2.2.8 Center Section Titles Using Flexbox

Flexboxes don't need to contain multiple elements to be considered useful. In this step, we're going to learn how to use flexbox to create a one-column layout, meaning it will only have one child. This isn't necessary, but adjusting these to be flexboxes will enable us to handle `justify` and `align` properties more easily across multiple screen sizes.

We'll start by editing the section titles in `index.html` file. We'll wrap each `<h2>` element in a `<div>` that we'll turn into a flexbox.

To do this, find every `<h2>` element with a class of `section-title` and make it look like this:

```
<!-- Wrap every h2 with a class of "section-title" in this div! -->
<div class="flex-row">
  <h2 class="section-title primary-border">
    What We Do
  </h2>
</div>
```

SHOW HINT

Next wrap the `<p>` element in the "What We Do" `section` element so it looks like this:

```
<div class="flex-row">
  <p>
    butcher selfies chambray shabby chic gentrify readymade yr Echo Pa
  </p>
</div>
```

We added a class of `flex-row` to each wrapping `<div>` element. We chose the class name `flex-row` because it represents a CSS rule that sets up a flexbox row. If we were to make a flexbox column, we could say `flex-column` instead. Remember that naming CSS classes can be as specific or generic as you want, but think about the bigger picture and how they might be reused throughout your page designs—just like you did with the `section-title` and `primary-border` classes.

Now that the HTML is in place, we can turn our attention to the CSS. Just like we did earlier, we'll start by removing styles:

1. From the `.section-title` CSS rule, remove the declarations for `display`, `padding`, and `margin-bottom`.

2. Remove the entire CSS rule for `.intro`.

3. From the `.steps` CSS rule, remove the `text-align` declaration.

4. From the `.trainers` CSS rule, remove the `text-align` declaration.

5. From the `.contact` CSS rule, remove the `text-align` declaration.

Now let's add a style rule for anything with a class of `flex-row`. At the bottom of `style.css`, where we have the utility classes for `text-left` and `text-right`, create a CSS rule that looks like this:

```
.flex-row {
  display: flex;
}
```

PAUSE

If we're creating a flexbox row, why didn't we need to use the `flex-direction: row;` declaration?

Show Answer

We just created a CSS rule that can be applied anytime we need a flexbox row container. It is often more efficient to create classes that can be used universally throughout a project rather than applying the same CSS property declaration to multiple items.

Now that the flex row has been created, add the following declarations to the `.section-title` CSS rule:

- `padding-bottom: 20px;`
- `text-align: center;`
- `margin: 0 auto 35px auto;`
- `width: 50%;`

PAUSE

How much margin does each side get if the declaration reads `margin: 0 auto 35px auto`?

Show Answer

Lastly, we need to center the text in the "What We Do" paragraph, so we'll finish by adding a `text-align: center` to the `.intro p` CSS rule. Go ahead and do that now.

Even though we can't see any immediate tangible payoff from the work we did in this section, these changes made our code more flexible and prepared it to become mobile-friendly. The next part of the lesson will involve a fairly large restructuring of the "What You Do" section of the site. As always, now is a good time to make sure your code is saved using Git.