ***'@when'*** at-rule is a conditional group rule that generalizes the individual conditional group rules such as '@media' and '@supports'. It is defined as:

```
@when <boolean-condition> {
  <stylesheet>
}
```

Where <boolean-condition> is a boolean algebra a la *Media Queries 4* § 3 Syntax, but with 'media()' and 'supports()' functions as leaves.

ISSUE 3 Define "boolean algebra, with X as leaves" in a generic way in Conditional, so all the conditional rules can reference it directly, rather than having to redefine boolean algebra on their own.

The ‘media()’ and ‘supports()’ functions are defined as:

```
media() = media( [ <mf-plain> | <mf-boolean> | <mf-range> ] )
supports() = supports( <declaration> )
```

A ‘media()’ or ‘supports()’ function is associated the boolean result that its contained condition is associated with.

§ 4. Chained Conditionals: the ‘@else’ rule

Usually, conditional group rules are independent; each one has a separate condition evaluated without direct reference to any other rule, and decides whether or not to apply its contained rules based solely on its condition.

This is fine for simple conditions, but makes it difficult to write a collection of conditionals that are meant to be mutually exclusive: authors have to very carefully craft their conditions to not activate when the other rules are meant to, and make sure the collection of conditionals don’t accidentally *all* exclude some situation which is then left unstyled.

The ***‘@else’*** rule is a conditional group rule used to form conditional rule chains, which associate multiple conditional rules and guarantee that only the first one that matches will evaluate its condition as true. It is defined as:

```
@else <boolean-condition>? {
  <stylesheet>
}
```

‘@else’ is interpreted identically to ‘@when’. If its <boolean-condition> is omitted, it’s treated as having a condition that’s always true.

A ***conditional rule chain*** is a series of consecutive conditional group rules, starting with a conditional group rule other than ‘@else’, followed by zero or more ‘@else’ rules. There cannot be anything between the successive conditional group rules other than whitespace and/or comments; any other token “breaks” the chain.

ISSUE 4 Should we require that only the last '@else' in a chain can have an omitted condition? It's not uncommon for me, when debugging code, to short-circuit an if-else chain by setting one of them to "true"; I presume that would be similarly useful in CSS? It's still pretty easy to see you've done something wrong if you omit the condition accidentally.

Within a conditional rule chain, the conditions of each conditional group rule are evaluated in order. If one of them is true, the conditions of all *following* conditional group rules in the chain evaluate to false, regardless of their stated condition.

An '@else' rule that is not part of a conditional rule chain is invalid and must be ignored.

EXAMPLE 1

For example, here's a (somewhat silly) conditional chain:

```
@when media(width >= 400px) and media(pointer: fine) and supports(display: flex
  /* A */
} @else supports(caret-color: pink) and supports(background: double-rainbow())
  /* B */
} @else {
  /* C */
}
```

Exactly one of the preceding rules will be chosen, even though the second rule doesn't exclude large widths, fine points, or flexbox support, and the last rule doesn't specify anything at all.

To achieve the same result without conditional rule chains, you'd need to write:

```
@media (width >= 400px) and (pointer: fine) {
  @supports (display: flex) {
    /* A */
  }
  @supports not (display: flex) {
    @supports (caret-color: pink) and (background: double-rainbow()) {
      /* B */
    }
    @supports not ((caret-color: pink) and (background: double-rainbow())) {
      /* C */
    }
  }
}
@media not ((width >= 400px) and (pointer: fine)) {
  @supports (caret-color: pink) and (background: double-rainbow()) {
    /* B */
  }
  @supports not ((caret-color: pink) and (background: double-rainbow())) {
    /* C */
  }
}
```

This is simultaneously hard to read, requires significant duplication of both conditions and contents, and is *very* difficult to write correctly. If the conditions got any more complicated (which

is not unusual in real-world content), the example would get *significantly* worse.

EXAMPLE 2

In this example, three different color font technologies are tested, in order of preference, plus a monochrome fallback. The most capable, COLRv1, supports both gradients and font variations; the next best choice, SVG, supports gradients while the least capable, COLRv0, supports flat color fill only.

The fallback has no test condition, so will always be chosen unless one of the earlier conditions succeeds.

```
@when font-tech(color-COLRv1) and font-tech(variations) {
  @font-face { font-family: icons; src: url(icons-gradient-var.woff2); }
}
@else font-tech(color-SVG) {
  @font-face { font-family: icons; src: url(icons-gradient.woff2); }
}
@else font-tech(color-COLRv0) {
  @font-face { font-family: icons; src: url(icons-flat.woff2); }
}
@else {
  @font-face { font-family: icons; src: url(icons-fallback.woff2); }
}
```

Notice that in this example, the variable color font is only downloaded if COLRv1 is supported and font variations are also supported.

Notice too that only one of the available options will be downloaded; this would not be the case without ‘@when’ and ‘@else’, as the next example shows.

EXAMPLE 3

In this example, although it appears that the fallback will not be used if COLRv1 is supported, in fact both fonts will be downloaded, which wastes bandwidth if it is not used.

The fallback might still be used for some characters; for example, if the color font supports only Latin, while the fallback supports Latin and Greek.

```
@font-face { font-family: icons; src: url(icons-fallback.woff2);
@supports font-tech(color-COLRv1) {
  @font-face { font-family: icons; src: url(icons-gradient-var.woff2); }
}
```

§ Security Considerations

No security issues have been raised against this document

§ Privacy Considerations

The ‘font-tech()’ and ‘font-format()’ functions may provide information about the user’s software such as its version and whether it is running with non-default settings that enable or disable certain features.

This information can also be determined through other APIs. However, the features in this specification are one of the ways this information is exposed on the Web.

This information can also, in aggregate, be used to improve the accuracy of fingerprinting of the user.

§ Acknowledgments

The ‘@when’ and ‘@else’ rules are based on a proposal by Tab Atkins.

§ Changes

§ Additions since Level 4

- Added ‘@when’ and ‘@else’.

- Extended supports queries to express font capabilities via ‘font-tech()’ and ‘font-format()’.

§ Conformance

§ Document conventions

Conformance requirements are expressed with a combination of descriptive assertions and RFC 2119 terminology. The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in the normative parts of this document are to be interpreted as described in RFC 2119. However, for readability, these words do not appear in all uppercase letters in this specification.

All of the text of this specification is normative except sections explicitly marked as non-normative, examples, and notes. [RFC2119]

Examples in this specification are introduced with the words “for example” or are set apart from the normative text with `class="example"`, like this:

> EXAMPLE 4
>
> This is an example of an informative example.

Informative notes begin with the word “Note” and are set apart from the normative text with `class="note"`, like this:

> Note, this is an informative note.

Advisements are normative sections styled to evoke special attention and are set apart from other normative text with `<strong class="advisement">`, like this:

> **UAs MUST provide an accessible alternative.**

§ Conformance classes

Conformance to this specification is defined for three conformance classes:

style sheet
: A CSS style sheet.

renderer
: A UA that interprets the semantics of a style sheet and renders documents that use them.

authoring tool
: A UA that writes a style sheet.

A style sheet is conformant to this specification if all of its statements that use syntax defined in this module are valid according to the generic CSS grammar and the individual grammars of each feature defined in this module.

A renderer is conformant to this specification if, in addition to interpreting the style sheet as defined by the appropriate specifications, it supports all the features defined by this specification by parsing them correctly and rendering the document accordingly. However, the inability of a UA to correctly render a document due to limitations of the device does not make the UA non-conformant. (For example, a UA is not required to render color on a monochrome monitor.)

An authoring tool is conformant to this specification if it writes style sheets that are syntactically correct according to the generic CSS grammar and the individual grammars of each feature in this module, and meet all other conformance requirements of style sheets as described in this module.

§ 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 component 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.

§ Implementations of Unstable and Proprietary Features

To avoid clashes with future stable CSS features, the CSSWG recommends following best practices for the implementation of unstable features and proprietary extensions to CSS.

§ 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 https://www.w3.org/Style/CSS/Test/. Questions should be directed to the public-css-testsuite@w3.org mailing list.

§ Index

§ Terms defined by this specification

§ Terms defined by reference

[css-conditional-3] defines the following terms:

- <supports-decl>
- @media
- @supports
- conditional group rule
- supports queries

[css-conditional-4] defines the following terms:

- <supports-selector-fn>

[css-fonts-4] defines the following terms:

- src

[css-fonts-5] defines the following terms:

- <font-format>

[css-syntax-3] defines the following terms:

- <stylesheet>

[css-values-4] defines the following terms:

- ?
- |

[mediaqueries-5] defines the following terms:

- <mf-boolean>
- <mf-plain>
- <mf-range>

§ References

§ Normative References

[CSS-CONDITIONAL-3]
David Baron; Elika Etemad; Chris Lilley. *CSS Conditional Rules Module Level 3*. 8 December 2020. CR. URL: https://www.w3.org/TR/css-conditional-3/

[CSS-CONDITIONAL-4]
David Baron. *CSS Conditional Rules Module Level 4*. 3 March 2020. WD. URL: https://www.w3.org/TR/css-conditional-4/

[CSS-FONTS-5]
Myles Maxfield; Chris Lilley. *CSS Fonts Module Level 5*. 29 July 2021. WD. URL: https://www.w3.org/TR/css-fonts-5/

[CSS-SYNTAX-3]
Tab Atkins Jr.; Simon Sapin. *CSS Syntax Module Level 3*. 16 July 2019. CR. URL: https://www.w3.org/TR/css-syntax-3/

[CSS-VALUES-4]
Tab Atkins Jr.; Elika Etemad. *CSS Values and Units Module Level 4*. 16 December 2021. WD. URL: https://www.w3.org/TR/css-values-4/

[MEDIAQUERIES-5]
Dean Jackson; Florian Rivoal; Tab Atkins Jr.. *Media Queries Level 5*. 31 July 2020. WD. URL: https://www.w3.org/TR/mediaqueries-5/

[RFC2119]
S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. March 1997. Best Current Practice. URL: https://datatracker.ietf.org/doc/html/rfc2119

§ Informative References

[CSS-FONTS-4]
John Daggett; Myles Maxfield; Chris Lilley. *CSS Fonts Module Level 4*. 29 July 2021. WD. URL: https://www.w3.org/TR/css-fonts-4/

§ Issues Index