Flexbox Crash Course

Truly centering an element is something Flexbox does with ease. A lot of ease. Let's take a quick look at what Flexbox is.

If you know me (from any online community) then you should know Flexbox happens to be my favorite concept in CSS. I find it beautiful.

This lesson will NOT teach you everything about Flexbox. You will, however, get a headstart on what flexbox is, and even get to use it in the sign-up project.

What is Flexbox

Flexbox is got a technical definition. For the sake of this course, I'll skip the technicalities.

You may see flexbox as the layout ninja for CSS.

When you need to deal with the layout in your styles, the CSS Flexbox model is likely to be your best bet.

How does it Work?

Flexbox feeds largely on the parent-child relationship in a given DOM structure.

Flexbox is kicked off by making a **parent** element a **flex container**

Forget the lingo. It is like saying, *Oh, I have found this parent element which I need for my layout. Alright, let me hand over to it some flexbox super powers*

Yeah, that's it.

How Do you Hand Over the Flexbox Super Powers?

Pretty simple.

Assuming the parent element in question is a `div` with a class name of `flexy`, just do this:

```css
.flexy {
    display: flex
}
```

Is that it?

Yes!

What's so Special?

The special bit is, `display: flex`. It hands over the flexbox superpowers to the `.flexy` div.

What May I do With the Flexbox Super Powers?

Power is enjoyed when used.

The flexbox power handed over to the parent element makes no sense if you do not make use of it.

Of the many things you can do with flexbox, a simple one is to **center a child element within a parent element**

This centering will be along both sides. Vertical and Horizontal.

How to Vertically Center an Item

When you hand over the flexbox superpowers to an element, eg `.flexy`, the children element within this parent are called **flex-items**

Also, the Superman can fly, but he has to clench his fist and jump up first. Spiderman has webs, but he also needed to learn how to spin them with precision.

The same goes for the Flexbox superpowers.

It does come with the responsibility of learning some **keywords**. Those are its own superpower armory.

To vertically center the items within a parent element, do this:

```
    .flexy {
    align-items: center
    }
```

Do you see the keyword, `align-items` ?

It aligns the children of the parent element to the center - along the vertical.

How to Horizontally Center an Item

Thanks to the flexbox superpowers!

To horizontally center an item within a parent element, do this:

```
    .flexy {
    justify-content: center
    }
```

Do you see the keyword, `justify-content` ?

It is responsible for aligning items along the horizontal.

How then do you perfectly Center an Item?

"Perfectly center" here refers to centering along the horizontal and vertical.

The solution is simple.

Just combine all the tricks above, like so:

```
    .flexy {
    display: flex;
    justify-content: center,
    align-items: center
    }
```

Let's see that at play!

Do you see the white beauty in the blue background?

Please refer to the output above.

The white beauty is the perfectly centered child element!

Please see the CSS tab above.

Applying this Knowledge to the Sign Up Project.

Below is current the state of the project

| Output |
| :---: |
| HTML |
| CSS (SCSS) |

Ohans Emmanuel

I have worked with Startup X' product, and I think it is fabulous. I highly recommend the use of this product. It is simply a life saver.

Signup Form

Email Address Password

☐ I Agree with the terms of service

SIGNUP

To use flexbox, let's hand to the `body` element the flexbox super powers!

That way we can perfectly center the main content section within it. Like this:

```
body {
  display: flex;
  justify-content: center;
  align-items: center;
}
```

With this, we now have a perfectly centered `main` content area section.

Output
HTML
CSS (SCSS)

Ohans Emmanuel

I have worked with Startup X' product, and I think it is fabulous. I highly recommend the use of this product. It is simply a life saver.

Signup Form

Email Address	Password

☐ I Agree with the terms of service

SIGNUP

Note that, I have removed the `margin: 0 auto` declaration as it is no longer useful to us.

Some Other Flexbox Magic

It appears there's still more Flexbox has to offer. A lot more.

Let's see some of it.

> 1. You can have more than one element with the flexbox super powers.

Self explanatory. Isn't it?

What this means is, more than one element can have `display: flex` set on it.

2. In more technical terms, when you set `display: flex` on an element, it is better referred to as a **flex-container**

That being said, you can have more than one flex container in a single document.

3. You may even have nested flex containers!

Applying More than One Flex Container

In the sign up project, let us make the `main` content section a `flex-container`, i.e. the inclusion of this:

```scss
main {
display: flex
}
```

This is what you notice:

Output
JavaScript
HTML
CSS (SCSS)

Interesting!

The children of the flex container, i.e the `<section>` and `<form>` are aligned side by side. Automagically!

For visual feedback, I have reduced the size of the image, and added a red border around the child elements. Please see the css tab above.

If you didn't notice this, please take a look at the output above. The form now sits by the side of the other content section.

Flexbox Default Behaviour

The reason behind the behaviour above is largely due to the default behaviour of flex containers.

Flex containers align their child elements horizontally, and side by side.

A quick trick is, if you need elements to align side by side, you may want to consider wrapping them in a **flex container**. i.e an element for which `display: flex` has been set.

Conclusion

Flexbox is awesome, and we have only scratched the surface of what is possible.

Towards the end of this course I will include the link to an all encompassing resource to master Flexbox.

Guess the best part?

The resource is free, and it was written by me!

I guess that does NOT make you sad 😉