2.2.5 Flex the Navigation

Next, we'll add flexbox to the navigation elements. When you resize the browser window, the `<nav>` element breaks onto the next line just fine but all the links in the `<nav>` break in weird places and start stacking. So let's make the `<nav>` element's `<ul>` element more responsive by converting it from an inline list to flexbox.

IMPORTANT

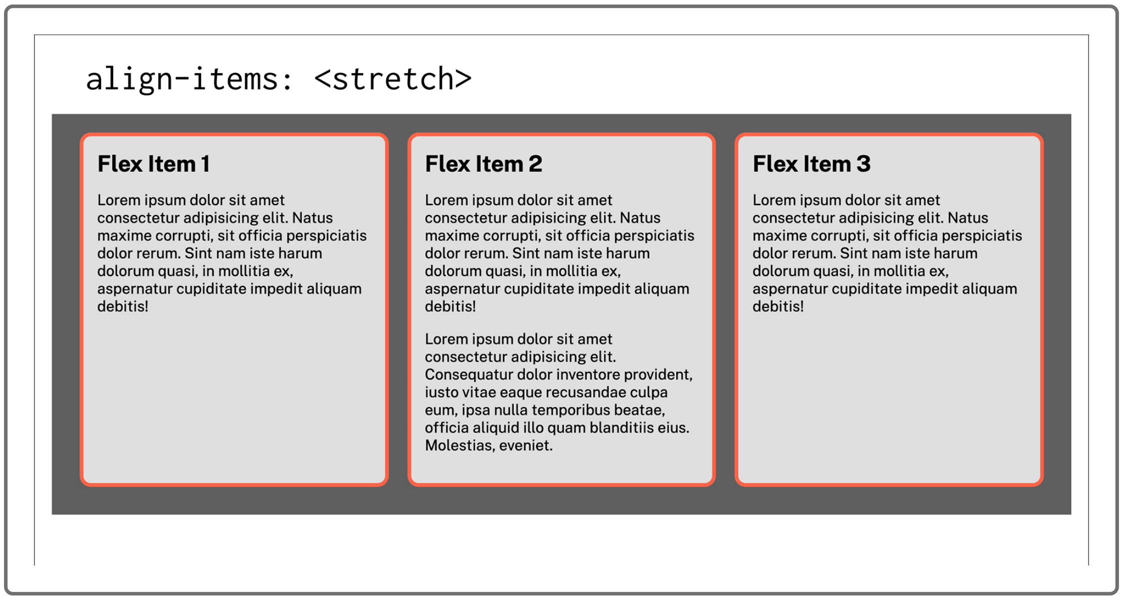
> You can create as many flexbox elements on a page as you need to. Just be very deliberate about it—it is not a cure-all for all layouts.

We need to remove some CSS before applying the flexbox styles:

- Completely remove the CSS rule for `header nav ul li`.
- Add a new CSS rule for `header nav ul` that looks like this:

```
header nav ul {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
  align-items: center;
  list-style: none;
}
```

We've already seen these first three property declarations, but what about those last two? The property `align-items` works like `justify-content` but on the opposite axis. To "justify" information means to position it on the main axis, but to "align" it means to position it along the cross axis. In this case, we're vertically centering our content in the `<nav>` element. This might sound like a simple task, but it was only when this property was invented that it became possible to achieve this type of vertical alignment without using JavaScript.


align-items: <stretch>
Flex Item 1
Lorem ipsum dolor sit amet consectetur adipisicing elit. Natus maxime corrupti, sit officia perspiciatis dolor rerum. Sint nam iste harum dolorum quasi, in mollitia ex, aspernatur cupiditate impedit aliquam debitis!
Flex Item 2
Lorem ipsum dolor sit amet consectetur adipisicing elit. Natus maxime corrupti, sit officia perspiciatis dolor rerum. Sint nam iste harum dolorum quasi, in mollitia ex, aspernatur cupiditate impedit aliquam debitis!
Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequatur dolor inventore provident, iusto vitae eaque recusandae culpa eum, ipsa nulla temporibus beatae, officia aliquid illo quam blanditiis eius. Molestias, eveniet.
Flex Item 3
Lorem ipsum dolor sit amet consectetur adipisicing elit. Natus maxime corrupti, sit officia perspiciatis dolor rerum. Sint nam iste harum dolorum quasi, in mollitia ex, aspernatur cupiditate impedit aliquam debitis!

IMPORTANT

When using a flexbox as a row, the horizontal (x) axis is known as the **main axis**. This is the direction we can control when laying out the flexbox children. The vertical (y) axis is the **cross axis**. If we were to use the `flex-direction` property and set the value to be `column` instead of `row` (the default value), those axes would be switched and the values used for `justify-content` and `align-items` would be applied differently. This is because `justify` always follows along the main axis and `align` always follows along the cross axis.

The other property we added, `list-style`, sets how a list item is styled (bullet points, dashes, etc.) and positioned (indented or outdented). We gave it a value of `none` to hide the bullet points. We did this because we're changing some of the styles for the `<header>`, which would result in the bullet points becoming visible.

DEEP DIVE ▼

Let's finish updating the navigation by making a few adjustments to the `<nav>` element's link styles so it looks like this:

```
header nav ul li a {
  margin: 0 30px;
  font-weight: lighter;
  font-size: 1.55vw;
}
```

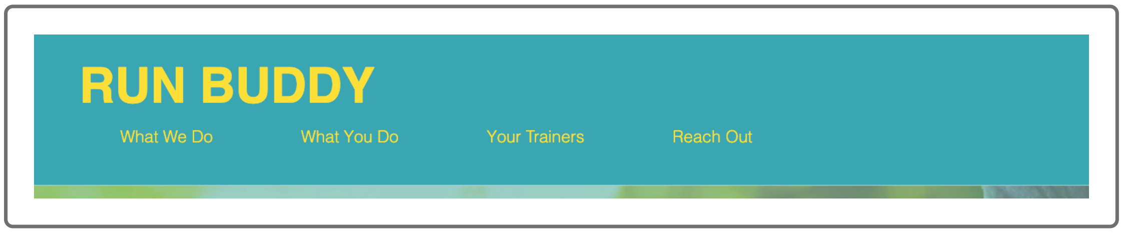
We're using the same properties as before but with a different value for `font-size`. The value we provided is using a newer unit of measurement called `vw`, which is short for **viewport width**.

REWIND

Remember that the **viewport** is the actual browser window size.

By setting the value to `1.55vw`, we're telling the browser that the font's size should be roughly 1.55 percent of the window's overall width. This means that if the browser grows or shrinks, that size will change relative to the new overall width. This unit of measurement is one of a few new methods we can use for flexible length values in CSS, but it can be tricky to nail down compared to absolute units of measurement like pixels.

We now have a `<nav>` element where the links grow and shrink with the screen size, but as you can see in the following image, they get a little too small when the screen is smaller. We'll circle back to fix this in a later lesson.



You're done with the entire `<header>` for now, so it's a good time to save your work in the `feature/flexbox` branch using Git.

Before we move on to the footer, if you'd liked to reinforce what you just learned, check out a game called **Flexbox Froggy (https://flexboxfroggy.com/)** .

DEEP DIVE ▼