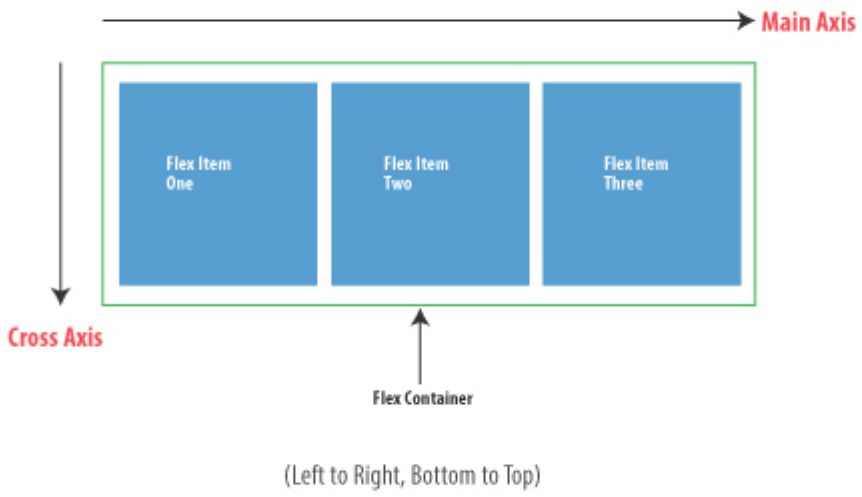
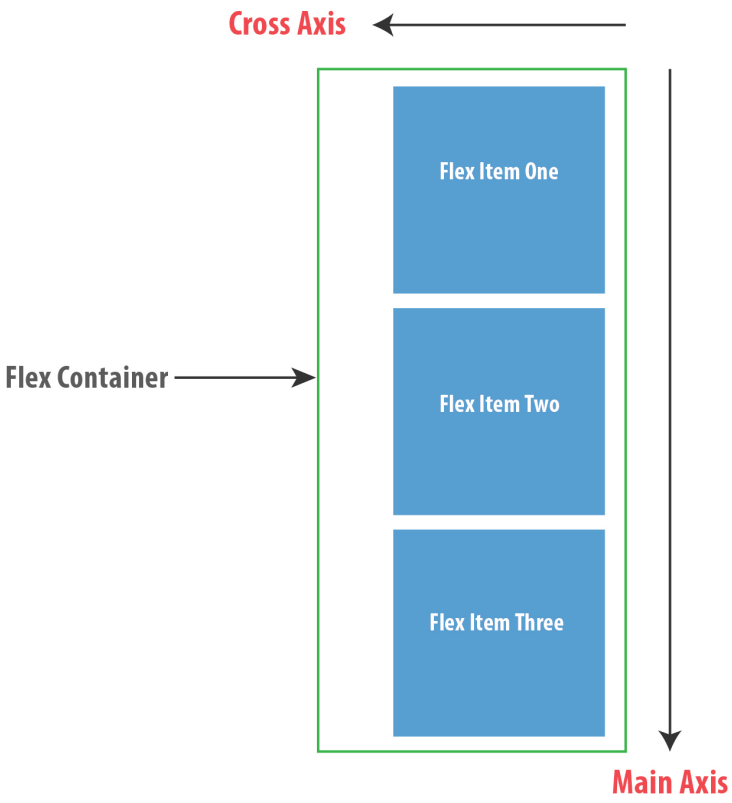
What happens when you switch flex-direction?

When starting off with learning the flexbox model, this part was the most confusing. I bet a lot of newcomers to the flex world find it that way too.

You remember when I talked about the default main and cross axis being in the “left-to-right” and “top-to-bottom” directions?



Well, you can change that too. Which is exactly what happens when you use `flex-direction: column` as described in an earlier section (remember the default value is `flex-direction: row`). When you do this, the main and cross axis are changed as seen below:



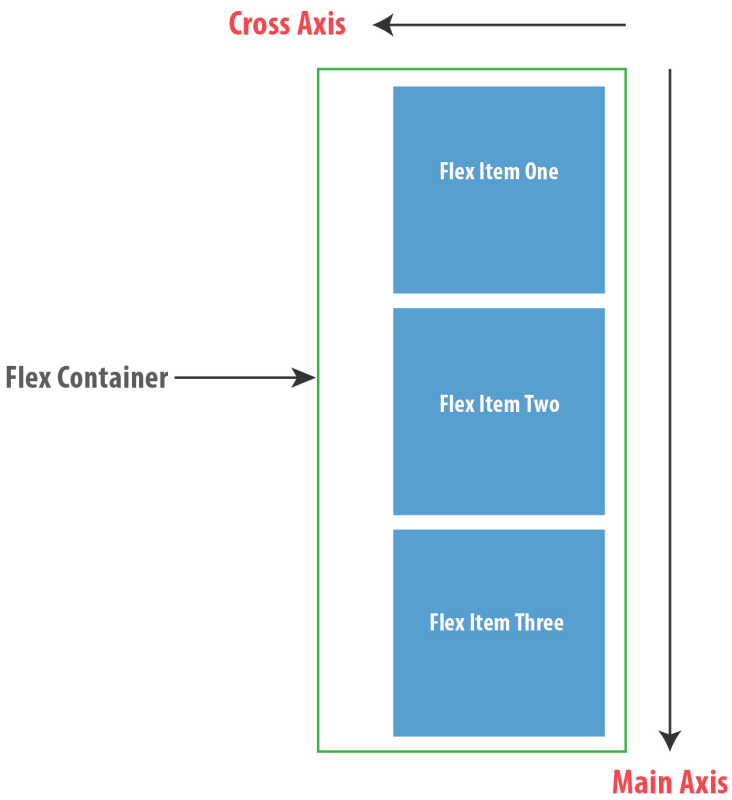
Top to Bottom, Right to Left)

If you've ever written any text in the English language, then you already know the language is written from left-to-right and top-to-bottom. That's equally the direction taken for the default main and cross axis of the flexbox too.

However, on switching the flex direction to `column`, it no longer follows the "English Language" pattern but Japanese. Oh yes, Japanese!

If you've written any text in the Japanese language, then this will be familiar. (for the records, I've never written any texts in Japanese). Japanese text is written from top-to-bottom and right-to-left. Think about that for a second. Not so weird, huh?

That explains why this can be a bit confusing for English writers.



Top to Bottom, Right to Left)

Let me walk you through an example. The standard unordered list with 3 list items, except this time I'll change the flex-direction.

```
        <ul>
                <li></li>
                <li></li>
                <li></li>
        </ul>
```

```
ul {
        display: flex;
        flex-direction: column;
}
```

Here's the look before the change in direction:

HTML CSS Output

```
ul {
        display: flex;
        border: 1px solid red;
        padding: 0;
        list-style: none;
        background-color: #e8e8e9;
    }

li {
        flex: 0 0 auto;
        background-color: #8cacea;
        margin: 8px;
        color: #fff;
        padding: 4px;
    }
```

output

List One List Two List Three

and after:

HTML CSS Output

```
ul {
        display: flex;
    flex-direction: column;
        border: 1px solid red;
        padding: 0;
        list-style: none;
        background-color: #e8e8e9;
    }

li {
        flex: 0 0 auto;
        background-color: #8cacea;
        margin: 8px;
        color: #fff;
        padding: 4px;
    }
```

output

List One

List Two

List Three

So what happened? The ‘text’ is now written in japanese style - from top-to-down (main-axis).

There’s something you may find funny, I’d love to point out.

You see the width of the items fill up the space, right?

If you were to change that before now, you’d just deal with the `flex-basis` and(or) `flex-grow` properties.

Let’s see how those affect our new layout.

```
li {
        flex-basis: 100px;
}
```

...and here’s what we’ve got:

HTML CSS Output

```
ul {
        display: flex;
    flex-direction: column;
        border: 1px solid red;
        padding: 0;
        list-style: none;
        background-color: #e8e8e9;
    }

li {
        flex: 0 0 auto;
    flex-basis: 100px;
      background-color: #8cacea;
        margin: 8px;
        color: #fff;
        padding: 4px;
    }
```

output

List One

List Two

List Three

Wow - what? The height is affected, but not the width?

I earlier said the flex-basis property defines the initial-width of every flex-item. I was wrong - or better put, I was thinking in “English”. Let’s switch to Japanese for a bit.

It doesn’t always have to be “width”.

Upon switching flex-direction, please note that every property that affected the main-axis now affects the **new** main axis.

A property like `flex-basis` that affected the width of the flex-items along the main-axis now affects the height NOT width. The direction has been switched!

So even if you used the `flex-grow` property, it’d affect the height too!

Essentially, every flex property that operated on the horizontal axis (the then main-axis) now operates on the new main-axis, vertically - It’s just a switch in directions.

Let’s see one more example. I promise you’d get a better understanding after this.

Reduce the width of the flex-items we looked at just before now, and they no longer fill the entire space:

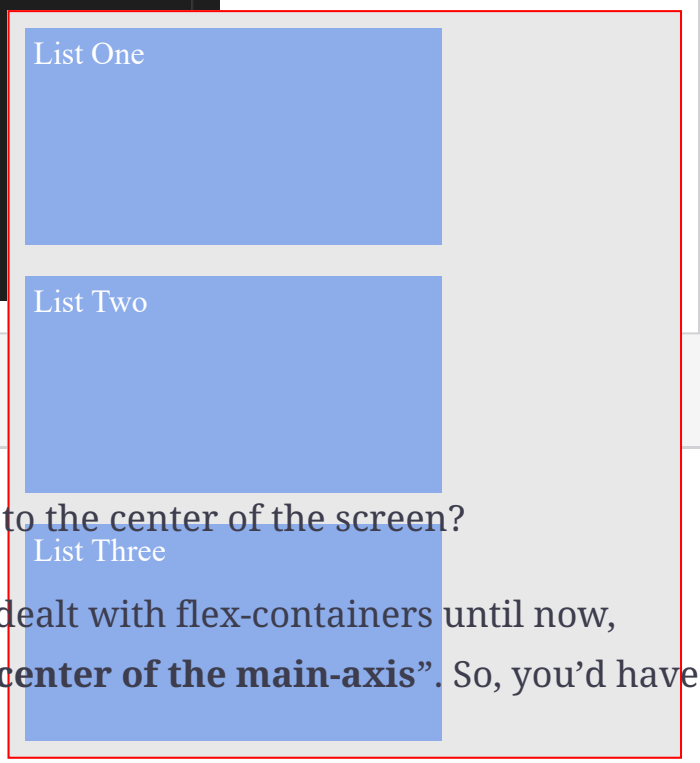
```
li {
        width: 200px;
}
```

HTML CSS Output

```
ul {
        display: flex;
    flex-direction: column;
        border: 1px solid red;
        padding: 0;
        list-style: none;
        background-color: #e8e8e9;
    }

li {
```

```
11          flex: 0 0 auto;
12        flex-basis: 100px;
13        width: 200px;
14          background-color: #8cacea;
15          margin: 8px;
16          color: #fff;
17          padding: 4px;
18        }
```

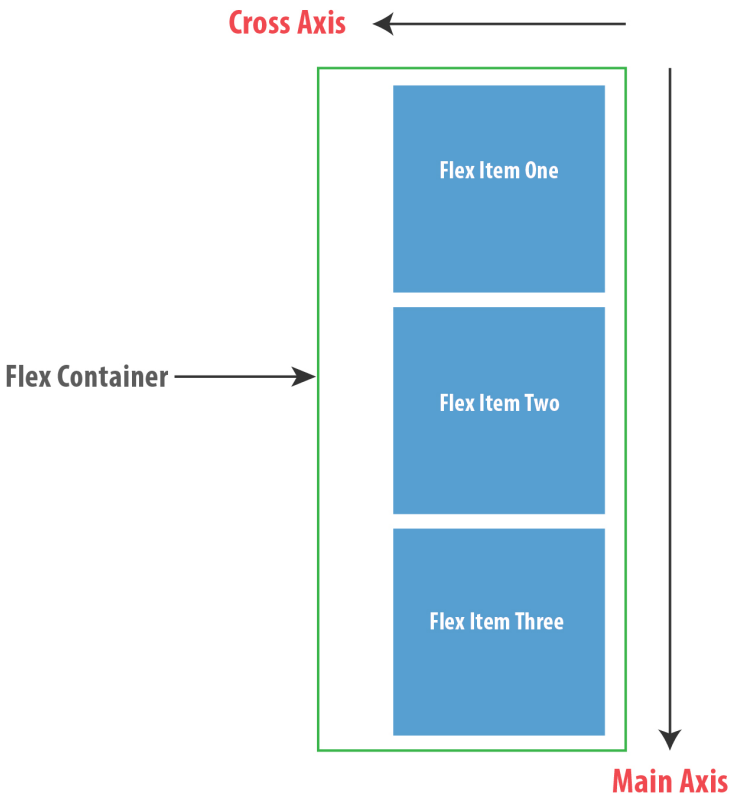


What if you wanted to move the list items to the center of the screen?

In English language, which is how you've dealt with flex-containers until now, that'd mean "**move the flex-items to the center of the main-axis**". So, you'd have used `justify-content: center`

Doing that now does NOT work.

Since the direction's changed, the center is along the cross-axis. Take a look again:



Top to Bottom, Right to Left)

So let's think in Japanese. The main-axis is from top-to-down, you don't need that. The cross-axis is from left-to-right. Sounds like what you need.

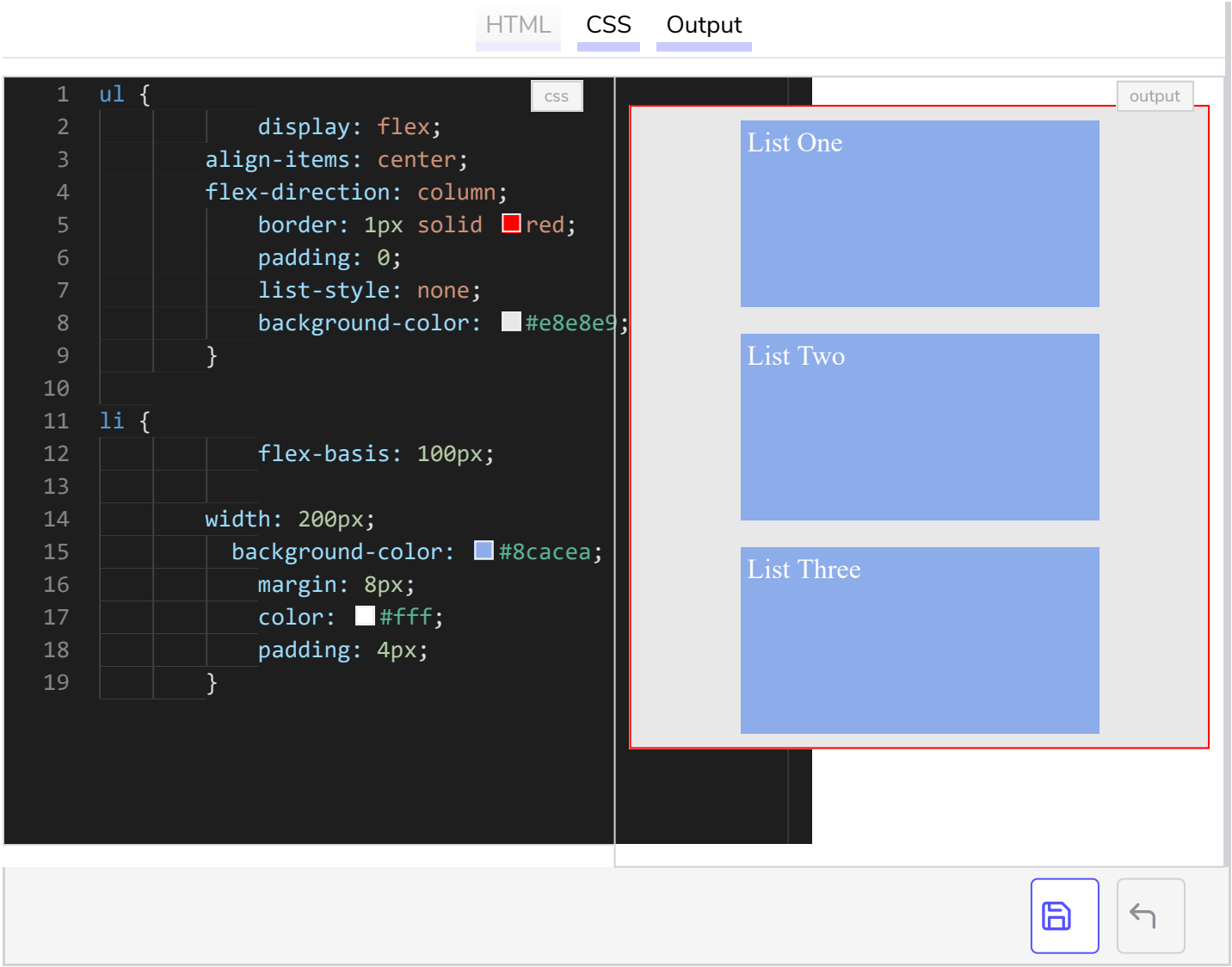
Hence, you need to "__move the flex-items from the start of the cross-axis to the center".

Any flex-container property rings a bell here? Yeah, the `align-items` property .

Remember the `align-items` property deals with alignment on the cross-axis. So to move those to the center, you'd do this:

```
li {
        align-items: center;
}
```

and voila! We've got our centered flex-items.



It can get a bit confusing, I know. Just go over it one more time.

While studying the flexbox model, I noticed a lot of CSS books skipped this part. A bit of thinking in Japanese would go a long way to help and it's worth understanding that these properties work based on the `flex-direction` in place.

I'm sure you learned something new again. I'm having fun explaining this. I hope you are loving this too :-)