CSS Conditional Rules Module Level 4

W3C Candidate Recommendation Snapshot, 17 February 2022

▶ **More details about this document**

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 3 [css-conditional-3], adding the ability to query support for particular selectors [SELECTORS-4] through the new 'selector()' notation for supports queries.

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 **Candidate Recommendation Snapshot** using the Recommendation track. Publication as a Candidate Recommendation does not imply endorsement by W3C and its Members. A Candidate Recommendation Snapshot has received wide review, is intended to gather implementation experience, and has commitments from Working Group members to royalty-free licensing for implementations. This document is intended to become a W3C Recommendation; it will remain a Candidate Recommendation at least until 17 April 2022 to gather additional feedback.

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 The features in level 3 are still defined in [css-conditional-3] and have not yet been copied here.

This level adds extensions to the '@supports' rule to allow testing for supported selectors.

§ 2. Extensions to the '@supports' rule

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

```
<supports-feature> = <supports-selector-fn> | <supports-decl>
<supports-selector-fn> = selector( <complex-selector> )
```

<supports-selector-fn>
: The result is true if the UA supports the selector provided as an argument to the function.

▼ TESTS

at-supports-selector-001.html	(live test) (source)
at-supports-selector-002.html	(live test) (source)
at-supports-selector-003.html	(live test) (source)
at-supports-selector-004.html	(live test) (source)
CSS-supports-L4.html (12 tests)	(live test) (source)

EXAMPLE 1

This example tests whether the column combinator (||) is supported in selectors, and if so uses it to style particular cells in a table.

```
@supports selector(col || td) {
  col.selected || td {
    background: tan;
  }
}
```

Any namespace prefixes used in a conditional group rule must have been declared, otherwise they are invalid [css-conditional-3]. This includes namespace prefixes inside the selector function.

▼ TESTS

at-supports-namespace-002.html (live test) (source)

INVALID EXAMPLE2

This example tries to check that attribute selectors with CSS qualified names are supported, but is invalid, because the namespace prefix has not been declared.

```
@supports selector(a[xlink|href]) {
  // do something, but fail
  }
}
```

EXAMPLE 3

This example checks that attribute selectors with CSS qualified names are supported.

```
@namespace x url(http://www.w3.org/1999/xlink);
@supports selector(a[x|href]) {
  // do something
  }
}
```

§ 2.1. Extensions to the definition of support

A CSS processor is considered to ***support a CSS selector*** if it accepts that selector (rather than discarding it as a parse error), and that selector doesn't contain unknown -webkit- pseudo-elements.

§ Security Considerations

No Security issues have been raised against this document

§ Privacy Considerations

The 'selector()' function 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 editors would like to thank all of the contributors to the previous level of this module.

§ Changes

§ Changes since the First Public Working Draft of 3 March 2020

- Added Privacy and Security sections.
- Added some examples
- Clarified that the requirement to declare namespace prefixes applies to selectors inside selector() (Issue 3220)

§ Additions since Level 3

- Added 'selector()' notation to supports queries.

§ 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.**

> ▼ TESTS
>
> Tests relating to the content of this specification may be documented in “Tests” blocks like this one. Any such block is non-normative.

§ 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.

§ CR exit criteria

For this specification to be advanced to Proposed Recommendation, there must be at least two independent, interoperable implementations of each feature. Each feature may be implemented by a different set of products, there is no requirement that all features be implemented by a single product. For the purposes of this criterion, we define the following terms:

independent
: each implementation must be developed by a different party and cannot share, reuse, or derive from code used by another qualifying implementation. Sections of code that have no bearing on the implementation of this specification are exempt from this requirement.

interoperable
: passing the respective test case(s) in the official 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 to the general public. The implementation may be a shipping product or other publicly available version (i.e., beta version, preview release, or "nightly build"). Non-

shipping product releases must have implemented the feature(s) for a period of at least one month in order to demonstrate stability.

3. is not experimental (i.e., a version specifically designed to pass the test suite and is not intended for normal usage going forward).

The specification will remain Candidate Recommendation for at least six months.

§ Index

§ Terms defined by this specification

§ Terms defined by reference

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

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

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

- css qualified name

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

- |

[SCROLL-ANIMATIONS] defines the following terms:

- selector()

[SELECTORS-4] defines the following terms:

- <complex-selector>
- column combinator
- unknown -webkit- pseudo-elements

§ References

§ Normative References

[CSS-CONDITIONAL-3]
David Baron; Elika Etemad; Chris Lilley. *CSS Conditional Rules Module Level 3*. 13 January 2022. CR. URL: https://www.w3.org/TR/css-conditional-3/

[CSS-VALUES-4]