Chapter 13: Media Queries

Parameter	Details
`mediatype`	(Optional) This is the type of media. Could be anything in the range of `all` to `screen`.
`not`	(Optional) Doesn't apply the CSS for this particular media type and applies for everything else.
`media feature`	Logic to identify use case for CSS. Options outlined below.
Media Feature	**Details**
`aspect-ratio`	Describes the aspect ratio of the targeted display area of the output device.
`color`	Indicates the number of bits per color component of the output device. If the device is not a color device, this value is zero.
`color-index`	Indicates the number of entries in the color look-up table for the output device.
`grid`	Determines whether the output device is a grid device or a bitmap device.
`height`	The height media feature describes the height of the output device's rendering surface.
`max-width`	CSS will not apply on a screen width wider than specified.
`min-width`	CSS will not apply on a screen width narrower than specified.
`max-height`	CSS will not apply on a screen height taller than specified.
`min-height`	CSS will not apply on a screen height shorter than specified.
`monochrome`	Indicates the number of bits per pixel on a monochrome (greyscale) device.
`orientation`	CSS will only display if device is using specified orientation. See remarks for more details.
`resolution`	Indicates the resolution (pixel density) of the output device.
`scan`	Describes the scanning process of television output devices.
`width`	The width media feature describes the width of the rendering surface of the output device (such as the width of the document window, or the width of the page box on a printer).
Deprecated Features	**Details**
`device-aspect-ratio`	**`Deprecated`** CSS will only display on devices whose height/width ratio matches the specified ratio. This is a`deprecated`feature and is not guaranteed to work.
`max-device-width`	**`Deprecated`** Same as `max-width` but measures the physical screen width, rather than the display width of the browser.
`min-device-width`	**`Deprecated`** Same as `min-width` but measures the physical screen width, rather than the display width of the browser.
`max-device-height`	**`Deprecated`** Same as `max-height` but measures the physical screen width, rather than the display width of the browser.
`min-device-height`	**`Deprecated`** Same as `min-height` but measures the physical screen width, rather than the display width of the browser.

Section 13.1: Terminology and Structure

Media queries allow one to apply CSS rules based on the type of device / media (e.g. screen, print or handheld) called **media type**, additional aspects of the device are described with **media features** such as the availability of color or viewport dimensions.

General Structure of a Media Query

```
@media [...] {
    /* One or more CSS rules to apply when the query is satisfied */
}
```

A Media Query containing a Media Type

```
@media print {
    /* One or more CSS rules to apply when the query is satisfied */
```

```
}
```

A Media Query containing a Media Type and a Media Feature

```
@media screen and (max-width: 600px) {
    /* One or more CSS rules to apply when the query is satisfied */
}
```

A Media Query containing a Media Feature (and an implicit Media Type of "all")

```
@media (orientation: portrait) {
    /* One or more CSS rules to apply when the query is satisfied */
}
```

Section 13.2: Basic Example

```
@media screen and (min-width: 720px) {
    body {
        background-color: skyblue;
    }
}
```

The above media query specifies two conditions:

1. The page must be viewed on a normal screen (not a printed page, projector, etc).
2. The width of the user's view port must be at least 720 pixels.

If these conditions are met, the styles inside the media query will be active, and the background color of the page will be sky blue.

Media queries are applied dynamically. If on page load the conditions specified in the media query are met, the CSS will be applied, but will be immediately disabled should the conditions cease to be met. Conversely, if the conditions are initially not met, the CSS will not be applied until the specified conditions are met.

In our example, if the user's view port width is initially greater than 720 pixels, but the user shrinks the browser's width, the background color will cease to be sky blue as soon as the user has resized the view port to less than 720 pixels in width.

Section 13.3: mediatype

Media queries have an optional `mediatype` parameter. This parameter is placed directly after the `@media` declaration (`@media mediatype`), for example:

```
@media print {
    html {
        background-color: white;
    }
}
```

The above CSS code will give the DOM `HTML` element a white background color when being printed.

The `mediatype` parameter has an optional `not` or `only` prefix that will apply the styles to everything except the specified mediatype *or* only the specified media type, respectively. For example, the following code example will apply the style to every media type except `print`.

```
@media not print {
    html {
        background-color: green;
    }
```

```
}
```

And the same way, for just showing it only on the screen, this can be used:

```
@media only screen {
    .fadeInEffects {
        display: block;
    }
}
```

The list of `mediatype` can be understood better with the following table:

Media Type	Description
`all`	Apply to all devices
`screen`	Default computers
`print`	Printers in general. Used to style print-versions of websites
`handheld`	PDA's, cellphones and hand-held devices with a small screen
`projection`	For projected presentation, for example projectors
`aural`	Speech Systems
`braille`	Braille tactile devices
`embossed`	Paged braille printers
`tv`	Television-type devices
`tty`	Devices with a fixed-pitch character grid. Terminals, portables.

Section 13.4: Media Queries for Retina and Non Retina Screens

Although this works only for WebKit based browsers, this is helpful:

```
/* ----------- Non-Retina Screens ----------- */
@media screen
  and (min-width: 1200px)
  and (max-width: 1600px)
  and (-webkit-min-device-pixel-ratio: 1) {
}

/* ----------- Retina Screens ----------- */
@media screen
  and (min-width: 1200px)
  and (max-width: 1600px)
  and (-webkit-min-device-pixel-ratio: 2)
  and (min-resolution: 192dpi) {
}
```

Background Information

There are two types of pixels in the display. One is the logical pixels and the other is the physical pixels. Mostly, the physical pixels always stay the same, because it is the same for all the display devices. The logical pixels change based on the resolution of the devices to display higher quality pixels. The device pixel ratio is the ratio between physical pixels and logical pixels. For instance, the MacBook Pro Retina, iPhone 4 and above report a device pixel ratio of 2, because the physical linear resolution is double the logical resolution.

The reason why this works only with WebKit based browsers is because of:

- The vendor prefix `-webkit-` before the rule.
- This hasn't been implemented in engines other than WebKit and Blink.

Section 13.5: Width vs Viewport

When we are using "width" with media queries it is important to set the meta tag correctly. Basic meta tag looks like this and it needs to be put inside the **<head>** tag.

```
<meta name="viewport" content="width=device-width,initial-scale=1">
```

Why this is important?

Based on MDN's definition "width" is

> The width media feature describes the width of the rendering surface of the output device (such as the width of the document window, or the width of the page box on a printer).

What does that mean?

View-port is the width of the device itself. If your screen resolution says the resolution is 1280 x 720, your view-port width is "1280px".

More often many devices allocate different pixel amount to display one pixel. For an example iPhone 6 Plus has 1242 x 2208 resolution. But the actual viewport-width and viewport-height is 414 x 736. That means 3 pixels are used to create 1 pixel.

But if you did not set the `meta` tag correctly it will try to show your webpage with its native resolution which results in a zoomed out view (smaller texts and images).

Section 13.6: Using Media Queries to Target Different Screen Sizes

Often times, responsive web design involves media queries, which are CSS blocks that are only executed if a condition is satisfied. This is useful for responsive web design because you can use media queries to specify different CSS styles for the mobile version of your website versus the desktop version.

```
@media only screen and (min-width: 300px) and (max-width: 767px) {
    .site-title {
        font-size: 80%;
    }

    /* Styles in this block are only applied if the screen size is atleast 300px wide, but no more
than 767px */
}

@media only screen and (min-width: 768px) and (max-width: 1023px) {
    .site-title {
        font-size: 90%;
    }

    /* Styles in this block are only applied if the screen size is atleast 768px wide, but no more
than 1023px */
}

@media only screen and (min-width: 1024px) {
```

```
    .site-title {
        font-size: 120%;
    }

    /* Styles in this block are only applied if the screen size is over 1024px wide. */
}
```

Section 13.7: Use on link tag

```
<link rel="stylesheet" media="min-width: 600px" href="example.css" />
```

This stylesheet is still downloaded but is applied only on devices with screen width larger than 600px.

Section 13.8: Media queries and IE8

Media queries are not supported at all in IE8 and below.

A Javascript based workaround

To add support for IE8, you could use one of several JS solutions. For example, **Respond** can be added to add media query support for IE8 only with the following code :

```
<!--[if lt IE 9]>
<script
   src="respond.min.js">
</script>
<![endif]-->
```

CSS Mediaqueries is another library that does the same thing. The code for adding that library to your HTML would be identical :

```
<!--[if lt IE 9]>
<script
   src="css3-mediaqueries.js">
</script>
<![endif]-->
```

The alternative

If you don't like a JS based solution, you should also consider adding an IE<9 only stylesheet where you adjust your styling specific to IE<9. For that, you should add the following HTML to your code:

```
<!--[if lt IE 9]>
<link rel="stylesheet" type="text/css" media="all" href="style-ielt9.css"/>
<![endif]-->
```

Note :

Technically it's one more alternative: using **CSS hacks** to target IE<9. It has the same impact as an IE<9 only stylesheet, but you don't need a separate stylesheet for that. I do not recommend this option, though, as they produce invalid CSS code (which is but one of several reasons why the use of CSS hacks is generally frowned upon today).