CSS media queries

Contents:

A **media query** consists of a media type and at least one expression that limits the style sheets' scope by using media features, such as width, height, and color. Media queries, added in deprecated CSS3, let the presentation of content be tailored to a specific range of output devices without having to change the content itself.

Syntax

Media queries consist of a media type and can, as of the CSS3 specification, contain one or more expressions, expressed as media features, which resolve to either true or false.

The result of the query is true if the media type specified in the media query matches the type of device the document is being displayed on **and** all expressions in the media query are true.

```
<!-- CSS media query on a link element -->
<link rel="stylesheet" media="(max-width: 767px)" href="mobile.css" />

<!-- CSS media query within a style sheet -->
<style type="text/css">
        @media (max-width: 640px) {
          .sidebar {
                display: none;
          }
        }
</style>
```

When a media query is true, the corresponding style sheet or styles are applied, following the normal cascading rules. Style sheets with media queries attached to their <link> tags will still download even if their media queries would return false (they will not apply, however).

Unless you use the “not” or “only” operators, the media type is optional and the all type will be implied.

Pseudo-BNF

```
media_query_list: <media_query> [, <media_query> ]*
media_query: [[only | not]? <media_type> [ and <expression> ]*]
  | <expression> [and <expression> ]*
expression: ( <media_feature> [: <value>]? )
media_type: all | aural | braille | handheld | print | projection | screen | tty | tv | embossed
media_feature: width | min-width | max-width
  | height | min-height | max-height
  | device-width | min-device-width | max-device-width
  | device-height | min-device-height | max-device-height
  | aspect-ratio | min-aspect-ratio | max-aspect-ratio
  | device-aspect-ratio | min-device-aspect-ratio | max-device-aspect-ratio
  | color | min-color | max-color
  | color-index | min-color-index | max-color-index
  | monochrome | min-monochrome | max-monochrome
  | resolution | min-resolution | max-resolution
  | scan | grid
```

- Media queries are case insensitive.
- Media queries involving unknown media types are always false.

Note: *Parentheses are required around expressions; failing to use them is an error.*

Logical operators for media query expressions

You can compose complex media queries using logical operators: **not**, **and** & **only**.

The **and** operator is used for combining multiple media features together into a single media query, requiring each chained feature to return true in order for the query to be true. The not operator is used to negate an entire media query. The only operator is used to apply a style only if the entire query matches, useful for preventing older browsers from applying selected styles. If you use the not or only operators, you *must* specify an explicit media type.

You can also combine multiple media queries in a comma-separated list; if any of the media queries in the list is true, the entire media statement returns true. This is equivalent to an 'or' operator.

i. and

The and keyword is used for combining multiple media features together, as well as combining media features with media types. A basic media query, a single media feature with the implied all media type, could look like this:

```
@media (min-width: 700px) { ... }
```

If, however, you wanted this to apply only if the display is in landscape, you could use an 'and' operator to chain the media features together.

```
@media (min-width: 700px) and (orientation: landscape) { ... }
```

Now the above media query will only return true if the viewport is 700px wide or wider and the display is in landscape. If, however, you only wanted this to apply if the display in question was of the media type TV, you could chain these features with a media type using an and operator.

```
@media tv and (min-width: 700px) and (orientation: landscape) { ... }
```

Now, the above media query will only apply if the media type is TV, the viewport is 700px wide or wider, and the display is in landscape.

comma-separated lists

Comma-separated lists behave like the logical operator or when used in media queries. When using a comma-separated list of media queries, if any of the media queries returns true, the styles or style sheets get applied. Each media query in a comma-separated list is treated as an individual query, and any operator applied to one media query does not affect the others. This means the comma-separated media queries can target different media features, types, and states.

For instance, if you wanted to apply a set of styles if the viewing device either had a minimum width of 700px or was a handheld in landscape, you could write the following:

```
@media (min-width: 700px), handheld and (orientation: landscape) { ... }
```

Above, if I were on a screen device with a viewport width of 800px, the media statement would return true because the first part, interpreted as @media all and (min-width: 700px) would apply to my device and therefore return true, despite the fact that my screen device would fail the handheld media type check in the second media query. Likewise, if I were on a handheld device held in landscape with a viewport width of 500px, while the first media query would fail due to the viewport width, the second media query would succeed and thus the media statement would return true.

ii. not

The not keyword applies to the whole media query and returns true if the media query would otherwise return false (such as monochrome on a color display or a screen width of 600px with a min-width: 700px feature query). A not will only negate the media query it is applied to and not to every media query if present in a comma-separated list of media queries.
The not keyword cannot be used to negate an individual feature query, only an entire media query. For example, the not is evaluated last in the following query:

```
@media not all and (monochrome) { ... }
```

This means that the query is evaluated
Like this: `@media not (all and (monochrome)) { ... }`
Rather than this: ~~`@media (not all) and (monochrome) { ... }`~~

As another example, check the media query: `@media not screen and (color), print and (color)`
It is evaluated like this: `@media (not (screen and (color))), print and (color)`

iii. only

The only keyword prevents older browsers that do not support media queries with media features from applying the given styles:

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

Media features in detail

Most media features can be prefixed with "min-" or "max-" to express "greater or equal to" or "less than or equal to" constraints. This avoids using the "<" and ">" symbols, which would conflict with HTML and XML. If you use a media feature without specifying a value, the expression resolves to true if the feature's value is non-zero.

***Note**: If a media feature doesn't apply to the device on which the browser is running, expressions involving that media feature are always false. For example, querying the aspect ratio of an aural device always results in false.*

i. color

Value: <color>
Media: visual
Accepts min/max prefixes: yes

Indicates the number of bits per color component of the output device. If the device is not a color device, this value is zero.

***Note**: If the color components have different numbers of bits per color component, the smallest number is used. For example, if a display uses 5 bits for blue and red and 6 bits for green, then the device is considered to use 5 bits per color component. If the device uses indexed colors, the minimum number of bits per color component in the color table is used.*

Examples

To apply a style sheet to all color devices:

```
@media all and (color) { ... }
```

To apply a style sheet to devices with at least 4 bits per color component:

```
@media all and (min-color: 4) { ... }
```

ii. color-index

Value: <integer>
Media: visual
Accepts min/max prefixes: yes

Indicates the number of entries in the color look-up table for the output device.

Examples

To indicate that a style sheet should apply to all devices using indexed color, you can do:

```
@media all and (color-index) { ... }
```

To apply a style sheet to indexed color devices with at least 256 colors:

```
<link rel="stylesheet" media="all and (min-color-index: 256)" href="stylesheet.css" />
```

iii. aspect-ratio

Value: <ratio>
Media: visual, tactile
Accepts min/max prefixes: yes

Describes the aspect ratio of the targeted display area of the output device. This value consists of two positive integers separated by a slash ("/") character. This represents the ratio of horizontal pixels (first term) to vertical pixels (second term).

Example

The following selects a special style sheet to use for when the display area is at least as wide as it is high.

```
@media screen and (min-aspect-ratio: 1/1) { ... }
```

This selects the style when the aspect ratio is either 1:1 or greater. In other words, these styles will only be applied when the viewing area is square or landscape.

iv. device-aspect-ratio

Value: <ratio>
Media: visual, tactile
Accepts min/max prefixes: yes

Describes the aspect ratio of the output device. This value consists of two positive integers separated by a slash ("/") character. This represents the ratio of horizontal pixels (first term) to vertical pixels (second term).

Example

The following selects a special style sheet to use for widescreen displays.

```
@media screen and (device-aspect-ratio: 16/9), screen and (device-aspect-ratio: 16/10) {...}
```

This selects the style when the aspect ratio is either 16:9 or 16:10.

v. device-height

Value: <length>
Media: visual, tactile
Accepts min/max prefixes: yes

Describes the height of the output device (meaning the entire screen or page, rather than just the rendering area, such as the document window).

Example

To apply a style sheet to a document when displayed on a screen that is less than 800 pixels long, you can use this:

```
<link rel="stylesheet" media="screen and (max-device-height: 799px)" />
```

vi. device-width

Value: <length>
Media: visual, tactile
Accepts min/max prefixes: yes

Describes the width of the output device (meaning the entire screen or page, rather than just the rendering area, such as the document window).

Example

To apply a style sheet to a document when displayed on a screen that is less than 800 pixels wide, you can use this:

```
<link rel="stylesheet" media="screen and (max-device-width: 799px)" />
```

vii. grid

Value: <integer>
Media: all
Accepts min/max prefixes: no

Determines whether the output device is a grid device or a bitmap device. If the device is grid-based (such as a TTY terminal or a phone display with only one font), the value is 1. Otherwise it is zero.

Example

To apply a style to handheld devices with a 15-character or narrower display:

```
@media handheld and (grid) and (max-width: 15em) { ... }
```

Note: *The "em" unit has a special meaning for grid devices; since the exact width of an "em" can't be determined, 1em is assumed to be the width of one grid cell horizontally, and the height of one cell vertically.*

viii. height

Value: <length>
Media: visual, tactile
Accepts min/max prefixes: yes

The height media feature describes the height of the output device's rendering surface (such as the height of the viewport or of the page box on a printer).

Note: *As the user resizes the window, Firefox switches style sheets as appropriate based on media queries using the* width *and* height *media features.*

ix. monochrome

Value: <integer>
Media: visual
Accepts min/max prefixes: yes
Indicates the number of bits per pixel on a monochrome (greyscale) device. If the device isn't monochrome, the device's value is 0.
Examples
To apply a style sheet to all monochrome devices:
```
@media all and (monochrome) { ... }
```
To apply a style sheet to monochrome devices with at least 8 bits per pixel:
```
@media all and (min-monochrome: 8) { ... }
```

x. orientation

Value: landscape | portrait
Media: visual
Accepts min/max prefixes: no
Indicates whether the viewport is in landscape (the display is wider than it is tall) or portrait (the display is taller than it is wide) mode.
Example
To apply a style sheet only in portrait orientation:
```
@media all and (orientation: portrait) { ... }
```

xi. resolution

Value: <resolution>
Media: bitmap
Accepts min/max prefixes: yes
Indicates the resolution (pixel density) of the output device. The resolution may be specified in either dots per inch (dpi) or dots per centimeter (dpcm).
Examples
To apply a style sheet to devices with at least 300 dots per inch of resolution:
```
@media print and (min-resolution: 300dpi) { ... }
```
To replace the old (min-device-pixel-ratio: 2) syntax:
```
@media screen and (min-resolution: 2dppx) { ... }
```

xii. scan

Value: progressive | interlace
Media: tv
Accepts min/max prefixes: no
Describes the scanning process of television output devices.
Example
To apply a style sheet only to progressive scanning televisions:
```
@media tv and (scan: progressive) { ... }
```

xiii. width

Value: <length>
Media: visual, tactile
Accepts min/max prefixes: yes
The width media feature describes the width of the rendering surface of the output device (width of the document window or width of the page box on a printer).
***Note**: As the user resizes the window, Firefox switches style sheets as appropriate based on media queries using the* width *and* height *media features.*
Examples
This media query specifies a style sheet that applies to printed media wider than 8.5 inches:
```
<link rel="stylesheet" media="print and (min-width: 8.5in)" href="mystyle.css" />
```
This query specifies a style sheet that is usable when the viewport is between 500 and 800 pixels wide:
```
@media screen and (min-width: 500px) and (max-width: 800px) { ... }
```