Balancing Text and Images with Flexbox

One of the cool things we can do with Flexbox is to balance the text and images as if they were in a two-cell table. It should be possible to do so with images but instead we'll simulate two cells with the code below. I normally don't do this but in this case I will use CSS to populate the first `div` element with the appropriate image.

```
<div class="column">
  <figure class="flex">
    <div></div>
    <div>
      <h3>Chrome Canary</h3>

      <p>I install both Canary and Release versions to
      make sure that the code I'm working on works in
      my target browsers</p>
      <p>I install both Canary and Release versions to
      make sure that the code I'm working on works in
      my target browsers</p>
      <p>I install both Canary and Release versions to
      make sure that the code I'm working on works in
      my target browsers</p>

    </div>
  </figure>
</div>
```

The CSS is where all the magic happens. I've broken in into different sections. In the first section we define our layout. In particular:

- We define the element with class `flex` to have display flex
- In odd children we change the default mode for flex to display elements in reverse order. This will display the image on the right side and the text on the left
- The first `div` child uses background attributes to manipulate the image.

This is not really doable with images inserted using the `img` tag
- The last `div` child will take twice the space of the first one

```
.flex {
    margin: 0;
    display: flex;
    border: 5px solid #333;
    margin-bottom: 2rem;
}

.flex:nth-child(odd) {
    flex-direction: row-reverse;
}

.flex div:first-child {
    flex: 1;
    background-size: cover;
    background-position: center;
}

.flex div:last-child {
    margin: 2rem;
    flex: 2;
}
```

In the second block of CSS we do some formatting for the text content of each section. The last paragraph, `.flex p:last-of-type` has an additional rule to eliminate the bottom margin; this makes sure the empty bottom margin of that element doesn't add to the total height of the text.

```
.flex h3 {
    font-size: 1.5rem;
    margin-top: 0;
    font-weight: 400;
}
.flex p {
    font-size: 1rem;
    line-height: 1.4;
```

```
    font-weight: 400;
}

.flex p:last-of-type {
    margin-bottom: 0;
}
```

This section adds the images as background images to the empty first div of each figure. I don't particularly like using background images because they make it harder to share and to work with outside of CSS.

For this kind of project working with images using the `img` tag on the page doesn't produce the same effect. Using `background-size: cover` is different than making the image fluid using percentages for width.

For each of the children of `.flex` we add a background image to the first `div` children. It can be the same image or a different one like we've done in this case.

```
.flex:nth-child(1) div:first-child {
    background-image: url("images/chrome-canary_128x128.png");
}

.flex:nth-child(2) div:first-child {
    background-image: url("images/chrome_128x128.png");
}

.flex:nth-child(3) div:first-child {
    background-image: url("images/firefox-developer-edition_128x128.png")
}

.flex:nth-child(4) div:first-child {
    background-image: url("images/firefox_128x128.png");
}
```

The final section of our CSS is a media query to accommodate smaller form factors and avoid the image looking ugly on iPhones and other smaller form factors. We accomplish this by changing the layout from horizontal to vertical (`flex-direction` changes to `columns`).

```
@media screen and (max-width: 600px) {
    .flex { flex-direction: column; }
    .flex div:first-child { min-height: 200px; }

    .flex:nth-child(odd) {
        flex-direction: column;
    }
}
```

The idea is to create a consistent layout for images and text. We can use this as the index page for a magazine or the starting point of additional experiments using Flexbox beyond gallery displays.

> Hat tip to Dudley Storey for the original idea.