Mobile Responsiveness

Basics

Media queries effectively create "conditional css". If a particular width of the screen is met, then the CSS within that media query will be triggered. Otherwise it will be ignored. While Bootstrap CSS allows us to have a fully mobile responsive websites, it's still important to be familiar with media queries so you can roll out your own custom mobile responsive designs in the future. The homework for this week will cover both building a responsive site in Bootstrap and one without Bootstrap.

Meta Viewport Tag

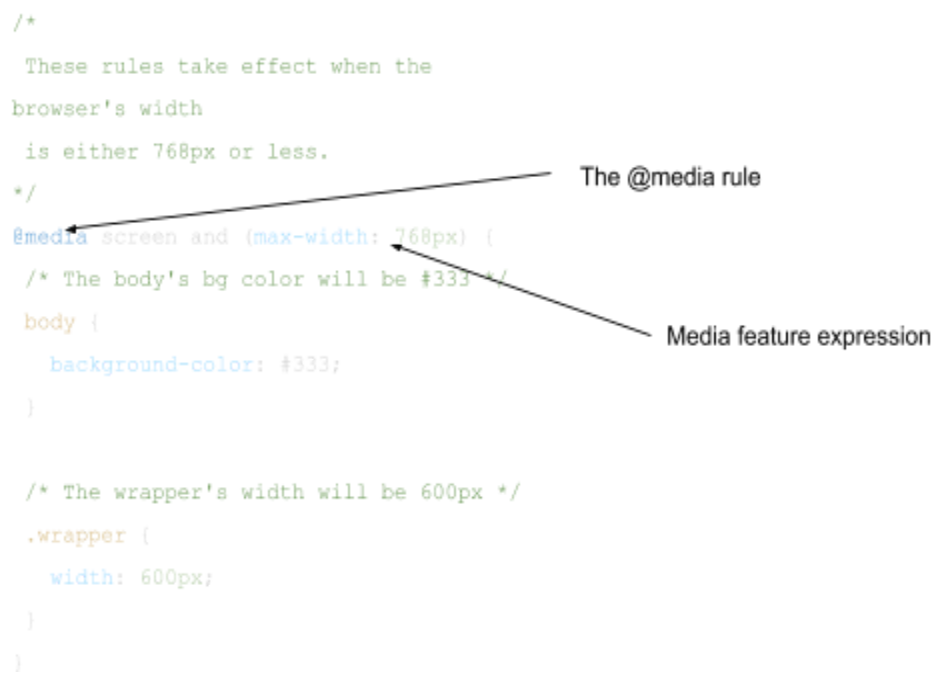
This is low priority to understand; it must just simply be done. The meta viewport tag is required for the Chrome Responsive Tools to actually function, so it's important we always add it to the head. To learn more about this meta tag: http://www.w3schools.com/css/css_rwd_viewport.asp and https://developer.mozilla.org/en-US/docs/Mozilla/Mobile/Viewport_meta_tag.

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

Media Queries

A media query is composed of an optional *media type* and any number of *media feature* expressions. Media queries are case-insensitive.

A media query computes to true when the media type matches the device on which a document is being displayed *and* all media feature expressions are true.



Common Media Feature Expressions

Max Width: Applies to screens N wide and less. Example below from W3Schools.

```
@media screen and (max-width: 992px) {
  body {
      background-color: blue;
  }
}
```

Min Width: Applies to screens: Applies to screens at least N wide and more. Example below from W3Schools.

```
@media screen and (min-width: 600px) {
  div.example {
      font-size: 50px;
      padding: 50px;
      border: 8px solid black;
      background: yellow;
  }
}
```

Min and Max Combination: Applies to screens at least N wide but no more than X wide. Example below from W3Schools.

```
@media screen and (max-width: 900px) and (min-width: 600px) {
  div.example {
      font-size: 50px;
      padding: 50px;
      border: 8px solid black;
      background: yellow;
  }
}
```

Breakpoints

You will often hear the term “breakpoints” during your career. This is really just shorthand nomenclature for media query syntax. Example: “One breakpoint is 768px.” This just means that you will be adding a media query such as the ones above where min or max width is 768px. Based on the needs of the design and your company’s rules about how to declare media queries (if they have them) will determine whether it is max or min.

Defining The Breakpoints

This is a contentious issue in the web developer community. There are 2 major ways that developers and designers define breakpoints:

1. Determining at what widths the content begins to display poorly and going from there.
2. Basing on common device sizes:
    a. 320px - smart phones
    b. 768px - tablets
    c. > 768 - everything else

My philosophy is to follow the first, but most companies I have worked for do the latter. Be prepared during an interview to discuss the pros and cons of each and be flexible to implement either solution based on the direction your employer/team chooses to take.

- https://responsivedesign.is/develop/browser-feature-support/media-queries-for-common-device-breakpoints/
- https://medium.freecodecamp.org/the-100-correct-way-to-do-css-breakpoints-88d6a5ba1862
- https://css-tricks.com/snippets/css/media-queries-for-standard-devices/
- https://stackoverflow.com/questions/16443380/common-css-media-queries-break-points

The Code Beyond the Media Queries

Instead of thinking in terms of hard-coded pixel widths, you will want to think in percentages. Explore how the browser reacts when you have elements that are 960px in width versus 85% width. The percentages allow for the layout to be “liquid”. It will always be a percentage of the browser, rather than a set width of the browser.

Crunching Numbers

1. Determine the largest monitor size for which the mock was created. For example, you might receive a mock that has content that is 960px wide, targeting a monitor that is 1220px wide (you will get this from the designer -- who might be you!)
2. Divide content monitor width by the content width to determine the percentage the content width should be: (960 / 1220) * 100 in our example. The width of the content area should be 78.6% (in this example). This will keep the content width proportions true to the designer’s intent at different device widths. This width might change at different breakpoints.
3. An element’s percentage width is based on its parent container. For example, an image that is set to 100% is 100% of its container and **not** 100% of the page (unless the page is its main container!). That means that an image whose width is 100% of its container, whose width is 20% of the page, will be 20% of the page!
4. Speaking of images, for an image to be responsive, its width should always be set to 100%.

Resources

- https://www.webdesignrankings.com/resources/rqrwd/
- https://www.lifewire.com/width-calculations-responsive-site-4136178
- http://responsv.com/flexible-math/

Excellent Overall Resources

- https://developers.google.com/web/fundamentals/design-and-ux/responsive/
- https://www.w3schools.com/css/css_rwd_mediaqueries.asp
- https://www.w3schools.com/css/css3_mediaqueries_ex.asp
- https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries
- https://alistapart.com/article/responsive-web-design

- https://www.wired.com/2011/09/the-boston-globe-embraces-responsive-design/