

# COMP 1017

Day 15

Introduction to Flexbox

So far, we've been designing websites with **one** big long block of content.

... but a lot of websites have more  
than one column of stuff.

So, how can we get our boxes to  
render side by side?

# Introducing Flexbox

weird flex, but ok

The CSS Flexible Box Layout Model (**Flexbox**) is a way to order, align, and lay out website content.

While it became an official W3C recommendation in 2012, we've been waiting for **some browsers** to reach their end of life before using this as the go-to layout technique.





Flexbox is our new go-to and, for the purpose of this course, we won't worry about backwards compatibility or using other techniques.

... so, how can we create multiple  
column layouts?

# Block-Level & Inline Elements

listen, blockhead!

What is an element's  
`display` property?

It is how the browser will **render** it  
and **lay it out** on the page by  
default.

Let's take a look at two types:  
block-level elements and  
inline elements.

By default, many HTML elements will render as a **block** — that is, it will take up the entire width of its container.

By default, **block-level** elements  
will always occupy their own  
unique row.



Even if there is enough space,  
**block-level** elements will also  
always start a new line.

... and we've seen tonnes of these!

<p>

<h1>-<h6>

<ol>,<ul>

<li>

<blockquote>

<address>

<table>

<tr>

<dl>

```
.box {  
    display: block;  
}
```

I'm a block-level element. I'm a block-level element. I'm a block-level element.  
I'm a block-level element. I'm a block-level element. I'm a block-level element  
I'm a block-level element. I'm a block-level element. I'm a block-level element  
I'm a block-level element. I'm a block-level element. I'm a block-level element

I'm a block-level element. I'm a block-level element. I'm a block-level element.  
I'm a block-level element. I'm a block-level element. I'm a block-level element  
I'm a block-level element. I'm a block-level element. I'm a block-level element  
I'm a block-level element. I'm a block-level element. I'm a block-level element

I'm a block-level element. I'm a block-level element. I'm a block-level element.  
I'm a block-level element. I'm a block-level element. I'm a block-level element  
I'm a block-level element. I'm a block-level element. I'm a block-level element  
I'm a block-level element. I'm a block-level element. I'm a block-level element

**Inline** elements are a little different. They do not start a new row and only take up as much space (width) as they need.

If there is any space leftover,  
**inline** elements are rendered  
**side-by-side.**

We've only seen a few  
**inline** elements, but we will be  
learning a few more.



`/* We have seen these.*/`

`/* We will see these.*/`

`<a>`

`<img>`

`<button>`

`<input>`

`<label>`

However, they don't accept `width` or `height` properties, and you cannot set the `top` and `bottom` margins.

```
.box {  
    display: inline;  
}
```

Clouds are free.

They just float  
around the sky  
all day and have  
fun.

Absolutely no  
pressure. You  
are just a  
whisper floating  
across a  
mountain. Isn't  
that fantastic?

So, let's use `flexbox`.

In order to use it, you must apply it  
to a parent container.

The parent becomes the  
flex container and the  
children become flex items.

container



items





```
div {  
    display: flex;  
}
```

There are *many* other properties  
for **flexbox**; don't worry about  
them yet.

For now, we'll use it to create multiple column layouts.

# Supplemental Video

Build an HTML + CSS Layout with Flexbox

<https://www.youtube.com/watch?v=aRMldKRYg6c>