2.2.11 Make a Three-Column Reach Out

Only one more flexbox to go, and this one is a breeze! If you can recall, we left the Reach Out section in disarray in the last lesson when we added a contact form and didn't style it:

This is a common approach to take when you know you'll be reworking the CSS anyway—why bother doing the work twice?

It's time to fix this. We'll change the layout to this:

One last time, think about this layout and create a visual wireframe of what it would look like as a flexbox parent/children relationship.

The first obvious change we need to make is the actual order in which these elements appear in the HTML document, so return to `index.html` and adjust the three elements below the section title's parent `<div>` to be in this order:

1. The `<div>` with the Run Buddy address and contact information
2. The `<div>` with a class of `contact-form`
3. The `<iframe>` for the Google Map embed

Then wrap the elements with a `<div>` that has a class of `contact-info` so you can use that to create a flexbox container.

The resulting HTML should look like this:

```
<div class="contact-info">
  <div>
    <h3>Run Buddy</h3>
    <p>
      Any questions or concerns before signing up?
      <br />
      Let us know and we'll be happy to talk to you!
    </p>

    <address>
      55 Main Street <br />
      Some Town, Ca <br />
      12345 <br />
      P: <a href="tel:555.786.2839">555.RUN.BUDZ (555.786.2839)</a><br
      E: <a href="mailto://info@runbuddy.io">info@runbuddy.io</a>
    </address>

  </div>

  <div class="contact-form">
    <h3>Contact Us</h3>
    <form>
      <label for="contact-name">Your Name</label>
      <input type="text" id="contact-name" placeholder="Your Name" />

      <label for="contact-message">Message</label>
      <textarea id="contact-message" placeholder="Message"></textarea>

      <button type="submit">Submit</button>
    </form>
  </div>
```

```
  <iframe
    src="https://www.google.com/maps/embed?pb=!1m14!1m12!1m3!1d12182.3
    frameborder="0" style="border:0" allowfullscreen></iframe>

</div>
```

Moving into `style.css`, let's start adjusting styles by removing some current ones that will get in our way:

- Find the CSS rule for `.contact-info iframe`:
  - Remove the `width`, but leave the `height`
- Find the CSS rule for `.contact-info div`:
  - Remove every property except `color`

Now we can add our flexbox. Create a CSS rule for `.contact-info` that has these properties:

- `display: flex;`
- `justify-content: space-between;`
- `flex-wrap: wrap;`

The last thing we need to do for the flexbox layout is to have all three child elements share the space as equally as possible. If you recall, we did that just a little while ago in the trainer section by giving them all a `flex: 1` style declaration. But these three elements don't share a common element name or even class, so how do we select all three of them at once to make this easier?

We could go back to the HTML and give them all a class, or we could skip that and select all of them with a familiar selector used in a different context:

```
.contact-info > * {
  flex: 1;
  margin: 15px;
}
```

Do you remember what the `*` selector did? Also, what's with this new `>` syntax?

SHOW HINT

Here, the `*` selector is saying "select all elements that are direct children of anything with a class of `contact-info`." The `>` symbol means it's looking specifically for direct children of the element listed first. In this case, it's selecting all children of the element with `contact-info` as its class.

Remember, however, that CSS reads right to left. This means that if the first selector it reads is `*`, it will try and find *every* element on the page first, then look at the `.contact-info`, and then filter out the elements that aren't a direct child of it. This wouldn't be a good approach in a big website, but it won't affect performance for a site this size.

In this case, it will apply the above styles to the three flexbox child elements in our `contact-info` flexbox container, so we are now all set with this section's flexbox layout.

We'll close out on this section by making some style updates:

- First, find the CSS rule for `.contact-info p, .contact-info address`:
    - Change the `font-size` to 16px

Now for the contact form elements:

- Create a CSS rule called `.contact-form input, .contact-form textarea` with:
  - `border: 1px solid #024e76`
  - `display: block`
  - `padding: 7px 15px`
  - `font-size: 16px`
  - `color: #024e76`
  - `width: 100%`
  - `margin-bottom: 15px`
  - `margin-top: 5px`
- Create a CSS rule called `.contact-form button` with:
  - `width: 100%`
  - `border: none`
  - `background: #fce138`
  - `color: #024e76`
  - `text-align: center`
  - `padding: 15px 0`
  - `font-size: 16px`

Save your changes and refresh the page in the browser. The Reach Out section should now resemble the mock-up.

Learning this much about layouts in CSS is challenging, but it is very rewarding when you realize how much control you can exercise over your HTML elements. There is nothing more important for a CSS developer than the ability to visualize a page as a series of containers, and it is a very exciting time for CSS because of tools like flexbox are creating highly customizable layouts that look similar to those in magazines and other print media.

We've finished this feature request and are ready to take the code from this branch and merge it into our `develop` branch. Then we'll be ready to tackle the next feature request on our list!