1.2.9 Design the Footer

Now let's make the `<footer>` look like this image:

As you kick off this section, think back to the styles you applied to the `<header>` and notice how many of these concepts are repeated.

Start by adding the following CSS to the `<footer>`:

```
footer {
  background: #fce138;
  width: 100%;
  padding: 40px 35px;
}
```

Did you notice that we used the same property declarations for the `<footer>` as we did in the `<header>` with just a few value tweaks to give it a different background color and padding values?

Now let's go ahead and tackle the rest of the content in the `<footer>` step by step:

1. Apply style to the left `<h2>` element.

```
footer h2 {
  display: inline;
  color: #024e76;
  font-size: 30px;
  margin: 0;
}
```

2. Apply style to the right `<div>` container that holds the rest of the content.

```
footer div {
  float: right;
  line-height: 1.5;
  text-align: right;
}
```

3. Apply style to the `<a>` element.

```
footer a {
  color: #024e76;
}
```

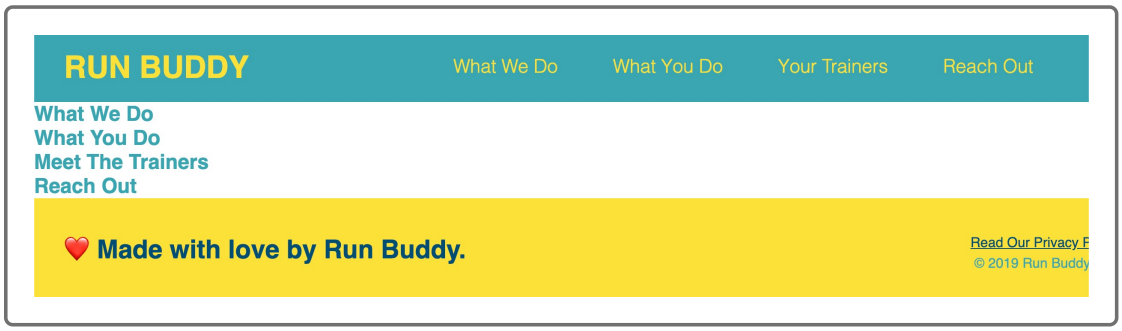
That wasn't so bad, was it? Did you notice that we repeated the same layout that was in the header by making the `<h2>` an inline element and the `<div>` float to the right?

And only two new properties were introduced:

- `line-height`: This assigns how much vertical space should be between lines of text content. The value (1.5) means we want the spacing to be 1.5 times the size of the font. This value varies depending on what font we're using, but 1.5 is a good baseline. The idea behind this is we don't want our lines of text too close or too far apart. This lets us finesse the spacing and make it more readable.

- `text-align`: This lets us align our text to the left, right, center, or justified. By default, it is left-aligned.

Check out the newly updated `<footer>` by saving your files and refreshing the page in the browser. You'll notice it doesn't look quite right, however, as this image shows:



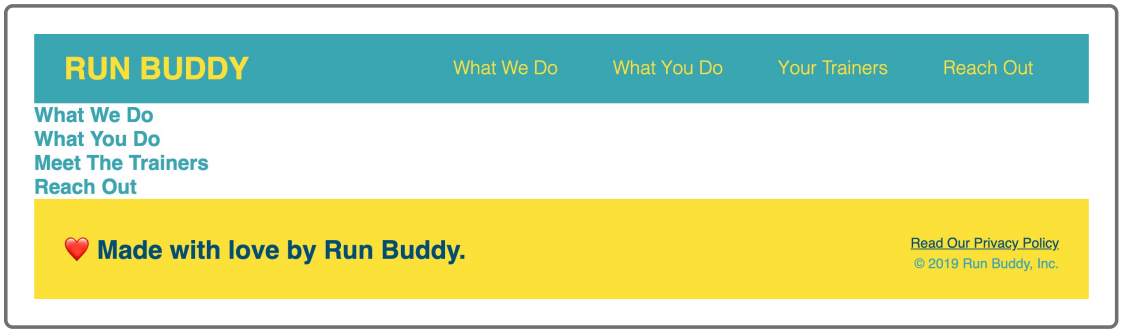
How could this be? Well, remember that whole "box model" discussion from earlier? Our issue has something to do with that. See, we told our `footer` CSS rule declaration to have both a `width` of 100% and `padding` of 35px on both the left and right sides. Having a width of 100% means it'll be the full width of it's parent (in this case, it's the `<body>` element), but then we add on a total of 70px to the left and right with padding and it makes our `<footer>` element wider than the actual screen!

Luckily, there's a fix that will instruct the browser to ignore padding in the overall width. Update the `*` selector CSS at the top of the file to look like this code:

```
* {
  margin: 0;
```

```
  padding: 0;
  box-sizing: border-box;
}
```

Save your CSS file and refresh in the browser, it should look like this image:



How did that one line of code fix it? Let's find out.

Box-Sizing: Calculating the Width and Height of Elements

The `box-sizing` property determines how to calculate the `width` and `height` of each element. There are two possible values for the `box-sizing` property: `content-size` and `border-box`.

- `content-size` is the default value and calculates the height and width of the element by only counting the `content` box of the CSS Box Model. This means that the `border` and `padding` must be calculated separately and added to the width and height to determine the size of the element.

- `border-box` calculates the height and width of the element by including the `border` and `padding` additions to the `content` box.

DEEP DIVE

DEEP DIVE

To learn more, check out the **MDN web docs on the box-sizing property (https://developer.mozilla.org/en-US/docs/Web/CSS/box-sizing)**.

Great work! You've officially completed all of the styles for the header and footer!

We covered a lot, so let's do a knowledge check:

Final score: 100%

Retake