1.7.7 Add a Second Style Sheet

Let's start by using the command line to create another CSS file.

Try doing this on your own. Here's an overview of what you need to do:

1. We want to add this file to the `css` directory, which is inside `assets`. Use the `cd` command to change the working directory to `assets/styles`.

2. Once inside the `css` directory, use the `touch` command to create a file named `secondary-styles.css`.

3. Add that style sheet to `privacy-policy.html` by adding a second `<link>` tag below the current one and set the `href` value to `./assets/css/secondary-styles.css`.

HIDE HINT

Use `pwd` and `ls` to navigate your computer's file system.

The result should be that the privacy policy page's `<head>` tag contains the following (in this order):

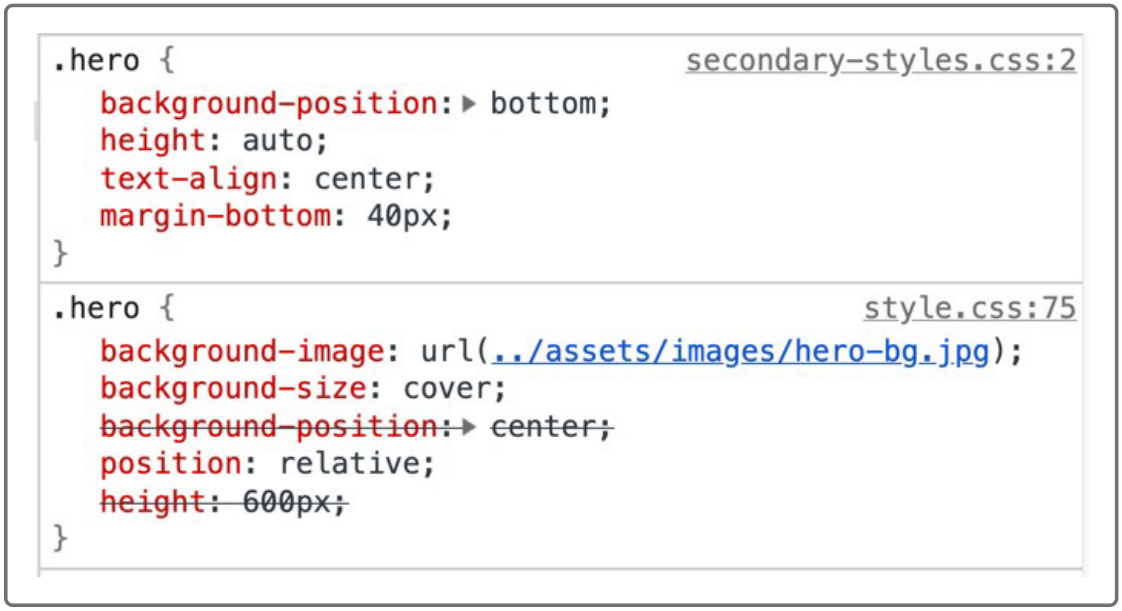
```
<link rel="stylesheet" href="./assets/css/style.css" />

<link rel="stylesheet" href="./assets/css/secondary-styles.css" />
```

Now that your files are in place, add the following style definitions to `secondary-styles.css` to take care of the hero section:

```
.hero {
  background-position: bottom;
  height: auto;
  text-align: center;
  margin-bottom: 40px;
}
```

That should have fixed the spacing issues in that section. We overrode the `background-position` and `height` properties and added `text-align` and `margin-bottom`. Did you notice that we didn't even list the `background-image` and `background-size` properties?

To get a better idea of what's happening, use Chrome's DevTools to see something like this image shows us:



As you can see, there are two sets of styles being applied to the `hero` class. One is in `secondary-styles.css` at line 2 (in this screenshot), and the other is in `style.css` at line 75. To explain how the browser chose which styles to apply and which to discard, just look at how the `secondary-styles` one is listed on top of the other one, as if it's taking precedence. That's because it is.

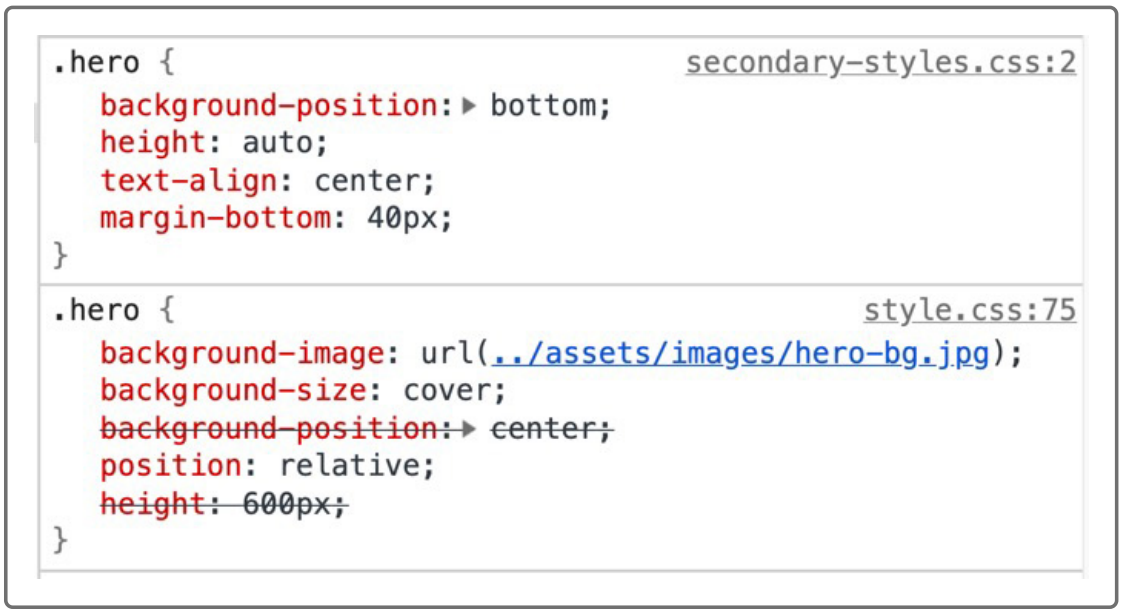
REWIND

This is an example of the CSS "cascade" in effect. Think back to Lesson 2 when CSS was introduced. There are the three factors in CSS that determine which styles get applied: importance, specificity, and source order. This is an example of source order affecting which style definitions win.

Try switching the order of the `<link>` elements in the `<head>` element to see how it affects source order.

To learn more, see the **MDN web docs on CSS cascade and inheritance (https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Cascade_and_inheritance)**.

shows:



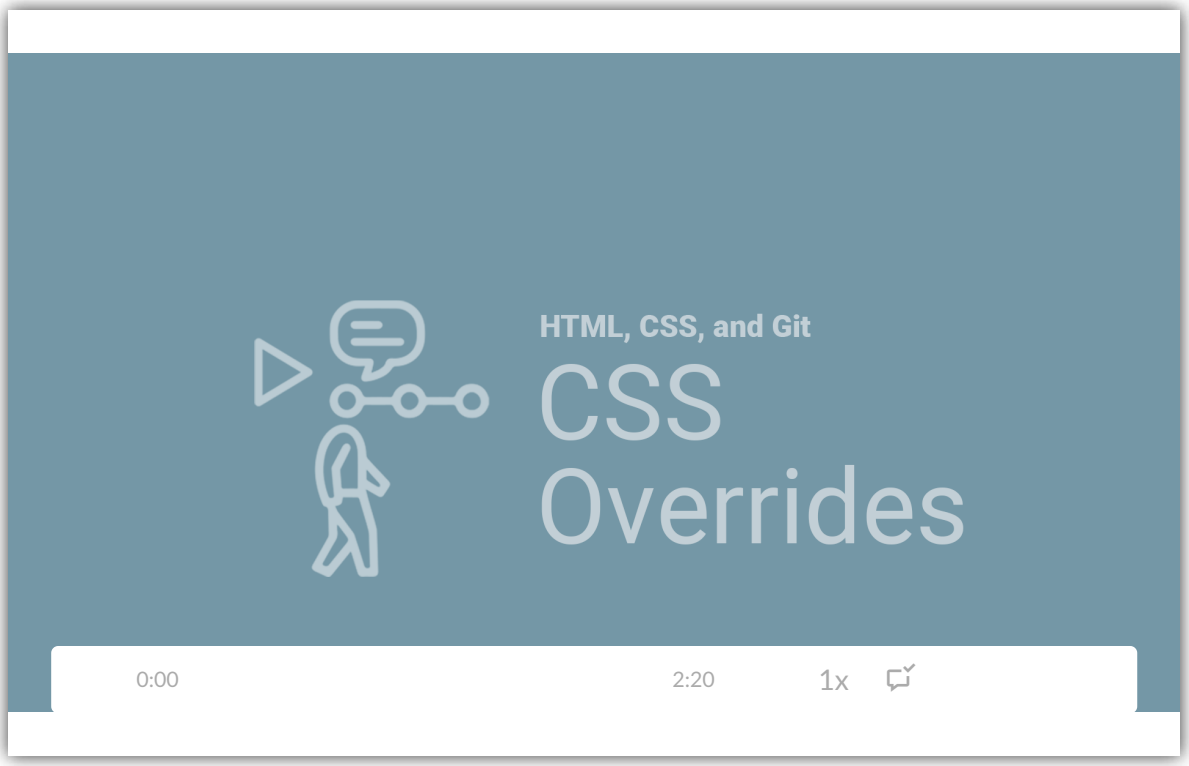
The hero section in `privacy-policy.html` sees two sets of CSS rules from the two style sheets and combines them, then prioritizes the values that come later in conflicting declarations. The resulting CSS rule for the `.hero` class is this in the browser:

```
.hero {
  background-image: url(../assets/images/hero-bg.jpg);
  background-size: cover;
  background-position: bottom;
  position: relative;
  height: auto;
  text-align: center;
  margin-bottom: 40px;
}
```

Now that all of the style overriding is done, we can turn our attention to creating the new style definitions for this page.

In `privacy-policy.html`, the `<link>` tag for `secondary-styles.css` comes after `style.css`. The browser reads these tags in order of appearance, so everything from `style.css` is applied first. Then it sees the styles defined in `secondary-styles.css` and applies those. Any conflicting property definitions are overridden by the declarations that are read last. This is what **source order** means: the declarations that come last prevail.

A useful feature of CSS is that it overrides at the declaration level, not the rule level. Any property declarations in the overridden rule will remain intact if the overriding rule does not redeclare them. We didn't define new values for `background-image`, `background-size`, or `position` because we want to use the same styles, so they get to carry over from the other style definition. To see declaration level overrides in action, check out the following video:



Again, we can see these CSS rules and overrides working together to create this updated hero section in Chrome's DevTools, as this image