{ Responsive Design. }

Objectives:

By the end of this chapter, you should be able to:

- Explain what responsive and mobile first design are
- Explain what a media query is
- Write media queries to allow for responsive design

Responsive + Mobile First Design

One thing to keep in mind is how your site looks on different devices. You might spend days meticulously crafting a beautifully designed site on your laptop, only to discover that it looks like garbage on your phone.

Responsive design is a way to approach designing your applications that is mindful of the fact that users don't all have screens of the same size. With a responsive design, the goal is to make your site beautiful for users on a variety of devices, while minimizing the amount of CSS you need to write.

One common approach to responsive design is to called *mobile first design*. A mobile-first approach forces you to think about mobile users before you think of desktop users. About 10 years ago, most web designs cared only about desktop or laptop users. Today most users browse on their phone more than they do on their laptop, so your design process needs to take the mobile user into consideration first, before the laptop user. Practically speaking this means that you should create css styles that are applied first to mobile devices. Advanced styles and other overrides for larger screens are then added into the stylesheet via *media queries*.

To get started with responsive design, we need to (1) understand how to write media queries, and (2) move from absolute measurement to relative measurement.

Media Queries

One of the fundamental building blocks of responsive design along with mobile first design is the ability to show different styles for different conditions. As with keyframes, media queries are declared using an at-rule. Here's a simple example of the syntax:

```
@media screen and (max-width: 480px) {
    /* add some styles for a mobile device here*/
}
```

You can check for a lot of things using media queries. Here are a few examples:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    @media (min-width: 480px) {
      body {
        background-color: red;
      }
    }

    @media (min-width: 540px) {
      body {
        background-color: orange;
      }
    }

    @media (min-width: 600px) {
      body {
        background-color: yellow;
      }
    }

    @media (min-width: 660px) {
      body {
        background-color: green;
      }
    }

    @media (min-width: 720px) {
      body {
        background-color: blue;
      }
    }

    @media (min-width: 780px) {
      body {
        background-color: purple;
      }
    }
  </style>
</head>
<body>
</body>
</html>
```

To see the effect of these media queries, open this page in Chrome and then drag the viewport so that it gets narrower and narrower. You should see the background color change as the window's width changes.

Here's another example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    #column1, #column2 {
      display: inline-block;
      width:50%;
      height: 400px;
    }

    #column1 {
      background-color: black;
    }

    #column2 {
      background-color: aqua;
    }

    @media (max-width: 700px) {
      #column1, #column2 {
        width: 100%;
      }
    }
  </style>
</head>
<body>
  <div id="column1"></div><div id="column2"></div>
</body>
</html>
```

This is an example of a two-column layout that collapses to a one-column layout on a narrow screen. If the window's width is more than 700 pixels, you can see that the black `div` and the `aqua` div are side-by-side, each taking up 50% of the available width. However, on narrower screens, they stack on top of each other. This a very common way to set columns that respond to the width of the screen; we'll see this in more detail when we use Bootstrap.

To see more examples, check out the MDN docs.