Understanding Media Queries

Media queries are an essential part of responsive design. Using media queries, designers can apply different styles and adapt layouts for different screen sizes and resolutions. The syntax for a media query is listed below.

```
@media not|only mediatype and (mediafeature and|or|not mediafeature) {
   CSS code {
     ...
}
 }
```

The @media syntax consists of:

- A media type — such as screen (used for computer screens, tablets, or mobile phones), print (used for printers), or all (used for all media devices — this is the default if nothing is defined)
- One or more media features that define the parameters — such as screen width or resolution
- CSS code placed inside curly brackets — { CSS code } — that define style rules

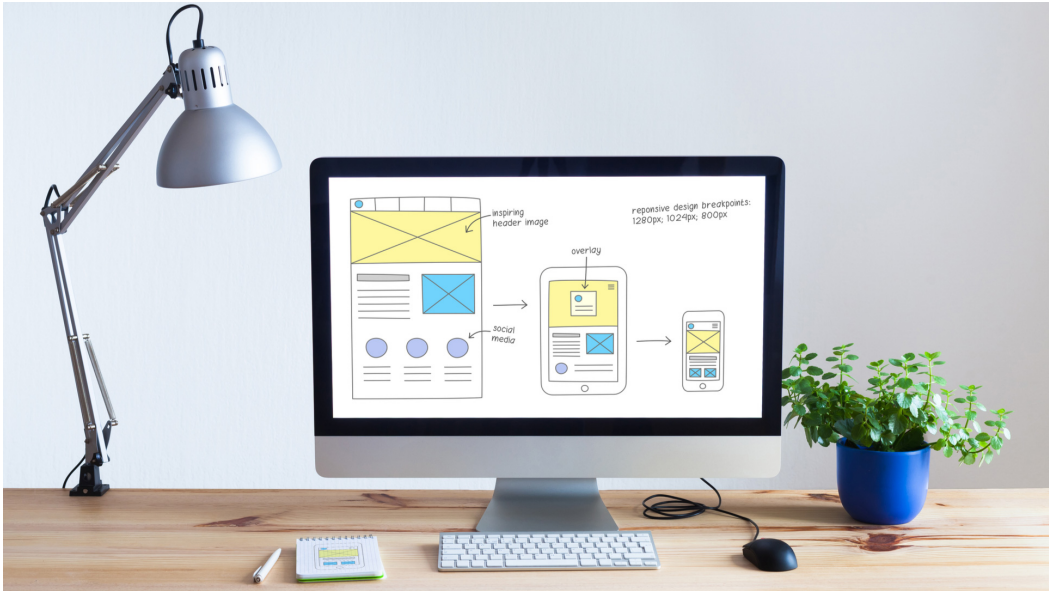
A media query is a logical expression that is resolved to be either true or false. The result of a media query is true if the media query matches the type of device the document is being displayed on. If this is the case, then all the related CSS styles in the media query are applied to the target device.

In other words, the @media at-rule applies part of a CSS style sheet based on the result of a media query. This CSS construct is a way of testing for certain conditions in your CSS, such as:

- How wide the browser window is, including a minimum, maximum, or a range of sizes
- Whether you are using a screen display or preparing a web page for print
- Whether the screen is oriented in portrait or landscape displays
- Or a combination of these conditions

Media queries may be placed directly into a webpage's HTML or included in a separate .CSS file that is referenced by the webpage. There is no limit on the number of media queries a .CSS file may contain. A designer can place as many media queries as the design requires, and may combine media queries to get the exact layout results desired.

Breakpoints



Typically, within a CSS document, the default styles are defined first, followed by the media queries for different screen sizes.

Breakpoints are used with media queries and are the points where parts of the website design will respond differently on each side of the breakpoint. For the purposes of responsive design, we normally concentrate on the width of the browser's window. The CSS inside of the media query is what describes what we want to do at a given width. The following table lists the frequently used breakpoint groups.

Frequently Used Device Breakpoints	
Device	**Breakpoint**
Extra small devices (phones, 599px and down)	`@media (max-width: 599px) { CSS code { ... } }`
Small devices (portrait tablets and large phones, 600px and up)	`@media (min-width: 600px) { CSS code { ... } }`
Medium devices (landscape tablets, 768px and up)	`@media (min-width: 768px) { CSS code { ... } }`
Large devices (laptops/desktops, 992px and up)	`@media (min-width: 992px) { CSS code { ... } }`
Extra large devices (large laptops and desktops, 1200px and up)	`@media (min-width: 1200px) { CSS code { ... } }`

** The CSS code for the style rules to be applied based on the query is placed between the curly brackets — { ... }.

Media Queries: Desktop-First versus Mobile-First Approach

Like all modules within a cascading style sheet, the ones that appear further down the list override the ones above them. **Desktop-first** designs for large screen layouts first, and then uses media queries to override any settings for smaller devices. The opposite, known as **mobile-first**, defines styles for mobile devices first, followed by media queries to override any settings for larger devices. "Mobile-first" development will be discussed in more depth in a later assignment in this course.

Example:

Consider the following CSS. In this example, the media query starts with an @media statement, followed by a parameter to test. We are testing to see if the maximum width is 400px. This is followed by a curly brace, which must be closed at the end of the media query.

For this specific media query, if the query is true — i.e., the device has a maximum width of 400px (or less) — the background color will be blue (per the CSS styling inside the media query). If the media query is false, the background color will be green (again per the CSS styling).

```
body {
      background-color: green;
```

```
      }

@media (max-width: 400px) {
       body {
            background-color: blue;
       }
      }
```

Let's add this CSS styling to the code tool below and see how it affects the browser display.

Instructions:

- Click on the "styles.css" link tab at the top of the code tool.
- Copy the CSS code above and then paste it into the CSS stylesheet window of the code tool.
- Click the 'run code' button and watch what happens.

Run Code | Save | Export | Reset

index.html styles.css

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <link rel="stylesheet" href="/styles.css" />
5    </head>
6    <body>
7      Adjust the size of your browser and watch the background color change
8    </body>
9  </html>
10
11
12
```

Theme Select:

rubyblue

By adding the CSS, we've set the background color on the body of our document to green. However, if the width of the browser window is 400 pixels wide or less, then the background of the document will be blue.

Wait! I Only See a Green Background?

Only seeing a green background? That is because the size of the viewport in the code tool is more than 400 pixels wide. Adjust the size of your browser window, and notice that when you narrow the window past a certain point, the background of the code tool display window will change to blue as the viewport is less than 400 pixels wide.

Other Examples of Media Query Syntax

CSS Media Query: Width Range

Media queries can also define a width range. The media query below indicates a width range from a minimum of 400px to a maximum of 700px.

```
@media (min-width: 400px) and (max-width: 700px) {
        ...
        }
```

CSS Media Query: Print

The following media query is valid for printing only, with a maximum width of 7.25 inches. While we don't see inches used often in CSS for obvious reasons, it may make perfect sense to use this dimension when printing web pages.

```
@media print and (max-width: 7.25in) {
   ...
 }
```

Exercise: Applying Media Queries to set Background Color with Desktop-First Approach

Now that you're familiar with media query syntax, let's apply three different media queries to set the background color using a desktop-first approach. Using this approach, the global styles are designed for a desktop screen, with media queries then added for smaller screen sizes. Therefore, with this approach, **max-width** is used to define the width range in the media queries.

Instructions:

Define the media queries based on the following screen widths.

If the screen width is:

- Greater than 500px, set the background-color to "lightblue"
- Between 301px and 500px, set the backgroud-color to "lightyellow"
- 300px or less, set the backgroud-color to "lightgreen"

Click on the 'css.styles' link in the code tool below. Then, add the appropriate media query parameters to the CSS style sheet in the code tool. When you have finished, click 'run code' and be sure to resize your browser window to see the background display changes.

Remember: In a cascading style sheet, the styles that appear further down the list override the ones above them. Therefore, the default styles are defined first, followed by the media queries for different screen sizes.

Run Code | Save | Export | Reset

index.html styles.css

```
<!doctype html>
<html>
```

```
 <head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="./styles.css"/>
</head>
 <body>
   <p>Remember to resize your browser window!</p>
   <p>Watch to see background color change as the viewport size changes.</p>
</body>
</html>
```

Theme Select:

rubyblue

Click on the following tab to check your answer.

Check Your Answer

The following is the correct CSS code:

```
    body {
  background-color: lightblue;
}

@media screen and (max-width: 500px) {
  body {
    background-color: lightyellow;
  }
}

@media screen and (max-width: 300px) {
  body {
    background-color: lightgreen;
  }
}
```

Click 'run code' and then resize your browser window to see the style effects that are applied by the different media queries.

Run Code Save Export Reset

index.html styles.css

Theme Select:

rubyblue