Front-End Web Development

Flexbox

GENERAL ASSEMBLY

Today's Learning Objectives

In this lesson, you will:

- Use flexbox properties to create responsive layouts.
- Apply flexbox properties to container elements to organize **`flex-item`** elements.
- Apply flexbox properties to **`flex-item`**s to differentiate unique elements.

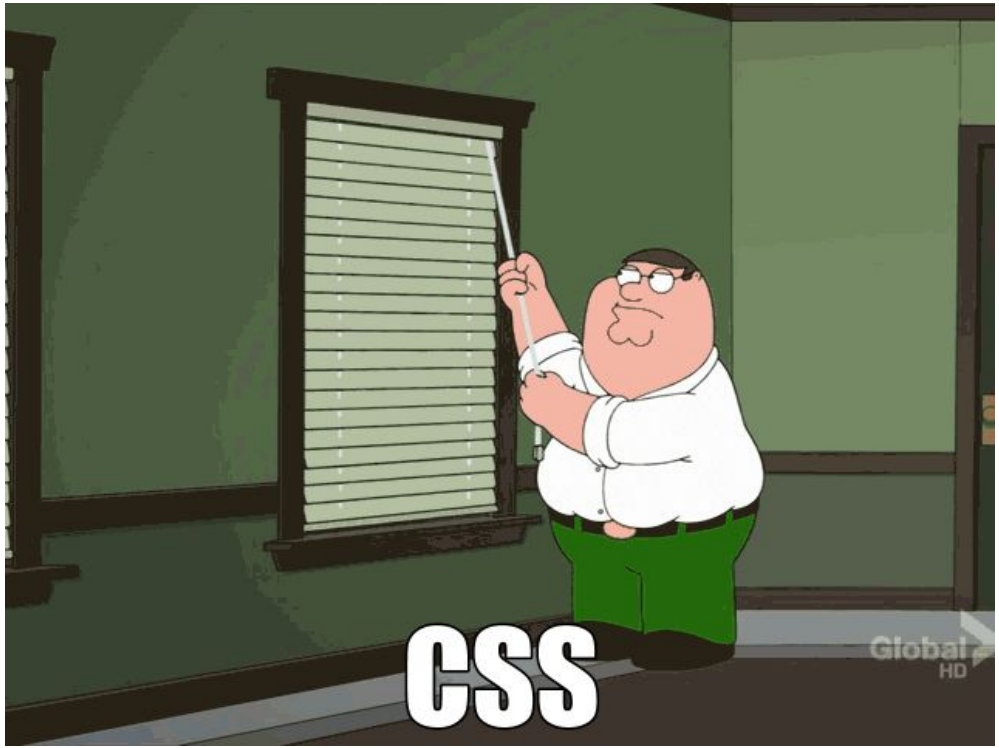
Front-End Web Development

CSS: Don’t Float On, Alright?

CSS only recently got effective layout tools with flexbox and CSS Grid. For years we were stuck with floats and clears, which are now unnecessary evils.

 © 2020 General Assembly

Float Layouts

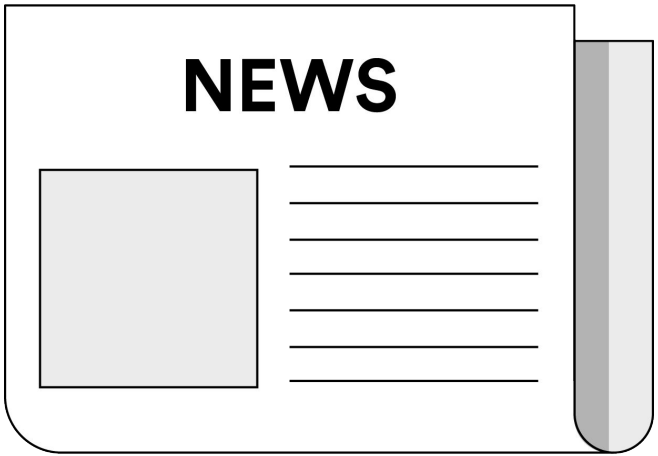


Why, Floats, Why?

CSS was written a LONG time ago, in a galaxy far, far away (the 1990s).

There was no concept of a multi-device universe or interactive websites. Everything was based around print layouts.

That's why we had the float-based system — it comes from print design where you **float** images in a sea of words, like in a magazine.



The Problem With Floats

Although floating was intended for wrapping text around an image newspaper-style, the lack of alternatives offered by CSS for page layouts meant it became abused for all manner of layout applications.

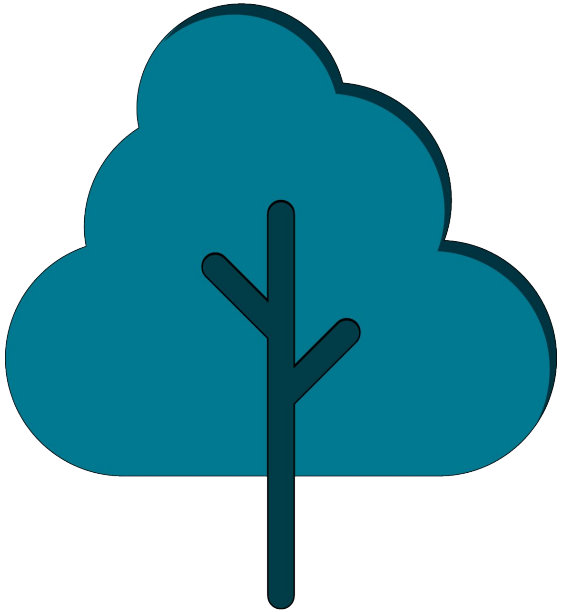
Unfortunately, float properties disrupt the normal flow of a document and cause all sorts of unintended interactions with other elements on the page.

This means you have to nudge a lot of other elements out of the way, especially if you're using multiple floated elements. You end up with a brittle system of spacing properties and a lot of spaghetti CSS!

Front-End Web Development

How Does Flexbox Work?

Key Idea: Remember the DOM Tree

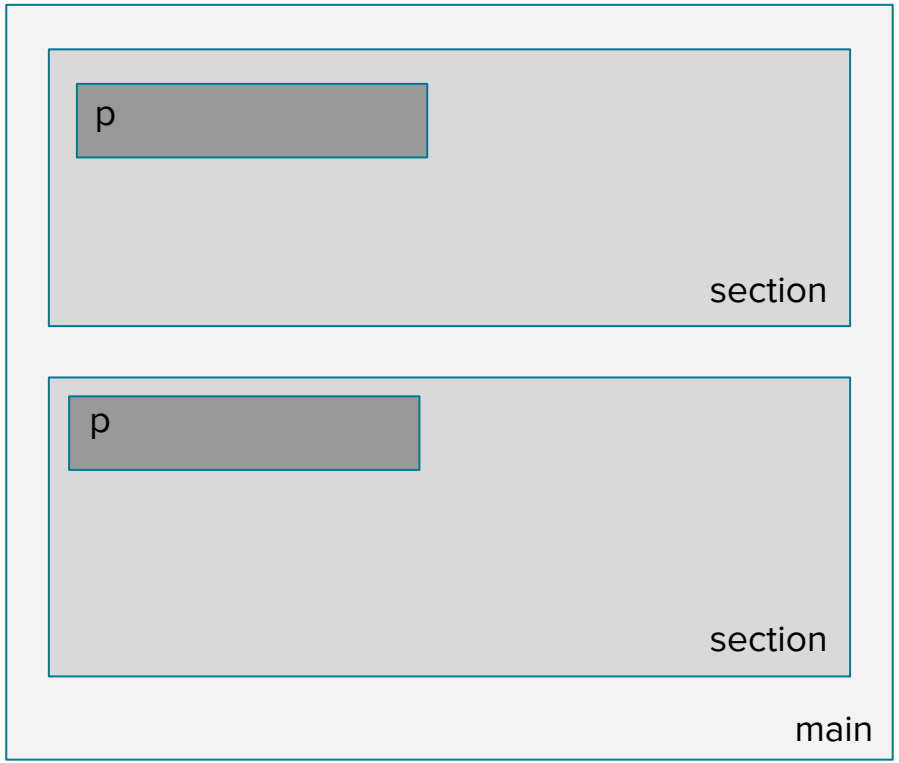


It's a visual diagram of a webpage's HTML structure.

We Can Visualize our HTML

```
<main>
  <section class="1">
    <p>Content</p>
  </section>
  <section class="2">
    <p>More content</p>
  </section>
</main>
```

Remember, by default, HTML elements are blocks that stack vertically on a page, like we see here on the right.



 © 2020 General Assembly

Flexbox Goes on the Parent Element

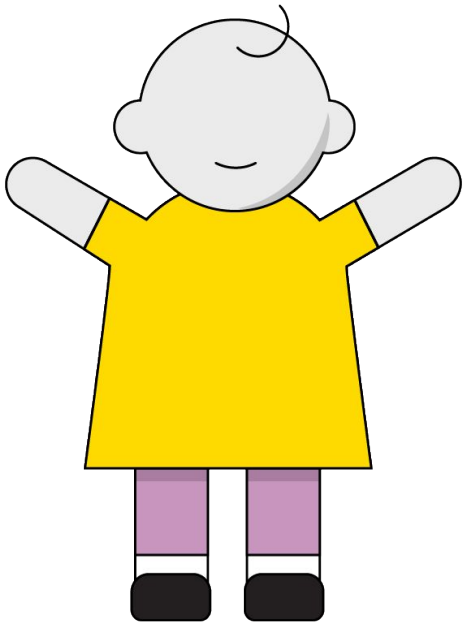
Flex **containers** are the elements that contain the elements you are trying to align.

Flex containers hold objects that contain child elements (like this bounce house). The parent element won't flex, but it will tell its children to be flexible!

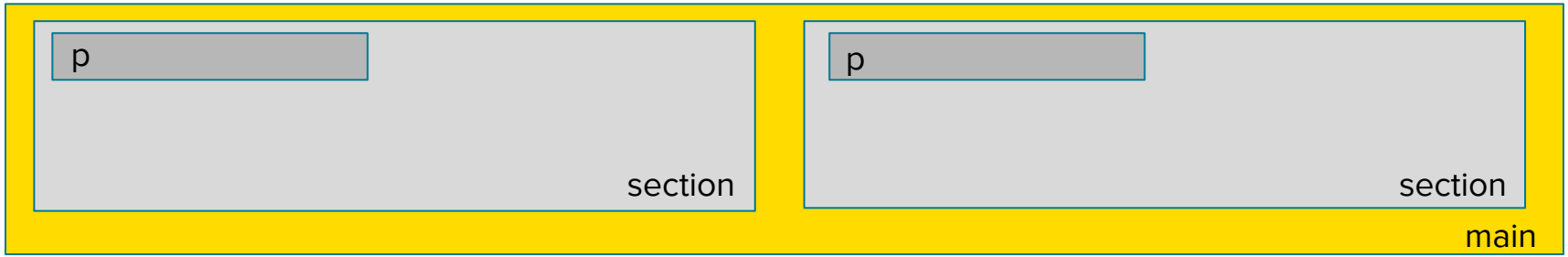
Children (Items)

Flex **items** are the children that go inside the parent container.

In simple layouts, the **child elements may not receive any special styling** — which may surprise you.



`display: flex;` in Action



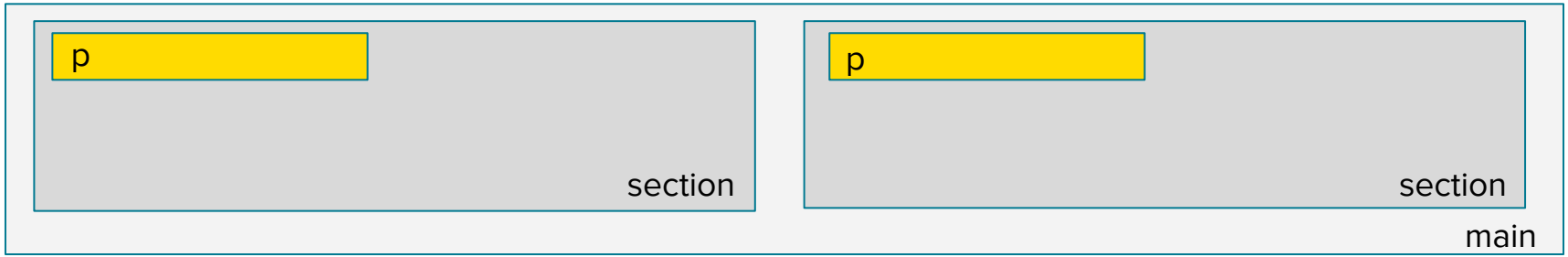

```
<main>
  <section class="1">
    <p>Content</p>
  </section>
  <section class="2">
    <p>More content</p>
  </section>
</main>
```

```
/* css */
main {
  display: flex;
}
```

When you put **`display: flex;`** into a parent element, its children become an orderly row. In this example, **`<main>`** controls the two **`<section>`** elements immediately beneath it in the DOM tree.

 © 2020 General Assembly

What Didn't Happen in This Example




```
<main>
  <section class="1">
    <p>Content</p>
  </section>
  <section class="2">
    <p>More content</p>
  </section>
</main>
```

```
/* css */
section {}
p {}
```

Notice that, despite the **<section>** elements being in a row, nothing happened to the **<p>**. This is because **flexbox only affects one level down in the DOM** — no more. Also notice that the **<section>** elements didn't receive any styles. They are controlled solely by the parent.

 © 2020 General Assembly

When Building Flexbox Layouts...

...Every container must have...

```
display: flex;
```

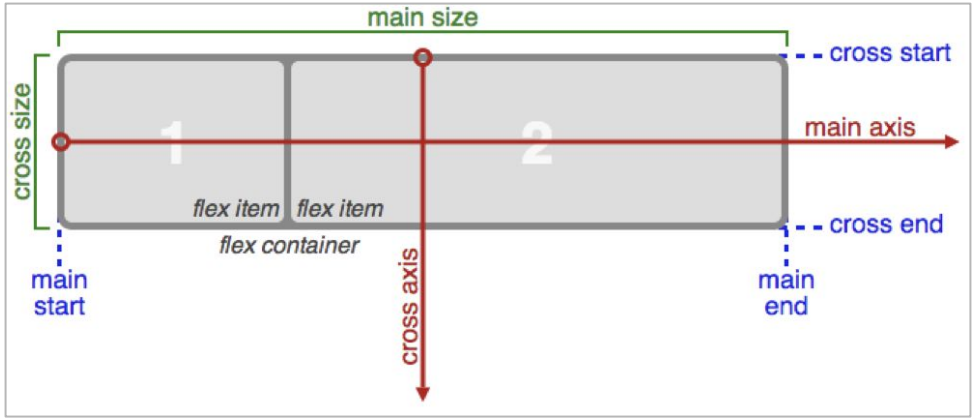
...which also includes the following properties set by default:

```
flex-direction: row;
justify-content: flex-start;
align-items: stretch;
```

What do you think those other properties do?

Flexbox Gives You Control Over Alignment

- The main axis is horizontal by default and controlled by **`justify-content`**.
- The cross axis is controlled by the **`align-items`** property of the parent.
- Direct children automatically fall into place.



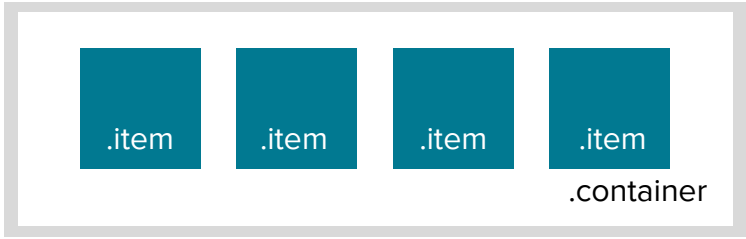
 © 2020 General Assembly

Flexbox Positioning

When you put `justify-content: center;` onto a parent element, its children squeeze together in the **horizontal center** of the container.

`align-items: center;` on the parent element moves children to the **vertical center** of the container.

Once again, notice there are **no flex positioning styles on the children (.item)**.




```
/* CSS */
.container {
  display: flex;
  justify-content: center;
  align-items: center;
}

.item {
 background-color: teal;
 height: 100px;
 margin: 5px;
 width: 100px;
}
```

 © 2020 General Assembly

Guided Walk-Through:

Flexing Our Layout Muscles

Refer to this interactive CodePen for the following examples of how to use flex properties on parent and child elements. They are many more properties:

Reference code:

https://codepen.io/GAmarketing/pen/QWWJvLx

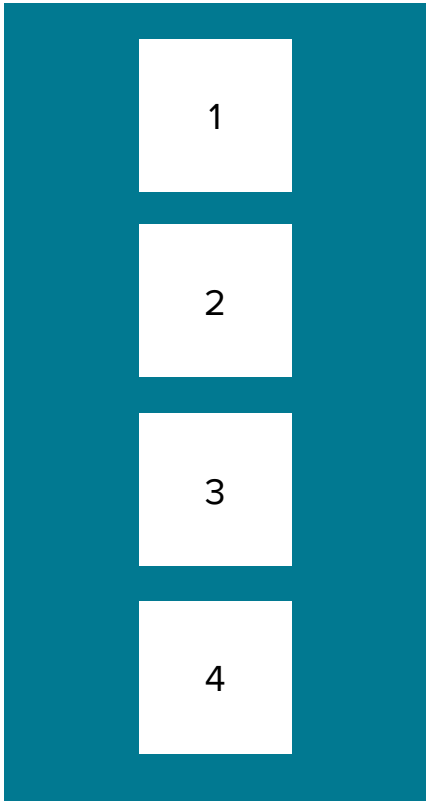
Front-End Web Development

Flex Parent Properties Reference

flex-direction

Think about orientation — flexbox layouts are inherently vertical or horizontal.

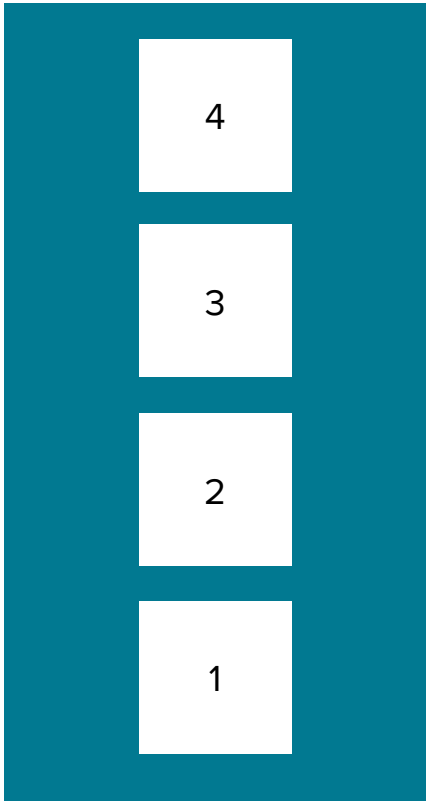
```
flex-direction: column;
```



`flex-direction` (Cont.)

You can easily flip the display order without reordering your HTML!

```
flex-direction: column-reverse;
```

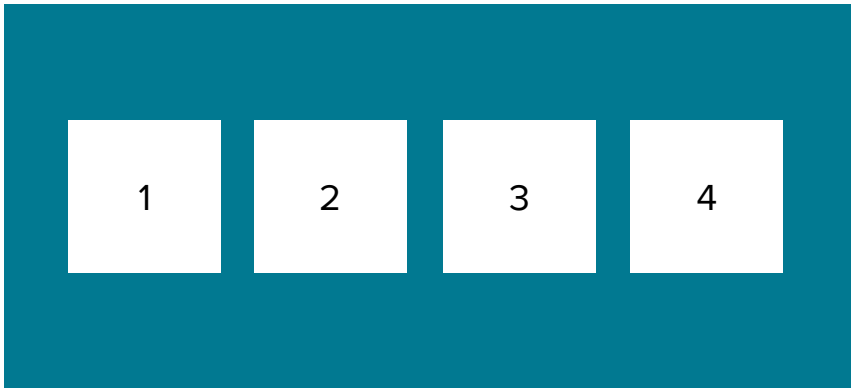


 © 2020 General Assembly

`flex-direction` (Cont.)

You can also do layouts in a row. This is the default, and flexbox will cram every child into this single row.

```
flex-direction: row;
```

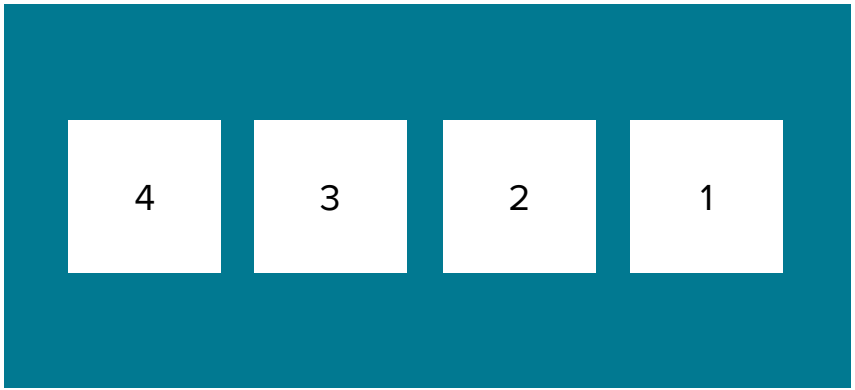


 © 2020 General Assembly

`flex-direction` (Cont.)

You can flip rows, too! This is very advantageous for users accustomed to right-to-left languages.

```
flex-direction: row-reverse;
```

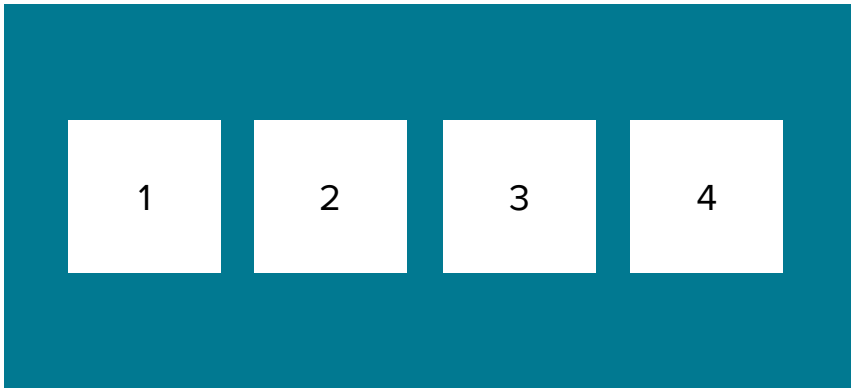


 © 2020 General Assembly

flex-wrap

By default, all boxes are stuffed into one row.

```
flex-wrap: nowrap;
```

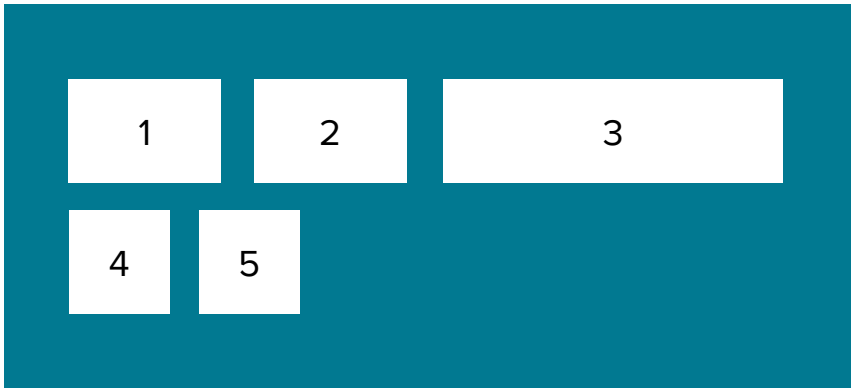


 © 2020 General Assembly

`flex-wrap` (Cont.)

But you can make them pop out into additional rows as needed.

```
flex-wrap: wrap;
```

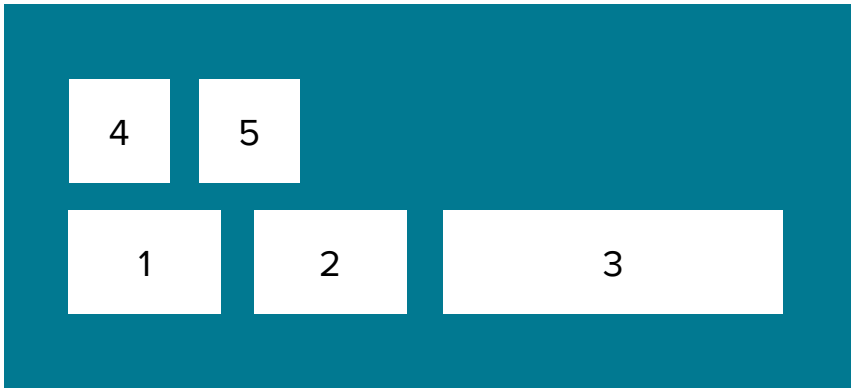


 © 2020 General Assembly

`flex-wrap` (Cont.)

They can display right to left and bottom to top as well.

```
flex-wrap: wrap-reverse;
```

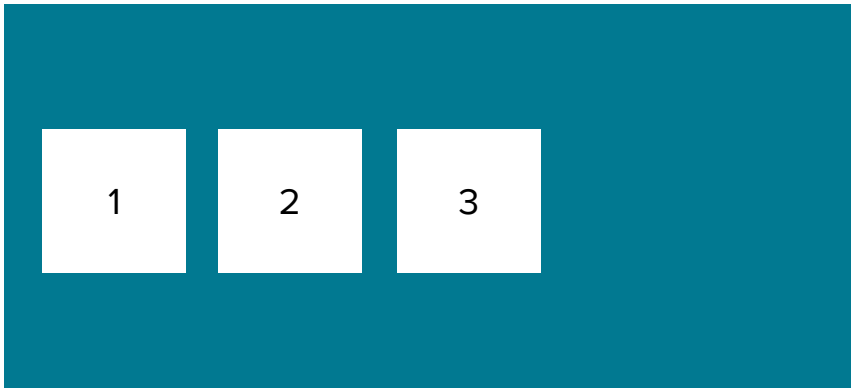


`justify-content`

This controls how boxes are spaced in flexbox rows/columns.

`flex-start` pushes everything toward the start direction.

```
justify-content: flex-start;
```



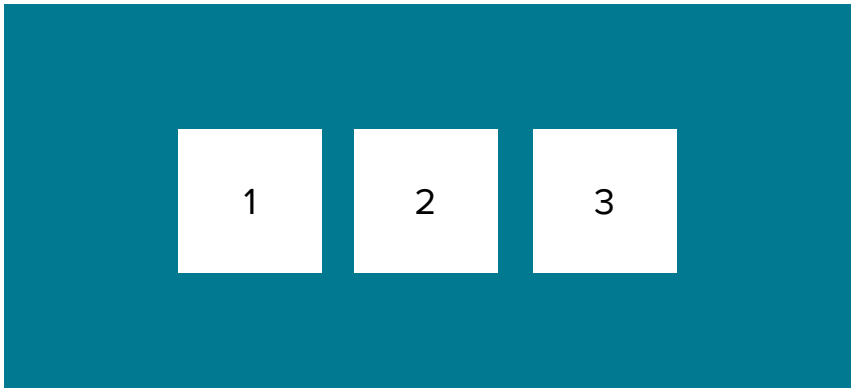
 © 2020 General Assembly

`justify-content` (Cont.)

Centering is easy! Note, however, that auto margins don't work in flex land.

Either go flex all the way or don’t go flex at all.

```
justify-content: center;
```



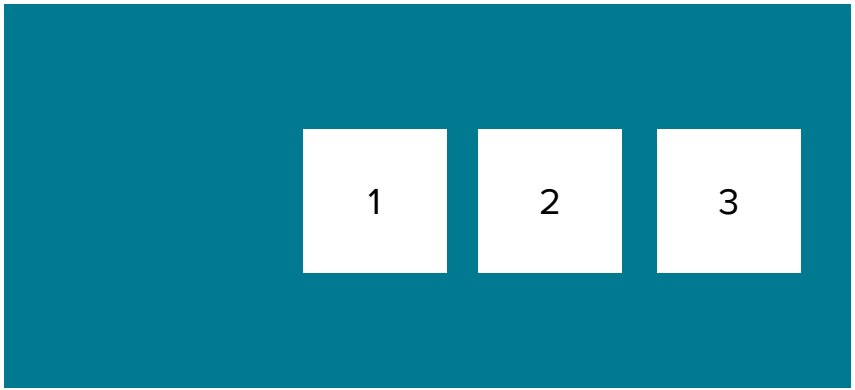
 © 2020 General Assembly

`justify-content` (Cont.)

Push everything to the end direction with **`flex-end`**.

This is similar to **`text-align: right`**; but for layouts!

`justify-content: flex-end;`

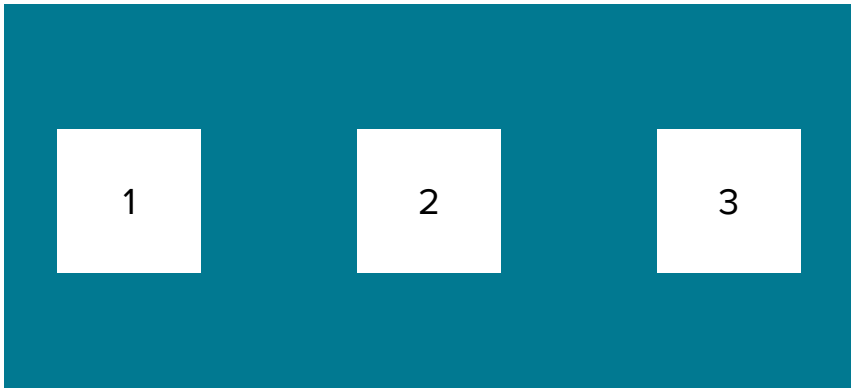


 © 2020 General Assembly

`justify-content` (Cont.)

Push stuff as far apart as possible using:

```
justify-content: space-between;
```

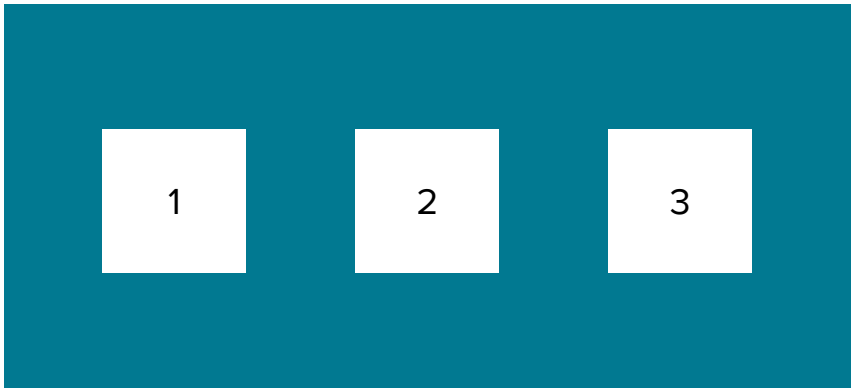


 © 2020 General Assembly

`justify-content` (Cont.)

Center with respect to total row/column width, leaving equal spacing between each item.

```
justify-content: space-around;
```



 © 2020 General Assembly

`align-items`

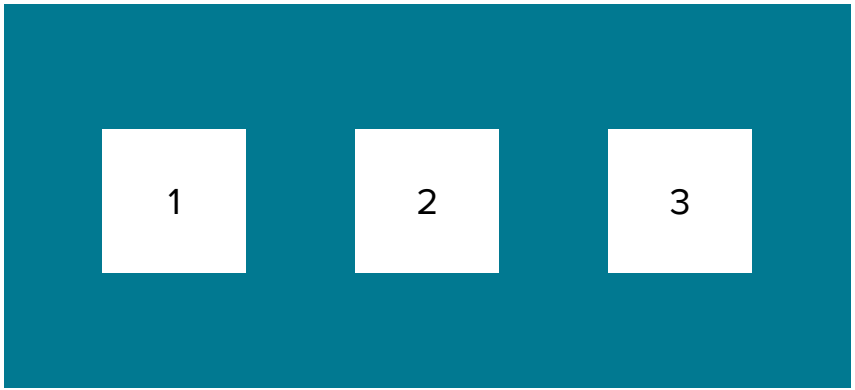
controls vertical alignment — hurrah!

It only took CSS 20 years...

align-items

Center children items vertically.

```
align-items: center;
```

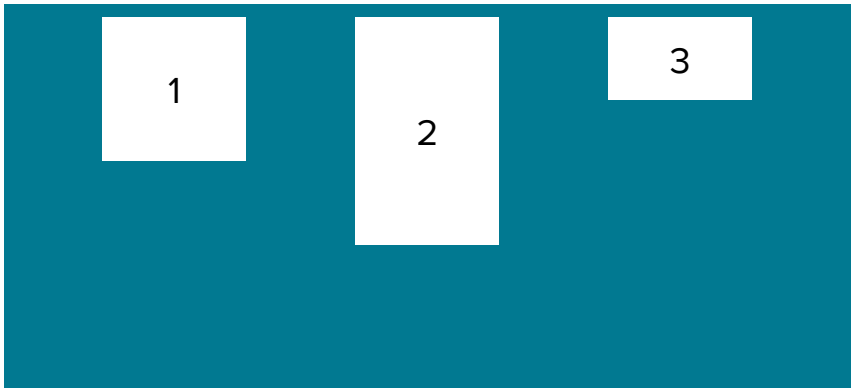


 © 2020 General Assembly

`align-items` (Cont.)

Align children with the top of the box, even if they have different heights.

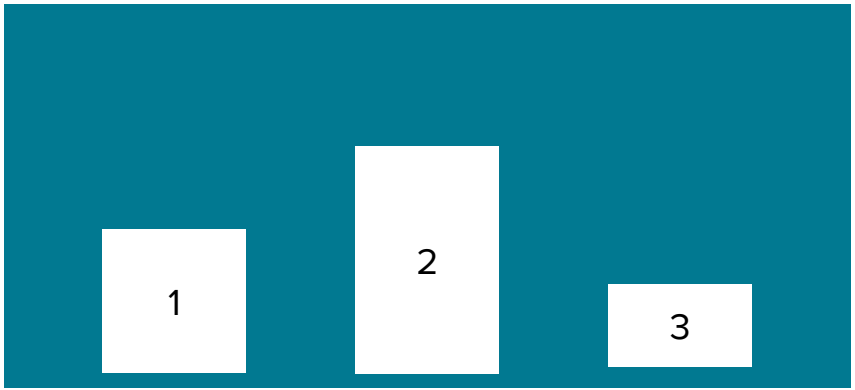
```
align-items: flex-start;
```



`align-items` (Cont.)

Bottom align children, even if they have different heights.

```
align-items: flex-end;
```



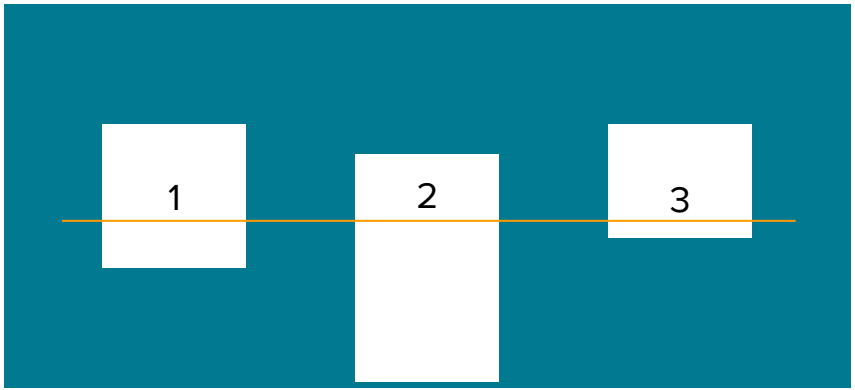
 © 2020 General Assembly

`align-items` (Cont.)

Align children by the middle baseline of the text in the child.

***Note**: The orange line is for demonstration only — you won't see this IRL.*

```
align-items: baseline;
```

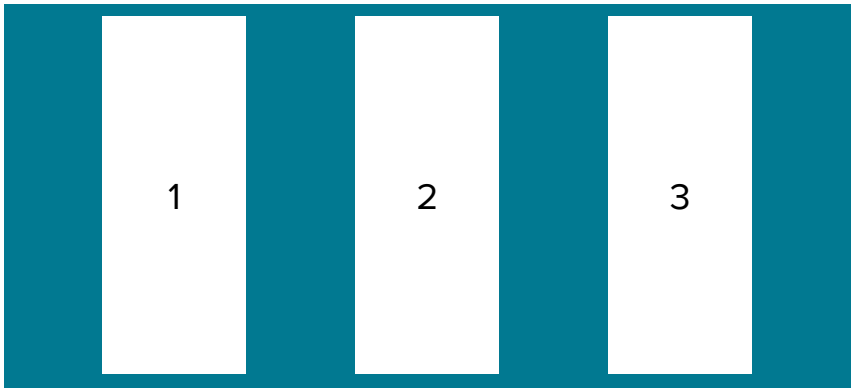


 © 2020 General Assembly

`align-items` (Cont.)

Stretch children to fill the container.

```
align-content: stretch;
```



 © 2020 General Assembly

Partner Exercise:

Nav With Flexbox

10 minutes

In this exercise, we'll build a navigation bar using flex properties. Look closely at the example image for spacing details!

Starter code:

https://codepen.io/GAmarketing/pen/OJJagJr

Solution code:

https://codepen.io/GAmarketing/pen/oNNQwjp

Front-End Web Development

Flex Child Properties Reference

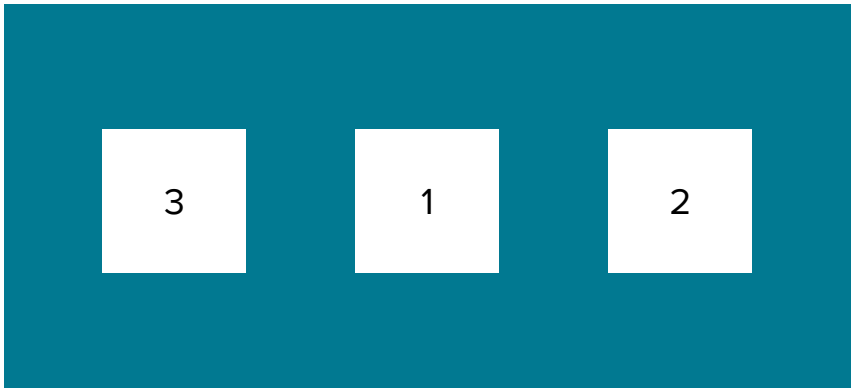
order

Change the order in which children render within a row.

```
order: 3;

order: 1;

order: 2;
```



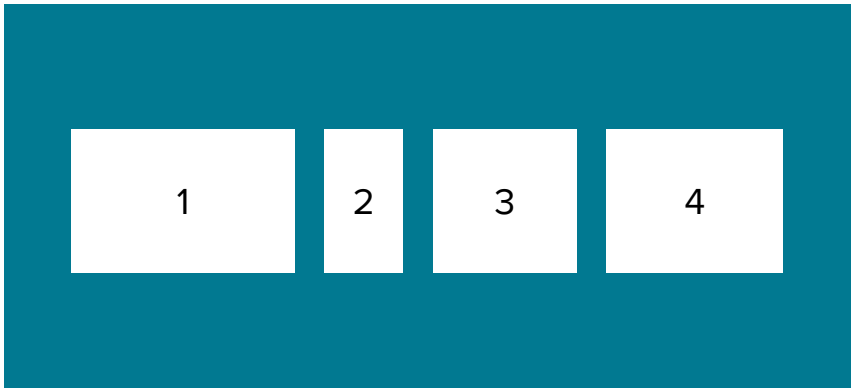
flex-grow

Control the rate at which individual children grow in width across viewports.

```
flex-grow: 2;
```

```
flex-grow: 0.5;
```

```
flex-grow: 1;
```

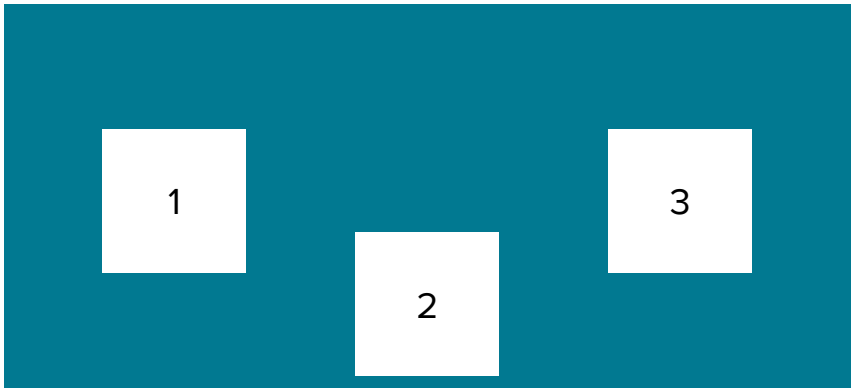


 © 2020 General Assembly

align-self

This works like **align-items** but for an individual child, so you can override on a per-item basis.

```
align-self: flex-end;
```



Front-End Web Development

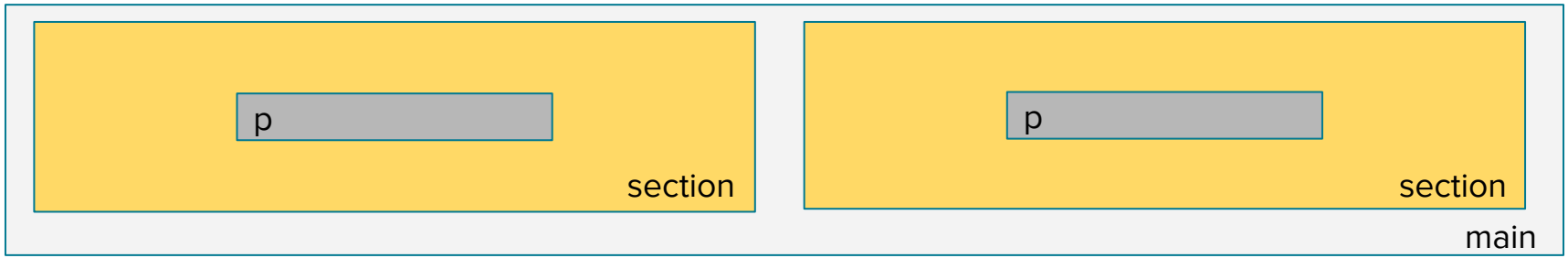
Nested Flexbox Layouts

Can You Flex Within Flex?

You can nest a flex container inside of a flex container — no problem.

Flex container properties only adjust their direct children, so you may need several layers of flexing to drill down to further descendent elements!

Nested Flexboxes Require Dual Flex Declarations




```
<main>
  <section class="1">
    <p>Content</p>
  </section>
  <section class="2">
    <p>More content</p>
  </section>
</main>
```

```
/* css */
main {
  display: flex;
}
section {
  align-items: center;
  display: flex;
  justify-content: center;
}
```

<section> now plays two roles: It's the child of **<main>** but also the parent of <p> elements. When flex is applied to any element — even a flex child — it will control the children directly beneath it. You nest flexboxes indefinitely using this method.

 © 2020 General Assembly

Solo Exercise:

Airbnb Mockup

60-90 minutes

Many professional websites use flex properties, including Airbnb! Practice your new flex skills by recreating the examples provided in the screenshots.

Starter code:

https://codepen.io/GAmarketing/pen/LYYBaEX

Solution code:

https://codepen.io/GAmarketing/pen/Yzzjqqb

Key Takeaways

Flex, Don't Float!

- Container elements given `display: flex` can arrange their children.
- Children elements can be given specific flex properties just for them.
- Flex always works along one main axis.

 © 2020 General Assembly

For Next Time

Layouts in the Second Dimension

- Complete Homework No. 2
- CSS Grid is coming up!


GA