2.3.6 Introducing Media Queries

As we've seen overall, there is no shortage of CSS tools to get the job done. We've used them to create flexible layouts, provide different amounts of spacing using margin and padding, and at some point we'll even be able to animate elements on the page.

The CSS tool we're going to use now is what's known as a **media query**. A media query sets special CSS rule conditions that apply certain rules when a certain condition is met. The most common use for it is to apply different CSS rules to a page depending on a device's screen width.

Let's look at an example:

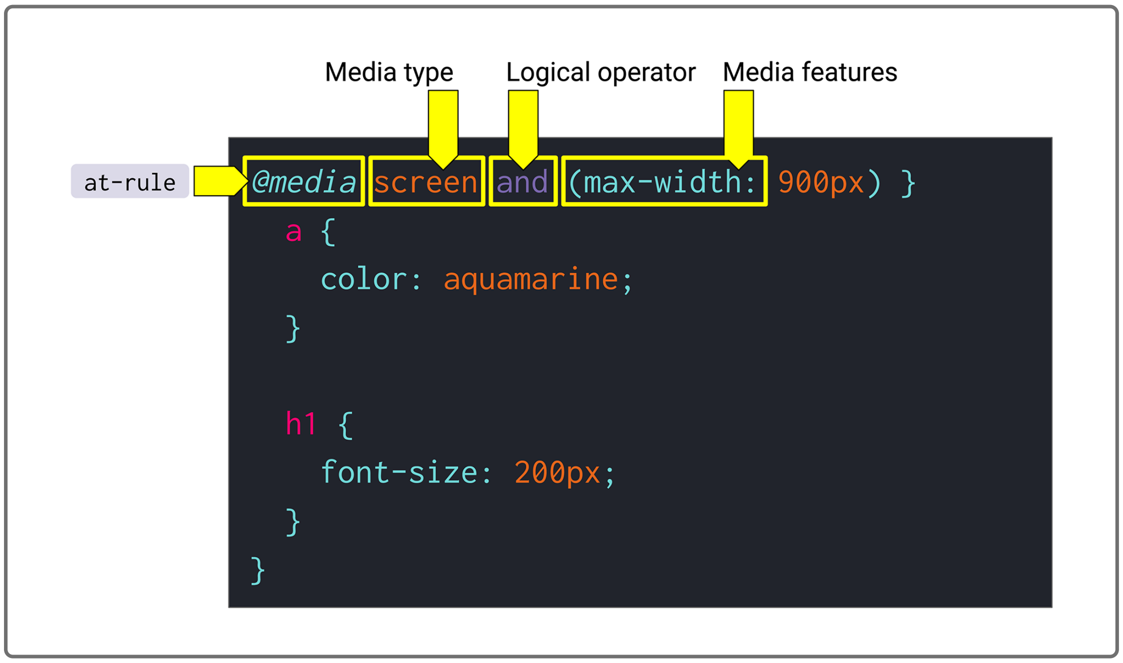
```
@media screen and (max-width: 900px) {
  a {
    color: aquamarine;
  }

  h1 {
    font-size: 200px;
  }
}
```

This is a very basic example of what a media query looks like in a style sheet. It can be made a lot more complex, but only in very specific use cases does it need to be.

This block of code says that we want to make all `<a>` elements aquamarine and give all `<h1>` elements a `font-size` of 200 pixels but only if the screen size is under 900 pixels wide.

Let's break down the syntax piece by piece:



```
@media screen and (max-width: 900px) }
  a {
    color: aquamarine;
  }

  h1 {
    font-size: 200px;
  }
}
```

- **At-rule**: A special CSS statement that instructs the style sheet to behave a certain way or apply certain styles when a condition is met. The most popular is `@media (rule)`, which applies styles to the page when a specific style value (call a "rule") is applied on the device. For more details, see the [MDN web docs on CSS at-rule syntax (https://developer.mozilla.org/en-US/docs/Web/CSS/At-rule)](https://developer.mozilla.org/en-US/docs/Web/CSS/At-rule).

- **Media type**: There are three types of media we can apply our CSS to: screen, print, or speech. Omitting this value in a media query makes it apply to all media types, so it's better to apply it selectively as all three have different needs:

- **Screen**: Applies the rule only to digital screens and devices.
- **Speech**: Applies to how screen readers can interpret element's styles. This can even be used to change the voice a screen reader uses, but it is a good practice to keep the default screen reader voice as site visitors using it will be acclimated to its speech patterns.
- **Print**: Applies the rule only when the page is printed or displayed in print preview.

- **Logical operator**: A term that can be used to create more complex media queries by combining conditions. In the example above, setting `screen` and `(max-width: 900px)` means that this media query should apply to screen media and only when the width of that screen is under 900 pixels. The `and` operator is the most prevalent, but there are others (such as `not` and `only`) that are used in more specific cases. To read more, see the **MDN web docs on logical operators (https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries#Logical_operators)**.

- **Media features**: We can also use style characteristics of the browser window to set conditions for media queries. For example, we can have a media query be applied when the device is a `screen` and the `max-width` of the page is 900px. This means if the page is being viewed on a digital screen and the screen is under 900px, the media query will be applied. If the content is being viewed on a screen 901px or above, the media query will be ignored.

NERD NOTE

Ever wonder why an article that's printed off the web doesn't include the ads and other unnecessary content that surround the article on the webpage? It's because that site uses media queries and the `print` media type to hide those elements when printing!

Notice how all the style rules we want to apply in this media query are contained in the media query's brackets `{ }`? It's almost like we're creating a style sheet within the style sheet, and that second style sheet will only come into play when the media query's conditions are met. That is actually exactly what's happening, and with proper organization we can use these to have our site look different at any screen size.

Applying media queries can be a slippery slope if you don't think critically about what you need to do and what screen sizes you should consider. Obviously, no one is asking us to make the page look different at every possible pixel width; that would be insane! Instead what we will do is take three popular device sizes and use them as our `max-width` values. These are known as **breakpoints** because they're the point where the CSS changes based on the screen size.

There is no standard for mobile device or tablet sizes. To accommodate this, we'll use use three device sizes that are commonly used as ballpark values for mobile phones, tablets, and smaller browser screens, respectively: `575px`, `768px`, and `980px`.

DEEP DIVE

Let's get these three media queries set up in our `style.css` file with some dummy styles to test if they work. It's important that they go in this order *after* all of the other styles in the style sheet:

```
/* MEDIA QUERY FOR SMALLER DESKTOP SCREENS AND SMALLER */
@media screen and (max-width: 980px) {
  header h1 a {
    /* this will be applied on any screen smaller than 980px */
    color: tomato;
  }
}

/* MEDIA QUERY FOR TABLETS AND SMALLER */
@media screen and (max-width: 768px) {
  header h1 {
    /* this will be applied on any screen between 768px and 575px */
    font-size: 80px;
  }
}

/* MEDIA QUERY FOR MOBILE PHONES AND SMALLER */
@media screen and (max-width: 575px) {
  header h1 {
    /* this will be applied on any screen smaller than 575px */
    font-size: 100px;
  }
}
```

With these in place, save the file and refresh the browser window, then resize the screen. Even better, if you open Chrome's DevTools and use the device simulator, you can jump right to the sizes you care about!

As you adjust the screen size and see the different styles being applied to the header's `<h1>` element, note the following:

- It has a color of `tomato` at anything below 980px. This is because when it's set to `max-width: 980px`, this style will be applied to anything below it unless another media query is written to specifically remove it.

- The `font-size` is being read from the `header h1` style rule we've had all along, until we hit 768 pixels, which is the standard width for a tablet like an iPad. At that point, it uses a new value of 80px. This value is large, but it's just to show how these new values take hold at a certain width.

- The `font-size` is overridden again when we hit the `575px` breakpoint.

SHOW PRO TIP

Order Matters

The order in which we write our media queries matters, not only in terms of how they're placed in relation each other but in how they're placed in `style.css` as a whole.

In our case, we put the biggest screen size first and worked our way down, using `max-width` as the way of determining if a media query should be applied or not. What would happen if we were to reverse that order by putting `575px` first and `980px` last? The result will show that the `font-size` will stay at 80px even when the screen gets smaller than 575px wide.

The reason this occurs is because of the "cascade" in CSS. The browser reads the style sheet from top to bottom and sees the media query for 575px, but then it continues and sees the media query for 768px, which will then take precedence over any conflicting style rules for 575px because it came later in the style sheet. If we were to change the value to `min-width`, however, it would work; but since we're using `max-width`, we have to work from large sizes to small instead of small to large.

Now consider what would happen if we placed our media queries at the top of `style.css` and not at the bottom. In this scenario, all of the CSS rules we applied in the media queries that override desktop styles would not be applied because the desktop styles were defined later than the media query styles. Again, this is because of the browser reads a style sheet from top to bottom.

IMPORTANT

> Writing media queries that handle screen sizes going from large to small is a good practice when you're working on an existing site that

> looks good on a desktop browser screen. When starting a new site from scratch, however, it is best practice to take a mobile-first approach and use `min-width` values instead of `max-width` ones.
>
> A mobile-first approach means we think about how the site looks on mobile before anything else. It is inherently more difficult to make a site look good on a small screen than on a large one. So think of how it looks on mobile first and build out those CSS styles. Then scale it up for tablets and computer screens.

Now that we know that our media queries work in the right order, we can focus on updating the Run Buddy site to look good on all devices. Before we move on, go ahead and remove the CSS style rules from the media queries, but keep the media queries so they look like this:

```
@media screen and (max-width: 980px) {

}
```

Once that's complete, make sure to save your work to the `media-queries` branch!