CSS Device Adaptation Module Level 1

W3C Working Draft, 29 March 2016

This version:
http://www.w3.org/TR/2016/WD-css-device-adapt-1-20160329/
Latest version:
http://www.w3.org/TR/css-device-adapt-1/
Editor's Draft:
https://drafts.csswg.org/css-device-adapt/
Previous Versions:
http://www.w3.org/TR/2015/WD-css-device-adapt-1-20151126/
Feedback:
www-style@w3.org with subject line "[css-device-adapt] ... *message topic* ..." (archives)
Issue Tracking:
Bugzilla
Inline In Spec
Editors:
Rune Lillesveen (Opera Software)
Florian Rivoal (Vivliostyle) florian@rivoal.net
Matt Rakow (Microsoft)
Former Editors:
Ryan Betts (Adobe Systems)
Øyvind Stenhaug (Opera Software)

Abstract

This specification provides a way for an author to specify, in CSS, the size, zoom factor, and orientation of the viewport that is used as the base for the initial containing block.
CSS is a language for describing the rendering of structured documents (such as HTML and XML) on screen, on paper, in speech, etc.

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 Draft 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 (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 "css-device-adapt" in the subject, preferably like this: "[css-device-adapt] *...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.
This document is governed by the 1 September 2015 W3C Process Document.

1. Introduction

This section is not normative.
CSS 2.1 [CSS21] specifies an initial containing block for continuous media that has the dimensions of the viewport. Mobile/handheld device browsers have a viewport that is generally a lot narrower than a desktop browser window at a zoom level that gives a CSS pixel size recommended by CSS 2.1.

The narrow viewport is a problem for documents designed to look good in desktop browsers. The result is that mobile browser vendors use a fixed initial containing block size that is different from the viewport size, and close to that of a typical desktop browser window. In addition to scrolling or panning, zooming is often used to change between an overview of the document and zoom in on particular areas of the document to read and interact with.
Certain DOCTYPEs (for instance XHTML Mobile Profile) are used to recognize mobile documents which are assumed to be designed for handheld devices, hence using the viewport size as the initial containing block size.
Additionally, an HTML <META> tag has been introduced for allowing an author to specify the size of the initial containing block, and the initial zoom factor directly. It was first implemented by Apple for the Safari/iPhone browser, but has since been implemented for the Opera, Android, and Fennec browsers. These implementations are not fully interoperable and this specification is an attempt at standardizing the functionality provided by the viewport <META> tag in CSS.
This specification is written from an implementation centric point of view, making it arguably difficult to read. Significant editorial work may be needed to make it more understandable to different audiences. It also should clarify which viewport is referred to by various js APIs. See this blog post by ppk for a good discussion of these issues.
Various issues about this specification and related specifications are listed in this report.

2. Values

This specification follows the CSS property definition conventions from [CSS3SYN].
Value types are defined in [CSS3VAL].

3. The viewport

In CSS 2.1 a viewport is a feature of a user agent for continuous media and used to establish the initial containing block for continuous media. For paged media, the initial containing block is based on the page area. The page area can be set through @page rules. Hence, @viewport applies to continuous media, and @page to paged media, and they will not interact or conflict.
This specification introduces a way of overriding the size of the viewport provided by the user agent (UA). Because of this, we need to introduce the difference between the initial viewport and the actual viewport.
initial viewport
This refers to the viewport before any UA or author styles have overridden the viewport given by the window or viewing area of the UA. Note that the initial viewport size will change with the size of the window or viewing area.
actual viewport
This is the viewport you get after the cascaded viewport descriptors, and the following constraining procedure have been applied.
When the actual viewport cannot fit inside the window or viewing area, either because the actual viewport is larger than the initial viewport or the zoom factor causes only parts of the actual viewport to be visible, the UA should offer a scrolling or panning mechanism.
It is recommended that initially the upper-left corners of the actual viewport and the window or viewing area are aligned if the base direction of the document is ltr. Similarly, that the upper-right corners are aligned when the base direction is rtl. The base direction for a document is defined as the computed value of the direction property for the first <BODY> element of an HTML or XHTML document. For other document types, it is the computed direction for the root element.
"dbaron: The question is, what does this do on the desktop browser? (And what's a desktop browser)". Need to say that a "desktop" browser typically have no UA styles, as opposed to the UA stylesheet outlined for current mobile behaviour, and that no UA styles for @viewport will give "desktop" behaviour per default (actual viewport is initial viewport).

4. The @viewport rule

UA vendors implementing this specification are strongly encouraged to do so both for their mobile and desktop browsers. The @viewport mechanism is designed to be usable and useful on all browsers, not only mobile ones. However, if support is only available on mobile browsers for a significant time, there is a risk that authors would write @viewport rules that work on mobile but do the wrong if applied by a desktop browser. This would make it difficult to later add support for @viewport in desktop browsers.
An example of such misguided use would be to write @viewport { width: 320px; } instead of @viewport { width: auto; } to make a document "mobile friendly".
The @viewport at-rule consists of the @-keyword followed by a block of descriptors describing the viewport.
The descriptors inside an @viewport rule are per document and there is no inheritance involved. Hence declarations using the inherit keyword will be dropped. They work similarly to @page descriptors and follow the cascading order of CSS. Hence, descriptors in @viewport rules will override descriptors from preceding rules. The declarations allow !important which will affect cascading of descriptors accordingly.
@viewport rules apply to top level documents only.
This example sets the viewport to at least 320px, but otherwise match window width if it is wider than 320px. Note that it is enough to set the width as the height will be resolved from the width when auto.

```
@viewport {
 width: 320px auto;
}
```

4.1. Syntax

The syntax for the @viewport rule is as follows (using the notation from the Grammar appendix of CSS 2.1 [CSS21]):

```
viewport
 : VIEWPORT_SYM S* '{' S* declaration? [ ';' S* declaration? ]* '}' S*
 ;
```

with the new token:

```
@{V}{I}{E}{W}{P}{O}{R}{T}  {return VIEWPORT_SYM;}
```

where:

```
V v|\\0{0,4}(56|76)(\r\n|[ \t\r\n\f])?|\\v
W w|\\0{0,4}(57|77)(\r\n|[ \t\r\n\f])?|\\w
```

The viewport non-terminal is added to the stylesheet production along with the ruleset, media, and page non-terminals:

```
stylesheet
 : [ CHARSET_SYM STRING ';' ]?
 [S|CDO|CDC]* [ import [ CDO S* | CDC S* ]* ]*
 [ [ ruleset | media | page | viewport ] [ CDO S* | CDC S* ]* ]*
 ;
```

It is also added to the nested_statement production defined in [CSS3-CONDITIONAL] to allow @viewport rules nested inside conditional group rules such as @media or @supports:

```
nested_statement
 : ruleset | media | page | font_face_rule | keyframes_rule |
 supports_rule | viewport
 ;
```

5. Viewport descriptors

This section presents the descriptors that are allowed inside an @viewport rule. Other descriptors than those listed here will be dropped.

Relative length values are resolved against initial values. For instance 'em's are resolved against the initial value of the font-size property. Viewport lengths (vw, vh, vmin, vmax) are relative to the initial viewport.

5.1. The min-width and max-width descriptors

min- and max- functionality can be achieved with media queries, should these be removed?

Name:	min-width
For:	@viewport
Value:	<viewport-length>
Initial:	auto
Percentages:	Refer to the width of the initial viewport
Media:	visual, continuous
Computed value:	auto, an absolute length, or a percentage as specified

Name:	max-width
For:	@viewport
Value:	<viewport-length>
Initial:	auto
Percentages:	Refer to the width of the initial viewport
Media:	visual, continuous
Computed value:	auto, an absolute length, or a percentage as specified

Specifies the minimum and maximum width of the viewport that is used to set the size of the initial containing block

where

<viewport-length> = auto | <length> | <percentage>

and the values have the following meanings:

auto
The used value is calculated from the other descriptors' values according to the constraining procedure.
The user-agent stylesheets recommended in the informative section don’t adequately represent current implementation behaviors. Should there be a more explicit mechanism for switching between UA default behavior and requesting the CSS pixel?
<length>
A non-negative absolute or relative length.
<percentage>
A percentage value relative to the width or height of the initial viewport at zoom factor 1.0, for horizontal and vertical lengths respectively. Must be non-negative.
The min-width and max-width descriptors are inputs to the constraining procedure. The width will initially be set as close as possible to the "initial viewport" width within the min/max constraints.

5.2. The width shorthand descriptor

Name:	width
For:	@viewport
Value:	<viewport-length>{1,2}
Initial:	See individual descriptors
Percentages:	See individual descriptors
Media:	visual, continuous
Computed value:	See individual descriptors

This is a shorthand descriptor for setting both min-width and max-width. One <viewport-length> value will set both min-width and max-width to that value. Two <viewport-length> values will set min-width to the first and max-width to the second.

5.3. The min-height and max-height descriptors

Name:	min-height
For:	@viewport
Value:	<viewport-length>
Initial:	auto
Percentages:	Refer to the height of the initial viewport
Media:	visual, continuous
Computed value:	auto, an absolute length, or a percentage as specified

Name:	max-height
For:	@viewport
Value:	<viewport-length>
Initial:	auto
Percentages:	Refer to the height of the initial viewport
Media:	visual, continuous
Computed value:	auto, an absolute length, or a percentage as specified

Specifies the minimum and maximum height of the viewport that is used to set the size of the initial containing block. The min-height and max-height descriptors are inputs to the constraining procedure. The height will initially be set as close as possible to the "initial viewport" height within the min/max constraints.

5.4. The height shorthand descriptor

Name:	height
For:	@viewport
Value:	<viewport-length>{1,2}

Initial:	See individual descriptors
Percentages:	See individual descriptors
Media:	visual, continuous
Computed value:	See individual descriptors

This is a shorthand descriptor for setting both min-height and max-height. One <viewport-length> value will set both min-height and max-height to that value. Two <viewport-length> values will set min-height to the first and max-height to the second.

5.5. The zoom descriptor

Name:	zoom
For:	@viewport
Value:	auto | <number> | <percentage>
Initial:	auto
Percentages:	The zoom factor itself
Media:	visual, continuous
Computed value:	auto, or a non-negative number or percentage as specified

Specifies the initial zoom factor for the window or viewing area. This is a magnifying glass type of zoom. Interactively changing the zoom factor from the initial zoom factor does not affect the size of the initial or the actual viewport.
Values have the following meanings:
auto
The zoom factor is UA-dependent. The UA may use the size of the area of the canvas on which the document is rendered to find that initial zoom factor. See this section for a proposed way of handling auto values for zoom.
<number>
A non-negative number used as a zoom factor. A factor of 1.0 means that no zooming is done. Values larger than 1.0 gives a zoomed-in effect and values smaller than 1.0 a zoomed-out effect.
<percentage>
A non-negative percentage value used as a zoom factor. A factor of 100% means that no zooming is done. Values larger than 100% gives a zoomed-in effect and values smaller than 100% a zoomed-out effect.

5.6. The min-zoom descriptor

Name:	min-zoom
For:	@viewport
Value:	auto | <number> | <percentage>
Initial:	auto
Percentages:	The zoom factor itself
Media:	visual, continuous
Computed value:	auto, or a non-negative number or percentage as specified

Specifies the smallest allowed zoom factor. It is used as input to the constraining procedure to constrain non-auto zoom values, but also to limit the allowed zoom factor that can be set through user interaction. The UA should also use this value as a constraint when choosing an actual zoom factor when the used value of zoom is auto.
Values have the following meanings:
auto
The lower limit on zoom factor is UA dependant. There will be no minimum value constraint on the zoom descriptor used in the constraining procedure
<number>
A non-negative number limiting the minimum value of the zoom factor.
<percentage>
A non-negative percentage limiting the minimum value of the zoom factor.

5.7. The max-zoom descriptor

Name:	max-zoom

For:	@viewport
Value:	auto | <number> | <percentage>
Initial:	auto
Percentages:	The zoom factor itself
Media:	visual, continuous
Computed value:	auto, or a non-negative number or percentage as specified

Specifies the largest allowed zoom factor. It is used as input to the constraining procedure to constrain non-auto zoom values, but also to limit the allowed zoom factor that can be set through user interaction. The UA should also use this value as a constraint when choosing an actual zoom factor when the used value of zoom is auto.
Values have the following meanings:
auto
The upper limit on zoom factor is UA dependant. There will be no maximum value constraint on the zoom descriptor used in the constraining procedure
<number>
A non-negative number limiting the maximum value of the zoom factor.
<percentage>
A non-negative percentage limiting the maximum value of the zoom factor.

5.8. The user-zoom descriptor

Name:	user-zoom
For:	@viewport
Value:	zoom | fixed
Initial:	zoom
Percentages:	N/A
Media:	visual, continuous
Computed value:	as specified

Specifies if the zoom factor can be changed by user interaction or not.
Values have the following meanings:
zoom
The user can interactively change the zoom factor.
fixed
The user cannot interactively change the zoom factor.

5.9. The orientation descriptor

Name:	orientation
For:	@viewport
Value:	auto | portrait | landscape
Initial:	auto
Percentages:	N/A
Media:	visual, continuous
Computed value:	as specified

This descriptor is used to request that a document is displayed in portrait or landscape mode. For a UA/device where the orientation is changed upon tilting the device, an author can use this descriptor to inhibit the orientation change. The descriptor should be respected for standalone web applications, and when the document is displayed in fullscreen. It is recommended that it is ignored for normal web navigation to avoid confusing the user.
Values have the following meanings:
auto
The UA automatically chooses the orientation based on the device's normal mode of operation. The UA may choose to change the orientation of the presentation when the device is tilted.
portrait
The document should be locked to portrait presentation.
landscape

The document should be locked to landscape presentation.

6. Constraining viewport descriptor values

6.1. Definitions

For the procedure below:
Descriptors refer to the values resolved/constrained to at that point in the procedure. They are initially resolved to their computed values.
width and height refer to the resolved viewport size and not the shorthand descriptors. They are both initially auto.
MIN/MAX computations where one of the arguments is auto resolve to the other argument. For instance, MIN(0.25, auto) = 0.25, and MAX(5, auto) = 5.
initial-width is the width of the initial viewport in pixels at zoom factor 1.0.
initial-height is the height of the initial viewport in pixels at zoom factor 1.0.

6.2. The procedure

The used values are resolved from the computed values going through the steps below.
User agents are expected, but not required, to re-run this procedure and re-layout the document, if necessary, in response to changes in the user environment, for example if the device is tilted from landscape to portrait mode or the window that forms the "initial viewport" is resized.
However, Media Queries and Device Adaption are tethered specifications. As a result, UAs that also support Media Queries must re-run this procedure and re-layout the document in all cases where changes in the user environment would cause them to re-evaluate Media Queries.

Resolve min-zoom and max-zoom values

1. If min-zoom is not auto and max-zoom is not auto, set max-zoom = MAX(min-zoom, max-zoom)

Constrain zoom value to the [min-zoom, max-zoom] range

1. If zoom is not auto, set zoom = MAX(min-zoom, MIN(max-zoom, zoom))

Resolve non-auto lengths to pixel lengths

1. Resolve absolute lengths and percentages to pixel values for the min-width, max-width, min-height, and max-height descriptors.

Resolve initial width and height from min/max descriptors

1. If min-width or max-width is not auto, set width = MAX(min-width, MIN(max-width, initial-width))
2. If min-height or max-height is not auto, set height = MAX(min-height, MIN(max-height, initial-height))

Resolve width value

1. If width and height are both auto, set width = initial-width
2. Otherwise, if width is auto, set width = height * (initial-width / initial-height), or width = initial-width if initial-height is 0.

Resolve height value

1. If height is auto, set height = width * (initial-height / initial-width), or height = initial-height if initial-width is 0.

7. Media Queries

For several media features, the size of the initial containing block and the orientation of the device affects the result of a media query evaluation, which means that the effect of @viewport rules on media queries needs extra attention.
From the Media Queries specification [MEDIAQ]:
“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.”
The UA must however cascade @viewport rules separately with the initial viewport size used for evaluating media feature expressions and other values that depend on the viewport size to avoid circular dependencies, but use the actual viewport size when cascading all other rules.
Procedure for applying CSS rules:

1. Cascade all @viewport rules using the initial viewport size for values and evaluations which rely on viewport size
2. Compute the actual viewport from the cascaded viewport descriptors
3. Cascade all other rules using the actual viewport size

The rationale for using the viewport descriptors obtained from applying the @viewport rules for evaluating media queries for style rules, is that media queries should match the actual viewport that the document will be layed out in and not the initial or the one specified in the UA stylesheet. Consider the example below given that the UA stylesheet has a viewport width of 980px, but an initial viewport width of 320px. The author has made separate styles to make the document look good for initial containing block widths above or below 400px. The actual viewport used will be 320px wide, and in order to match the styles with the "actual viewport" width, the viewport resulting from applying the @viewport rules should be used to evaluate the media queries.

Given an initial viewport width of 320px and a UA stylesheet viewport width of 980px, the first media query will not match, but the second will.

```
@viewport {
 width: auto;
}

@media screen and (min-width: 400px) {
 div { color: red; }
}

@media screen and (max-width: 400px) {
 div { color: green; }
}
```

Another example:

The media query below should match because the @viewport rule is applied before the media query is evaluated.

```
@media screen and (width: 397px) {
 div { color: green; }
}

@viewport {
 width: 397px;
}
```

Below is an example where an @viewport rule relies on a media query affected by the viewport descriptors.

The green color should be applied to a div because the initial viewport width is used to evaluate the media query for the second @viewport rule, but the actual viewport is used for evaluating the media query when applying style rules.

```
@viewport {
 width: 397px;
}

@media screen and (width: 397px) {
 @viewport {
 width: 500px;
 }
}

@media screen and (width: 397px) {
 div { color: green; }
}
```

It is recommended that authors do not write @viewport rules that rely on media queries whose evaluation is affected by viewport descriptors. Is is also recommended that the @viewport rule(s) is placed as early in the document as possible to avoid unnecessary re-evaluation of media queries or reflows.

8. CSSOM

The @viewport rule is exposed to the CSSOM through a new CSSRule interface.

8.1. Interface CSSRule

The following rule type is added to the CSSRule interface. It provides identification for the new viewport rule.

```
partial interface CSSRule {
 const unsigned short VIEWPORT_RULE = 15;
};
```

8.2. Interface CSSViewportRule

The CSSViewportRule interface represents the style rule for an @viewport rule.

```
interface CSSViewportRule : CSSRule {
 readonly attribute CSSStyleDeclaration style;
};
```

style of type CSSStyleDeclaration, readonly

This attribute represents the viewport descriptors associated with this @viewport rule.

9. Viewport <META> element

This section is not normative.

This section describes a mapping from the content attribute of the viewport <META> element, first implemented by Apple in the iPhone Safari browser, to the descriptors of the @viewport rule described in this specification.

In order to match the Safari implementation, the following parsing algorithm and translation rules rely on the UA stylesheet below. See the section on UA stylesheets for an elaborate description.

```
@viewport {
 width: extend-to-zoom 980px;
 min-zoom: 0.25;
 max-zoom: 5;
}
```

Note that these values might not fit well with all UAs. For instance, with a min-zoom of 0.25 you will be able to fit the whole width of the document inside the window for widths up to 1280px on a 320px wide device like the original iPhone, but only 960px if you have a 240px display (all widths being given in CSS pixel units).

9.1. Properties

The recognized properties in the viewport <META> element are:

- width
- height
- initial-scale
- minimum-scale
- maximum-scale
- user-scalable

9.2. Parsing algorithm

Below is an algorithm for parsing the content attribute of the <META> tag produced from testing Safari on the iPhone. The testing was done on an iPod touch running iPhone OS 4. The UA string of the browser: "Mozilla/5.0 (iPod; U; CPU iPhone OS 4_0 like Mac OS X; en-us) AppleWebKit/532.9 (KHTML, like Gecko) Version/4.0.5 Mobile/8A293 Safari/6531.22.7". The pseudo code notation used is based on the notation used in [Algorithms].

The whitespace class contains the following characters (ascii):

- Horizontal tab (0x09)
- Line feed (0x0a)
- Carriage return (0x0d)
- Space (0x20)

The recognized separator between property/value pairs is comma for the Safari implementation. Some implementations have supported both commas and semicolons. Because of that, existing content use semicolons instead of commas. Authors should be using comma in order to ensure content works as expected in all UAs, but implementors may add support for both to ensure interoperability for existing content.

The separator class contains the following characters (ascii), with comma as the preferred separator and semicolon as optional:

- Comma (0x2c)
- Semicolon (0x3b)

```
Parse-Content(S)
i ← 1
while i ≤ length[S]
 do while i ≤ length[S] and S[i] in [whitespace, separator, '=']
 do i ← i + 1
 if i ≤ length[S]
 then i ← Parse-Property(S, i)

Parse-Property(S, i)
start ← i
```

while i ≤ length[S] and S[i] not in [whitespace, separator, '=']
 do i ← i + 1
if i > length[S] or S[i] in [separator]
 then return i
property-name ← S[start .. (i - 1)]
while i ≤ length[S] and S[i] not in [separator, '=']
 do i ← i + 1
if i > length[S] or S[i] in [separator]
 then return i
while i ≤ length[S] and S[i] in [whitespace, '=']
 do i ← i + 1
if i > length[S] or S[i] in [separator]
 then return i
start ← i
while i ≤ length[S] and S[i] not in [whitespace, separator, '=']
 do i ← i + 1
property-value ← S[start .. (i - 1)]
Set-Property(property-name, property-value)
return i

Set-Property matches the listed property names case-insensitively. The property-value strings are interpreted as follows:

1. If a prefix of property-value can be converted to a number using strtod, the value will be that number. The remainder of the string is ignored.
2. If the value can not be converted to a number as described above, the whole property-value string will be matched with the following strings case-insensitively: yes, no, device-width, device-height
3. If the string did not match any of the known strings, the value is unknown.

9.3. extend-to-zoom

In order to be able to implement the functionality from <META> viewport where the viewport width or height is extended to fill the viewing area at a given zoom level, we introduce a UA internal value to the list of <viewport-length> values called extend-to-zoom. It will be used in width and height declarations in the translation outlined in the section below.

This new value is necessary in order to implement the mapping for two reasons. First, whether resolving the width/height needs to extend the pixel length to the visible width/height for a given zoom factor depends on the current initial width/height. <meta name="viewport" content="width=400, initial-scale=1"> yields a width of 400px for an initial-width of 320px, and 640px for an initial width of 640px. This can not be expressed as normative min/max descriptors that would constrain correctly when the initial width changes like for an orientation change.

Secondly, the extended width/height also relies on cascading viewport properties from different sources, including min-zoom and max-zoom from the UA stylesheet. For instance, if the UA stylesheet has max-zoom: 5, and the initial width is 320px, <meta name="viewport" content="width=10"> will resolve to 64px.

Resolving 'extend-to-zoom'

The 'extend-to-zoom' value is resolved to pixel or auto lengths as part of step 3 of the constraining procedure. Since this is a *non-normative* descriptor value, the resolution is described here. Note that max-descriptors need to be resolved to pixel lengths *before* min-descriptors when 'extend-to-zoom' is a valid value.

Let extend-zoom = MIN(zoom, max-zoom)
For non-auto extend-zoom, let:
extend-width = initial-width / extend-zoom
extend-height = initial-height / extend-zoom
Then, resolve for extend-to-zoom as follows:

- If extend-zoom is auto:
- If max-width is 'extend-to-zoom', set max-width = auto
- If max-height is 'extend-to-zoom', set max-height = auto
- If min-width is 'extend-to-zoom', set min-width = max-width
- If min-height is 'extend-to-zoom', set min-height = max-height
-

If extend-zoom is non-auto:
If max-width is 'extend-to-zoom', set max-width = extend-width
If max-height is 'extend-to-zoom', set max-height = extend-height
If min-width is 'extend-to-zoom', set min-width = MAX(extend-width, max-width)
If min-height is 'extend-to-zoom', set min-height = MAX(extend-height, max-height)

9.4. Translation into @viewport descriptors

The Viewport <META> element is placed in the cascade as if it was a <STYLE> element, in the exact same place in the dom, that only contains a single @viewport rule.
Each of the property/value pair from the parsing in the previous section are translated, and added to that single at-rule as follows:

Unknown properties

Unknown properties are dropped.

The width and height properties

The width and height viewport <META> properties are translated into width and height descriptors, setting the min-width/min-height value to extend-to-zoom and the max-width/max-height value to the length from the viewport <META> property as follows:

1. Non-negative number values are translated to pixel lengths, clamped to the range: [1px, 10000px]
2. Negative number values are dropped
3. device-width and device-height translate to 100vw and 100vh respectively
4. Other keywords and unknown values translate to 1px

Some existing UA implementations use device dimensions in CSS pixels, and some use the window dimensions (CSS pixels) for device-width / device-height. Above, we translate to 100vw / 100vh which are the window dimensions. The rationale is that the device dimensions would not be what the author intended for UAs where the window is resizable or does not fill the screen of the device.
This <META> element:
<meta name="viewport" content="width=500, height=600">
translates into:
@viewport {
 width: extend-to-zoom 500px;
 height: extend-to-zoom 600px;
}
For a viewport <META> element that translates into an @viewport rule with a non-auto zoom declaration and no width declaration:

- If it adds a 'height' descriptor, add:
- width: auto;
-
- Otherwise, add:
- width: extend-to-zoom;
-

to the @viewport rule.
This <META> element:
<meta name="viewport" content="initial-scale=1.0">
translates into:
@viewport {
 zoom: 1.0;
 width: extend-to-zoom;
}
This <META> element:
<meta name="viewport" content="initial-scale=2.0,
height=device-width">
translates into:
@viewport {
 zoom: 2.0;
 width: auto;
 height: extend-to-zoom 100%;
}

The initial-scale, minimum-scale, and maximum-scale properties

The properties are translated into 'zoom', 'min-zoom', and 'max-zoom' respectively with the following translations of values.

1. Non-negative number values are translated to <number> values, clamped to the range [0.1, 10]
2. Negative number values are dropped
3. yes is translated to 1
4. device-width and device-height are translated to 10

5. no and unknown values are translated to 0.1

For a viewport <META> element that translates into an @viewport rule with no max-zoom declaration and a non-auto min-zoom value that is larger than the max-zoom value of the UA stylesheet, the min-zoom declaration value is clamped to the UA stylesheet max-zoom value.

The user-scalable property

The user-scalable property is translated into user-zoom with the following value translations.

1. yes and no are translated into zoom and fixed respectively.
2. Numbers ≥ 1, numbers ≤ -1, device-width and device-height are mapped to zoom
3. Numbers in the range <-1, 1>, and unknown values, are mapped to fixed

This <META> element:

<meta name="viewport" content="width=480, initial-scale=2.0, user-scalable=1">

will translate into this @viewport block:

```
@viewport {
 width: 480px;
 zoom: 2.0;
 user-zoom: zoom;
}
```

10. Handling auto for zoom

This section is not normative.

This section presents one way of picking an actual value for the zoom descriptor when the used value is auto.

Given an initial viewport with size (initial-width, initial-height), and a finite region within the canvas where the formatting structure is rendered (rendered-width, rendered-height). That region is at least as large as the actual viewport.

Then, if the used value of zoom is auto, let the actual zoom factor be:

zoom = MAX(initial-width / rendered-width, initial-height / rendered-height)

The actual zoom factor should also be further limited by the [min-zoom, max-zoom] range.

11. UA stylesheets

This section is informative

Traditional user agents, used mostly on desktop and laptop computers, can easily be resized to fit most websites inside the initial viewport without breaking the layout. Using the recommendations below, sites not adding any @viewport rules themselves will continue to look and function like they always have.

11.1. Large screen UA styles

If a user agent has an initial viewport size large enough to fit common documents without breaking the layout, or which can easily to resized to do so, the recommendation is to have *no* UA styles. That means that it will have all descriptors initially set to auto, and behave as it would have without support for viewport descriptors.

If a user agent supports changing orientation, and the landscape mode's size fits common documents as described above but the portrait mode's size does not, it is recommended to set a minimum layout width equal to that of the width in landscape mode.

```
@viewport {
 min-width: 1024px;
}
```

11.2. Small screen UA styles

For smaller screen UAs, the UA could set the minimum viewport width to typically used as an initial viewport width of a traditional user agent (as described above).

```
@viewport {
 min-width: 980px;
}
```

It is recommended that limitations in zooming capabilities are not reflected in the UA styles but rather only affect the used values for zoom. The min-zoom/max-zoom UA styles mentioned in the Viewport META section are there to give an accurate description of how the Safari browser behaves, not as part of a recommended UA stylesheet.

Appendix A. Changes

This appendix is *informative*.

This appendix describes changes from the 15 September 2011 First Public Working Draft.

- Made various editorial improvements and clarifications.
- Added OM Interfaces.
- Added semi-colon as separator in meta viewport.
- Created UA stylesheets section.
- Added recommendation for when to respect orientation property.
- Dropped support for the resolution descriptor.
- Decouple width/height and zoom, introducing extend-to-zoom value for meta viewport translation.
- Made normative rules about interaction of @viewport and @media.
- Allow 0 for <viewport-length> and zoom values
- Removed support for device-width/height.
- Apply @viewport to top level document only.
- Extend [CSS3-CONDITIONAL] rather than CSS21 for nesting in @media.

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:
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 , 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.

Requirements for Responsible Implementation of CSS

The following sections define several conformance requirements for implementing CSS responsibly, in a way that promotes interoperability in the present and future.

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.

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.

Implementations of CR-level Features

Once a specification reaches the Candidate Recommendation stage, implementers should release an unprefixed implementation of any CR-level feature they can demonstrate to be correctly implemented according to spec, and should avoid exposing a prefixed variant of that feature.

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.

Index

Terms defined by this specification

Terms defined by reference

- [css-cascade-4] defines the following terms:
- inherit
-
- [css-conditional-3] defines the following terms:
- @media
- @supports
- conditional group rules
- nested_statement
-
- [css-fonts-3] defines the following terms:
- font-size
-
- [css-page-3] defines the following terms:
- @page
-
- [CSS3VAL] defines the following terms:
- <length>
- <number>
- <percentage>
- {a,b}
- |
-
- [css-writing-modes-3] defines the following terms:
- direction
-
- [cssom-1] defines the following terms:
- CSSRule
- CSSStyleDeclaration
-
- [mediaqueries-4] defines the following terms:
- device-aspect-ratio
-

References

Normative References

[CSS-CASCADE-4]
Elika Etemad; Tab Atkins Jr.. CSS Cascading and Inheritance Level 4. 14 January 2016. CR. URL: http://dev.w3.org/csswg/css-cascade/
[CSS-CONDITIONAL-3]
CSS Conditional Rules Module Level 3 URL: https://drafts.csswg.org/css-conditional-3/
[CSS-FONTS-3]
John Daggett. CSS Fonts Module Level 3. 3 October 2013. CR. URL: http://dev.w3.org/csswg/css-fonts/
[CSS-PAGE-3]
CSS Paged Media Module Level 3 URL: https://drafts.csswg.org/css-page-3/
[CSS-WRITING-MODES-3]
Elika Etemad; Koji Ishii. CSS Writing Modes Level 3. 15 December 2015. CR. URL: http://dev.w3.org/csswg/css-writing-modes-3/
[CSS21]
Bert Bos; et al. Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification. 7 June 2011. REC. URL: http://www.w3.org/TR/CSS2
[CSS3-CONDITIONAL]
David Baron. CSS Conditional Rules Module Level 3. 4 April 2013. CR. URL: http://www.w3.org/TR/css3-conditional/
[CSS3SYN]
Tab Atkins Jr.; Simon Sapin. CSS Syntax Module Level 3. 20 February 2014. CR. URL: http://www.w3.org/TR/css-syntax-3/
[CSS3VAL]
Tab Atkins Jr.; Elika Etemad. CSS Values and Units Module Level 3. 11 June 2015. CR. URL: http://dev.w3.org/csswg/css-values/

[CSSOM-1]
Simon Pieters; Glenn Adams. CSS Object Model (CSSOM). 17 March 2016. WD. URL: https://drafts.csswg.org/cssom/
[MEDIAQ]
Florian Rivoal; et al. Media Queries. 19 June 2012. REC. URL: http://www.w3.org/TR/css3-mediaqueries/
[MEDIAQUERIES-4]
Florian Rivoal; Tab Atkins Jr.. Media Queries Level 4. 26 January 2016. WD. URL: http://dev.w3.org/csswg/mediaqueries4/
[RFC2119]
S. Bradner. Key words for use in RFCs to Indicate Requirement Levels. March 1997. Best Current Practice. URL: https://tools.ietf.org/html/rfc2119

Informative References

[Algorithms]
Thomas H. Cormen; et al. Introduction to Algorithms, Second Edition, MIT Press.

Property Index

No properties defined.

@viewport Descriptors

Name	Value	Initial	Computed value	Media	Percentages
min-width	<viewport-length>	auto	auto, an absolute length, or a percentage as specified	visual, continuous	Refer to the width of the initial viewport
max-width	<viewport-length>	auto	auto, an absolute length, or a percentage as specified	visual, continuous	Refer to the width of the initial viewport
width	<viewport-length>{1,2}	See individual descriptors	See individual descriptors	visual, continuous	See individual descriptors
min-height	<viewport-length>	auto	auto, an absolute length, or a percentage as specified	visual, continuous	Refer to the height of the initial viewport
max-height	<viewport-length>	auto	auto, an absolute length, or a percentage as specified	visual, continuous	Refer to the height of the initial viewport
height	<viewport-length>{1,2}	See individual descriptors	See individual descriptors	visual, continuous	See individual descriptors
zoom	auto | <number> | <percentage>	auto	auto, or a non-negative number or percentage as specified	visual, continuous	The zoom factor itself
min-zoom	auto | <number> | <percentage>	auto	auto, or a non-negative number or percentage as specified	visual, continuous	The zoom factor itself
max-zoom	auto | <number> | <percentage>	auto	auto, or a non-negative number or percentage as specified	visual, continuous	The zoom factor itself
user-zoom	zoom | fixed	zoom	as specified	visual, continuous	N/A
orientation	auto | portrait | landscape	auto	as specified	visual, continuous	N/A

IDL Index

```
partial interface CSSRule {
 const unsigned short VIEWPORT_RULE = 15;
};

interface CSSViewportRule : CSSRule {
 readonly attribute CSSStyleDeclaration style;
};
```

Issues Index

This specification is written from an implementation centric point of view, making it arguably difficult to read. Significant editorial work may be needed to make it more understandable to different audiences. It also should clarify which viewport is referred to by various js APIs. See this blog post by ppk for a good discussion of these issues.
Various issues about this specification and related specifications are listed in this report.
"dbaron: The question is, what does this do on the desktop browser? (And what's a desktop browser)". Need to say that a "desktop" browser typically have no UA styles, as opposed to the UA stylesheet outlined for current mobile behaviour, and that no UA styles for @viewport will give "desktop" behaviour per default (actual viewport is initial viewport).
min- and max- functionality can be achieved with media queries, should these be removed?
The user-agent stylesheets recommended in the informative section don't adequately represent current implementation behaviors. Should there be a more explicit mechanism for switching between UA default behavior and requesting the CSS pixel?