

# Introduction to Flexbox

Learning Flexbox can be a pain in the butt. For most persons, it's not particularly fun at first. It challenges you to rethink how you've dealt with layouts in css.

Don't fret. I really would walk you through all you need to know. So, let's get started.

## Why Use Flexbox?

CSS has evolved a lot over the past few years. Designers loved the introduction of filters, transitions, and transforms. But something was missing. Something we all craved.

The problem of crafting intelligent page layouts with CSS seemed to have persisted for too long, and this got many of us writing hacky CSS.

We always had to deal with floats, table display hacks, and the consequences they brought. If you've written CSS for sometime, you can probably relate to this. And if not, welcome to a better world!

It seems like our prayers as designers and front-end developers have finally been heard. This time, in grand style.

Now we can all ditch those hacky CSS tricks. No more incessant use of floats, table-cell displays.

It's time to embrace a cleaner modern syntax for crafting intelligent layouts. Welcome the CSS Flexbox model.

## What Is Flexbox?

According to the specification, the flexbox model provides for an efficient way to

layout, align, and distribute space among elements within your document, even when the viewport and the size of your elements is dynamic or unknown.

If that sound too formal, I understand the feeling. In just a bit, I'll explain what that means in plain English.

Whether you write css in your dreams or you're just getting started, you'll feel right at home.

## How do I start using the flexbox model?

This is the first question everyone asks, and the answer is much simpler than you may have expected.

To start using the Flexbox model, all you need to do is first define a **flex-container**.

Okay, did I make you more confused there? Let me explain.

In regular HTML, laying out a simple list of items takes this form.

```
<ul> <!--parent element-->
  <li></li> <!--first child element-->
  <li></li> <!--second child element-->
  <li></li> <!--third child element-->
</ul>
```



If you glanced at that, you must have seen that the unordered list, `ul` houses the list elements, `li`.

You'd call the `ul` the parent element, and the `li` the child element.

To use the flexbox model, you must make a parent element a **flex container** (*aka flexible container*).

You do this by setting `display: flex` or `display: inline-flex` for the inline variation. It's that simple, and from there you're all set to use the Flexbox model.

Told you it wasn't as difficult as you expected.

Set a certain parent element to be a **flex-container** by using `display: flex` | |

## inline-flex

Using an unordered list and a bunch of list elements, below is what initiating a Flexbox formatting context looks like.

```
/*Make parent element a flex container*/
ul {
  display: flex; /*or inline-flex*/
}
```



Style the list items just a bit, so you may see what's going on here.

```
li {
  width: 100px;
  height: 100px;
  background-color: #8cacea;
  margin: 8px;
}
```



Here is what you should have:

HTML CSS Output

```
1  /*Make parent element a flex container*/
2  ul {
3      display: flex; /*or inline-flex*/
4      list-style: none;
5  }
6
7  li {
8      width: 100px;
9      height: 100px;
10     background-color: #8cacea;
11     margin: 8px;
12 }
```

output

You may not have noticed, but something's happened already. The Flexbox formatting context is now initiated.

Remember that by default, “divs” in CSS stack vertically, from top to bottom.



The image above is the result you may have hoped for. However, with the inclusion of that simple one-liner, `display: flex`, you can immediately see a change in layout.

The list elements are now stacked horizontally, from left to right. Just like they would if you used float



The Flexbox model kicks in as soon as you introduce `display: flex` on any parent element.

You may not understand why that change in the orientation of the list elements came to be. I promise I'll go into the inner workings of that very soon. For now, blind trust would suffice.

blind trust would suffice.

There's one more thing I need to call your attention to.

As soon as you set the display property to flex, the unordered list automatically becomes the **flex container** and the child elements (in this case, the list elements `li`) become **flex items**.

These terms would come up over and over again as I walk you through some more interesting things the Flexbox model has in place.

I have used two key words, and I'd like to lay more emphasis on them as they are vital to understanding what lies ahead.

1. Flex container : The parent element you've set `display: flex` on.
2. Flex items : The children elements within a Flex container.

**Flex container:** the parent element in which flex items are contained

**Flex item(s):** the child/children of the flex-container

This is the foundation for using the flexbox model, and as soon as that's understood, more interesting things lie ahead.

Here is where I wrap up the Introduction to the Flexbox model.

See you in the next lesson.