# Flexbox

flex containers + flex items

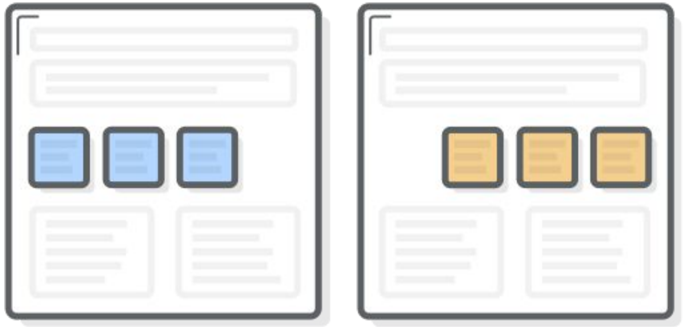
## Flexbox

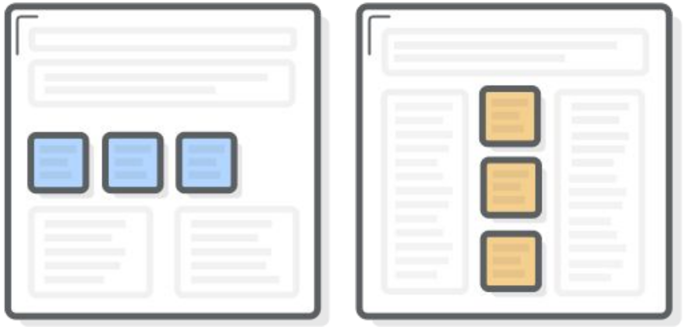
The “Flexible Box” or “Flexbox” layout mode offers an alternative to Floats for defining the overall appearance of a web page. Whereas floats only let us horizontally position our boxes, flexbox gives us *complete* control over the alignment, direction, order, and size of our boxes.

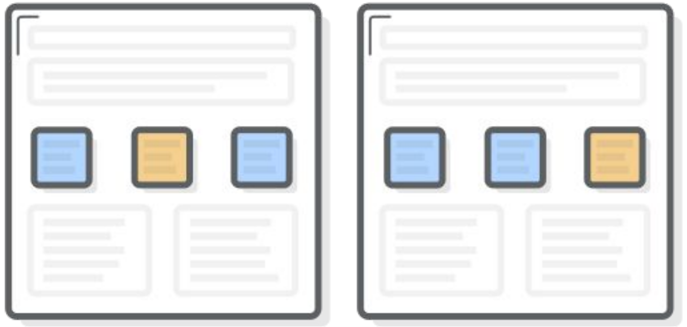
## Flexbox vs Grid

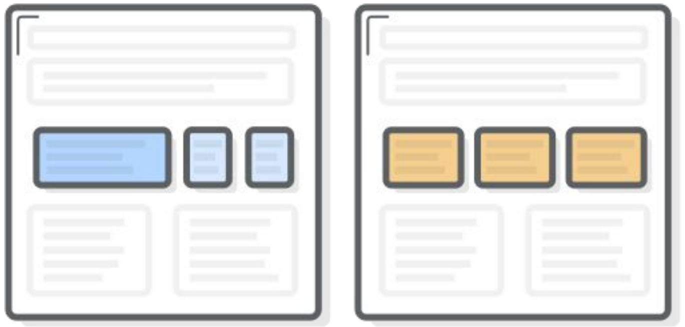
CSS has always been used to lay out our web pages, but it's never done a very good job of it. First, we used tables, then floats, positioning and inline-block, but all of these methods were essentially hacks and left out a lot of important functionality (vertical centering, for instance). Flexbox helped out, but it's intended for simpler one-dimensional layouts, not complex two-dimensional ones (Flexbox and Grid actually work very well together).

Flexbox is generally used for smaller, simpler layouts and Grid for complicated 2-Dimensional ones.  Flexbox is also very useful for handling a large amount of different content sizes.


ALIGNMENT


DIRECTION

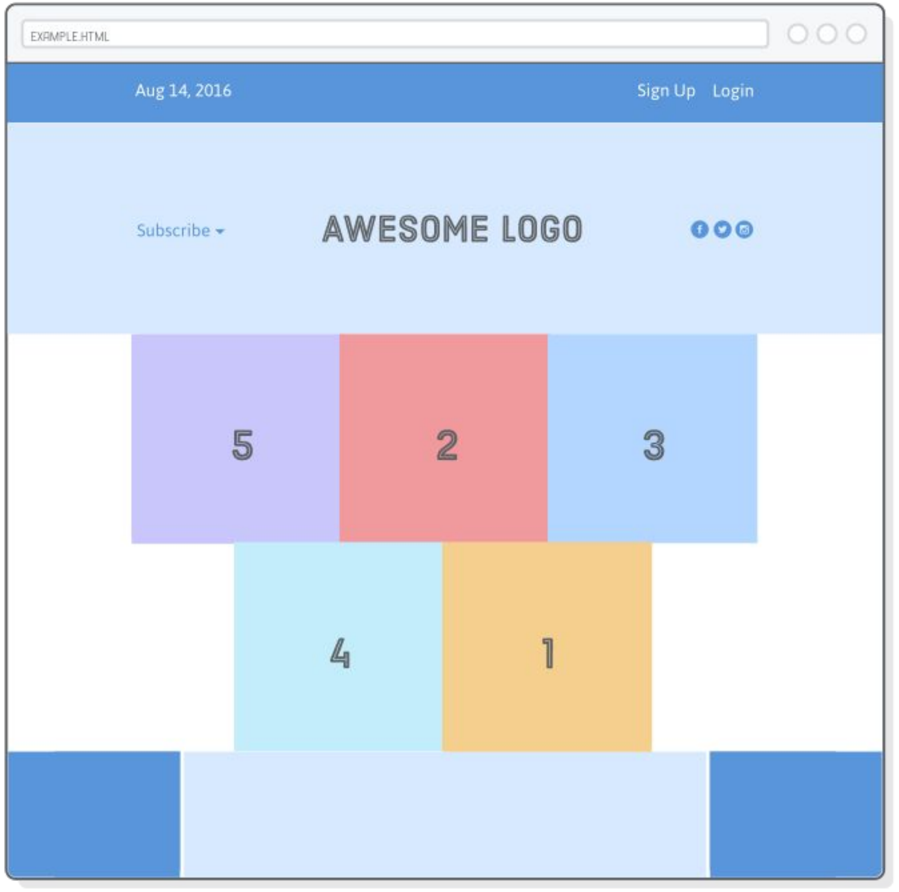

ORDER


SIZE

## Class files

class6/flexbox/flexbox.html

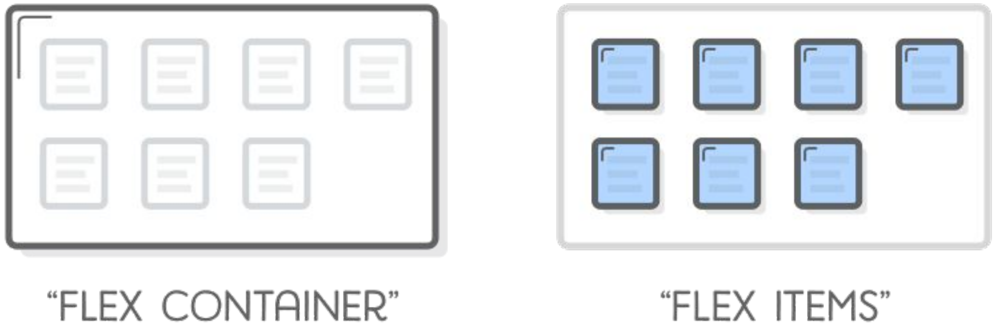
class6/flexbox/flexbox.css



Finished Example

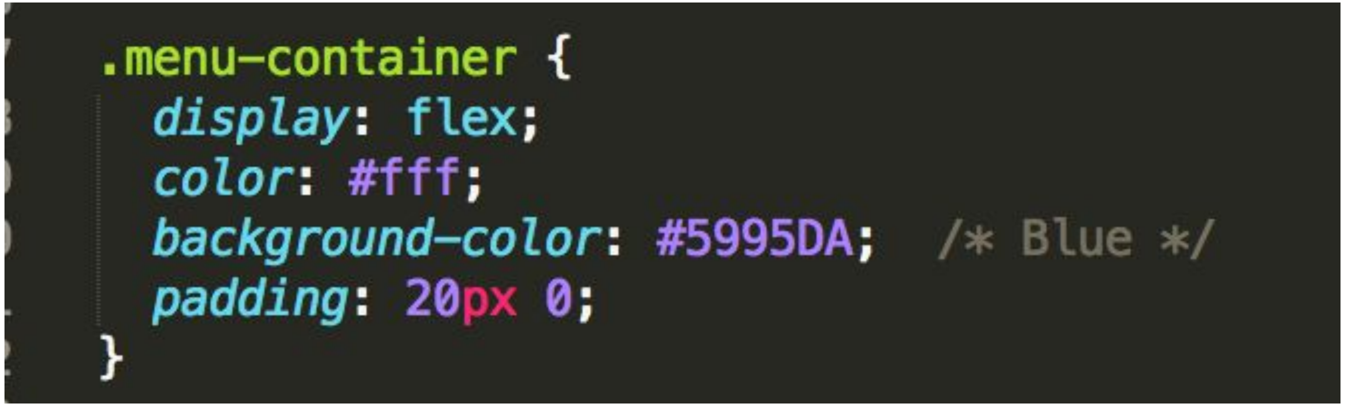
## Flexbox Overview

Flexbox uses two types of boxes:  **flex containers** and **flex items**.  The job of a flex container is to group a bunch of flex items together and define how they're positioned.



## Flex Containers

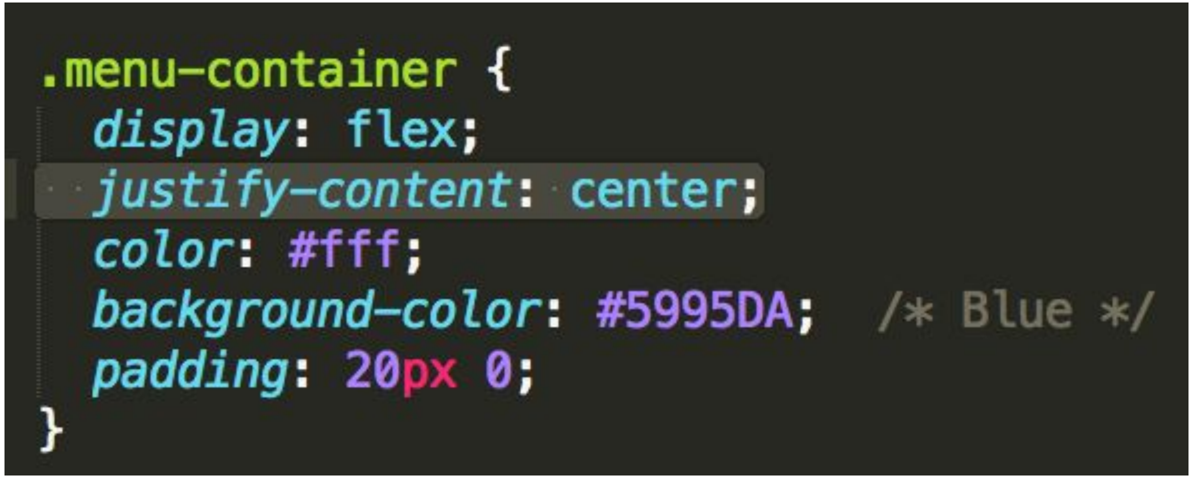
The first step in using flexbox is to turn one of our HTML elements into a flex container. We do this with the display property. By giving it a value of flex, we’re telling the browser that everything in the box should be rendered with flexbox instead of the default box model.



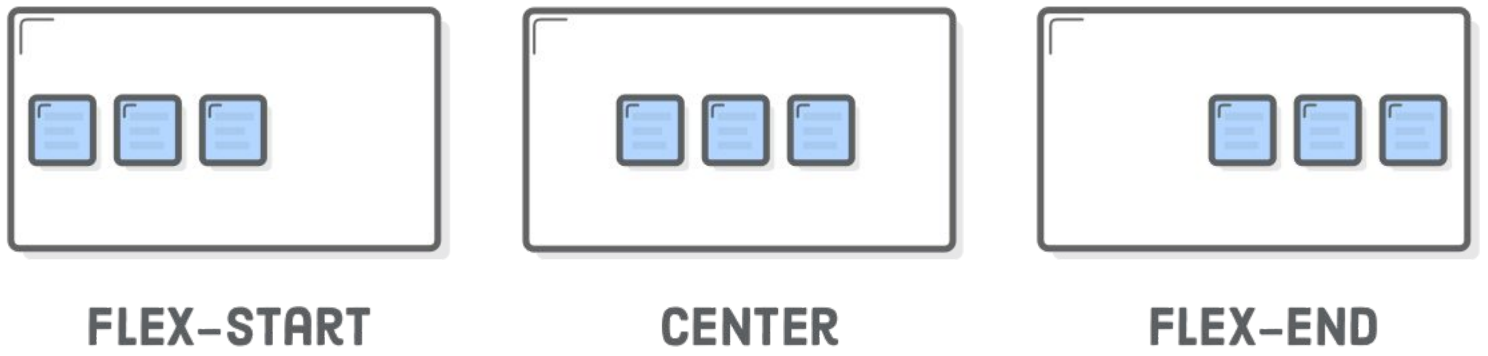
```
.menu-container {
  display: flex;
  color: #fff;
  background-color: #5995DA;  /* Blue */
  padding: 20px 0;
}
```

## Aligning Flex Items

Now we can specify the horizontal alignment of the items:



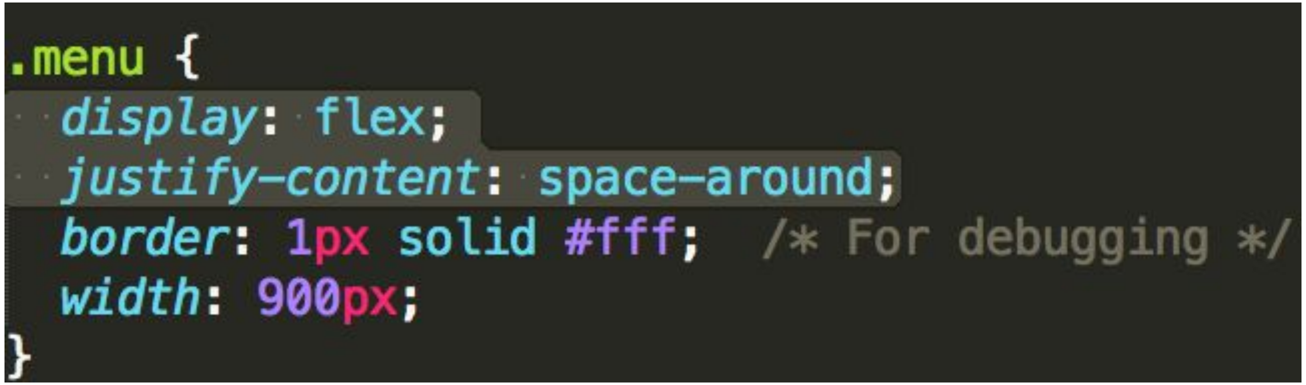
```css
.menu-container {
  display: flex;
  justify-content: center;
  color: #fff;
  background-color: #5995DA;  /* Blue */
  padding: 20px 0;
}
```



Other values for justify-content are shown below:

- center
- flex-start
- flex-end
- space-around
- space-between

## Multiple Flex Items



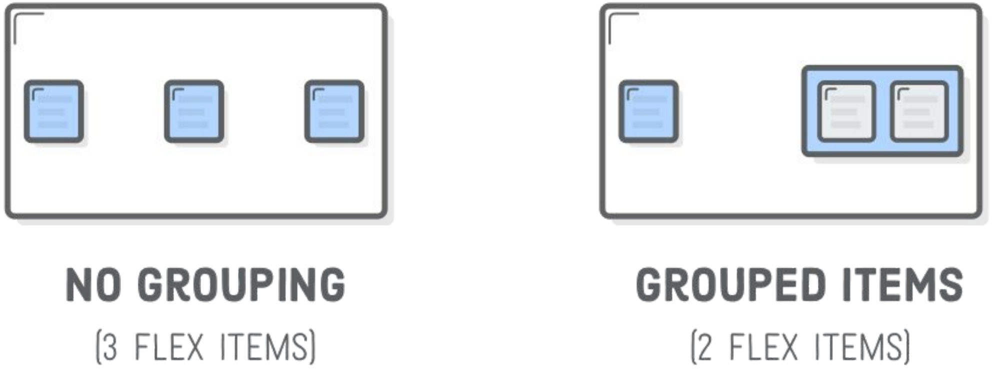
```
.menu {
  display: flex;
  justify-content: space-around;
  border: 1px solid #fff;  /* For debugging */
  width: 900px;
}
```

This turns our .menu into a nested flex container, and the space-around value spreads its items out across its entire width.

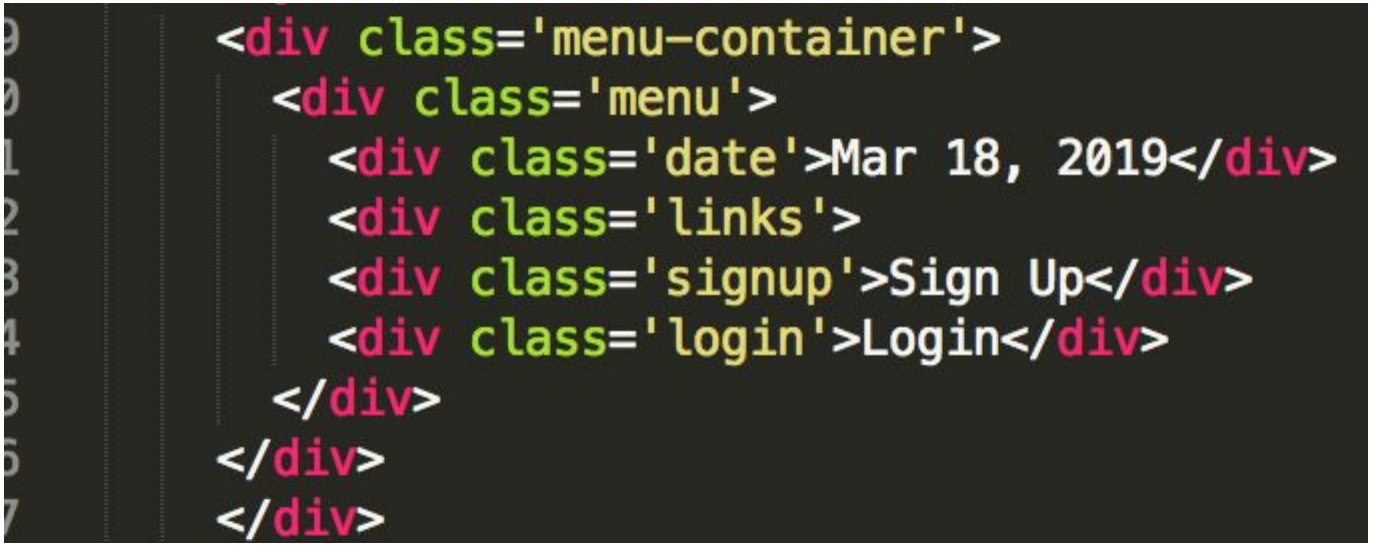
EXAMPLE.HTML
Aug 14, 2016
Sign Up
Login

## Grouping Flex Items

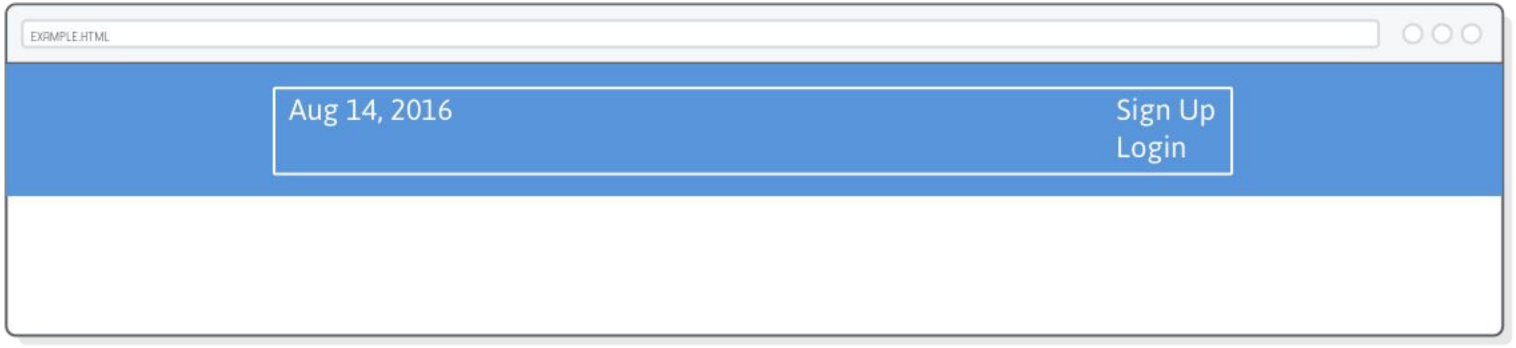
Flex containers only know how to position elements that are one level deep (i.e., their child elements). They don't care one bit about what's inside their flex items. This means that grouping flex items is another weapon in your layout-creation arsenal. Wrapping a bunch of items in an extra <div>results in a totally different web page.



For example, let’s say you want both the **Sign Up** and **Login** links to be on the right side of the page, as in the screenshot below. All we need to do is stick them in another <div>:



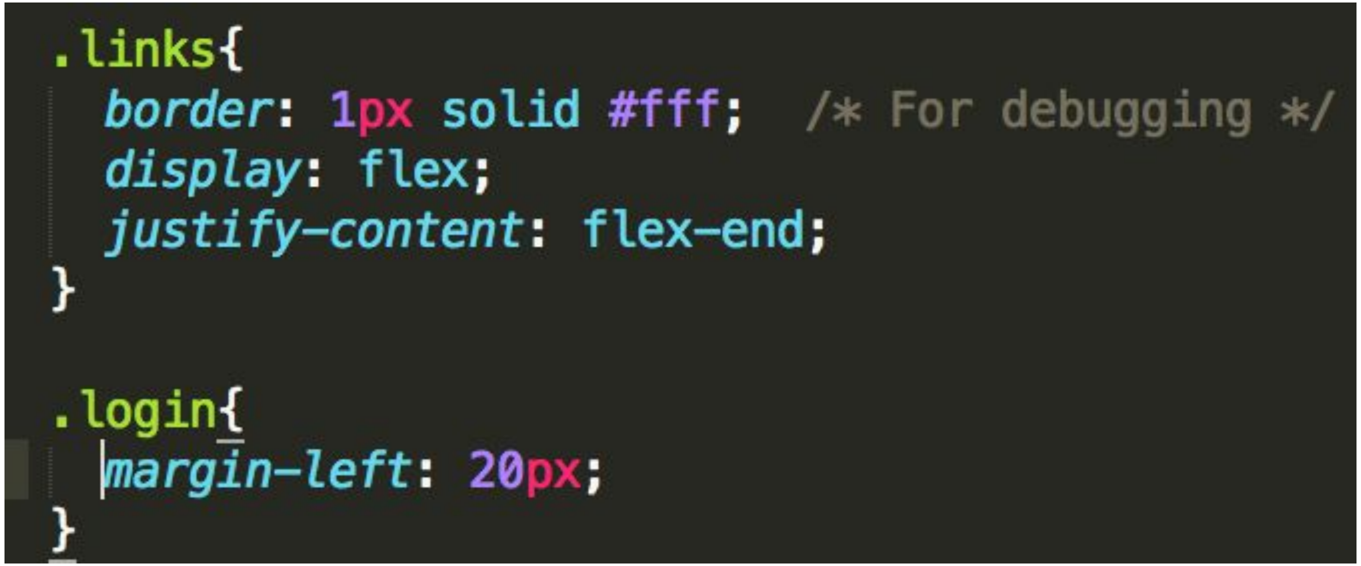
```
<div class='menu-container'>
  <div class='menu'>
    <div class='date'>Mar 18, 2019</div>
    <div class='links'>
    <div class='signup'>Sign Up</div>
    <div class='login'>Login</div>
  </div>
</div>
</div>
```


EXAMPLE.HTML

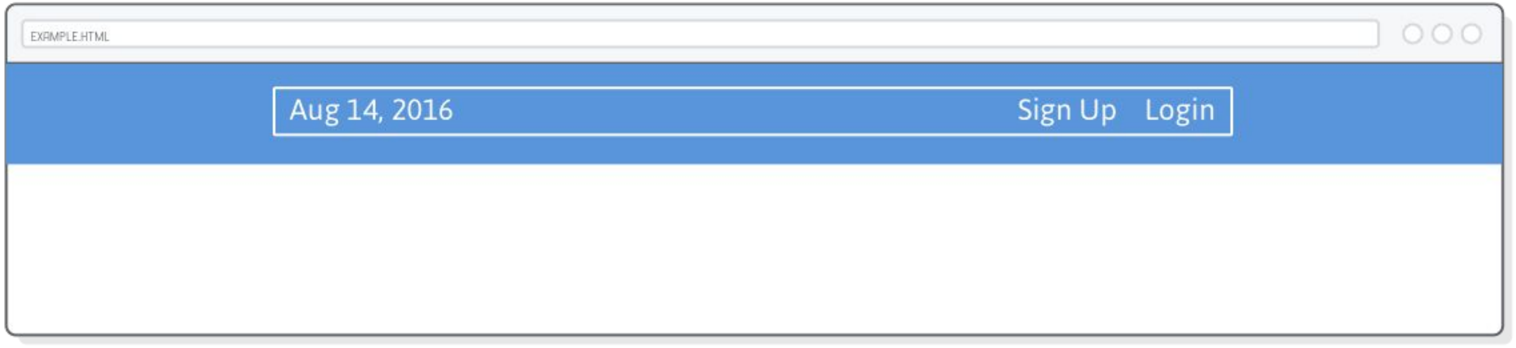
Aug 14, 2016

Sign Up
Login

But, now we need to lay out the .links element because it’s using the default block layout mode. The solution: more nested flex containers! Add a new rule to our styles.css file that turns the .links element into a flex container:

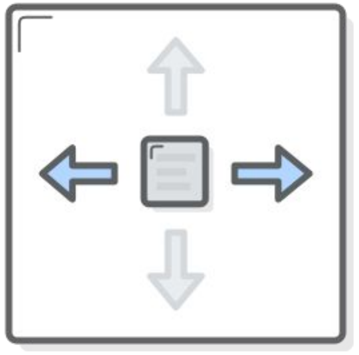


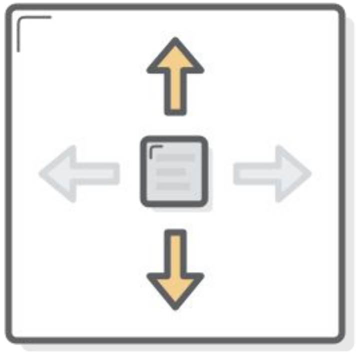
```
.links{
  border: 1px solid #fff;  /* For debugging */
  display: flex;
  justify-content: flex-end;
}

.login{
  margin-left: 20px;
}
```

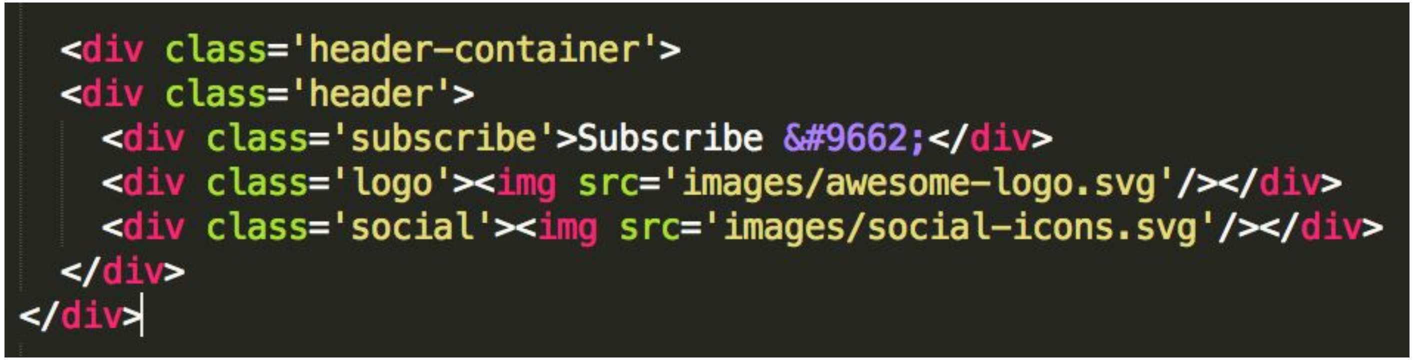

EXAMPLE.HTML
Aug 14, 2016
Sign Up
Login

## Cross-Axis (Vertical) Alignment

Flex containers can also define the vertical alignment of their items.

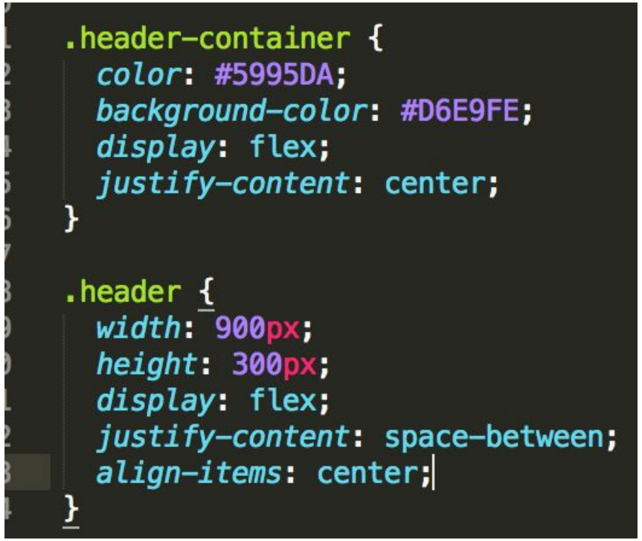






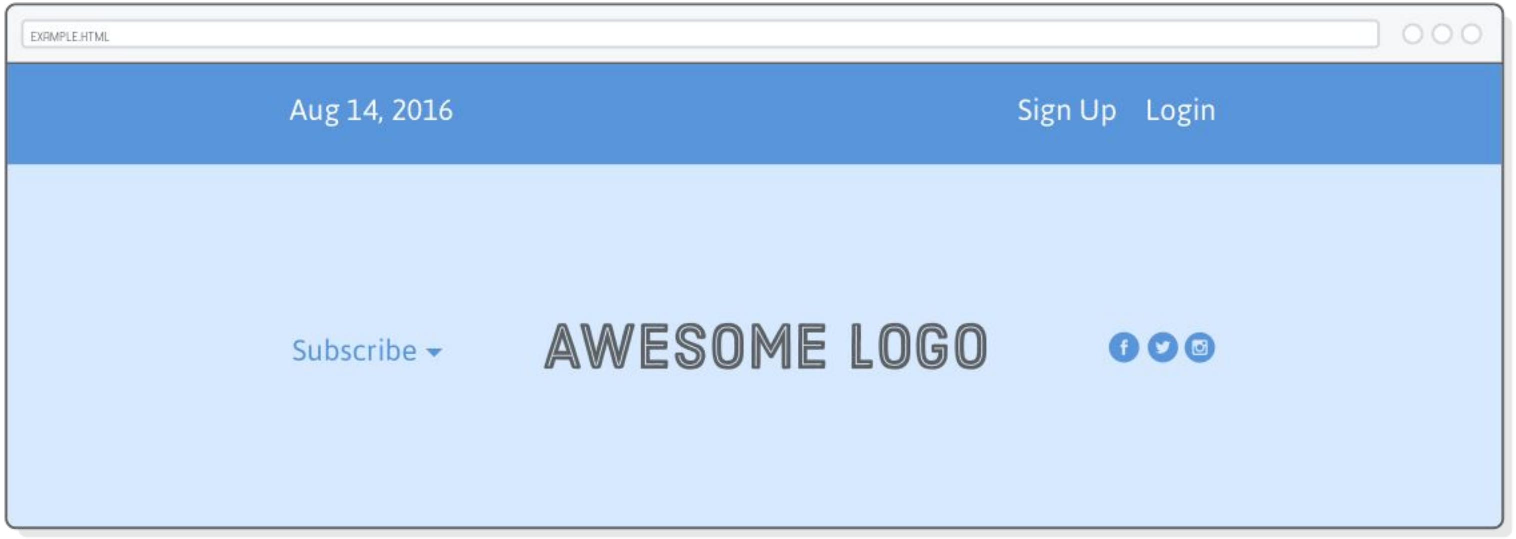
```html
  <div class='header-container'>
  <div class='header'>
    <div class='subscribe'>Subscribe &#9662;</div>
    <div class='logo'><img src='images/awesome-logo.svg'/></div>
    <div class='social'><img src='images/social-icons.svg'/></div>
  </div>
</div>
```

.html



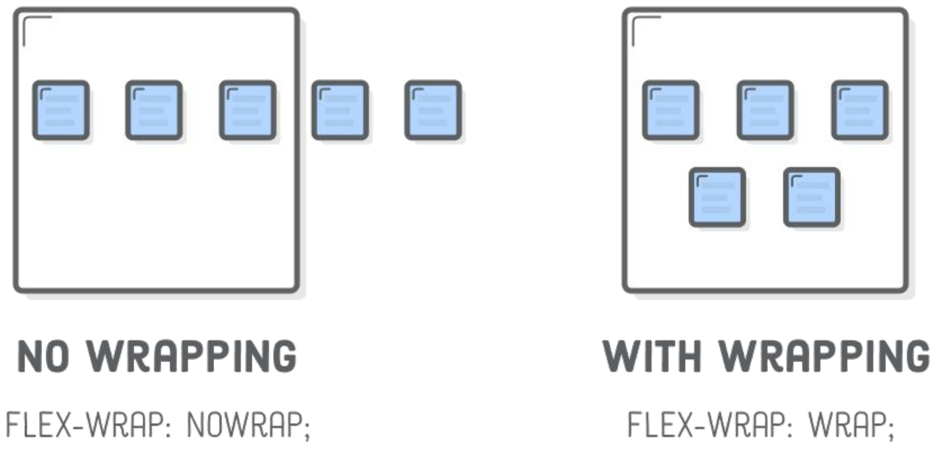
```css
.header-container {
  color: #5995DA;
  background-color: #D6E9FE;
  display: flex;
  justify-content: center;
}

.header {
  width: 900px;
  height: 300px;
  display: flex;
  justify-content: space-between;
  align-items: center;
}
```

.css

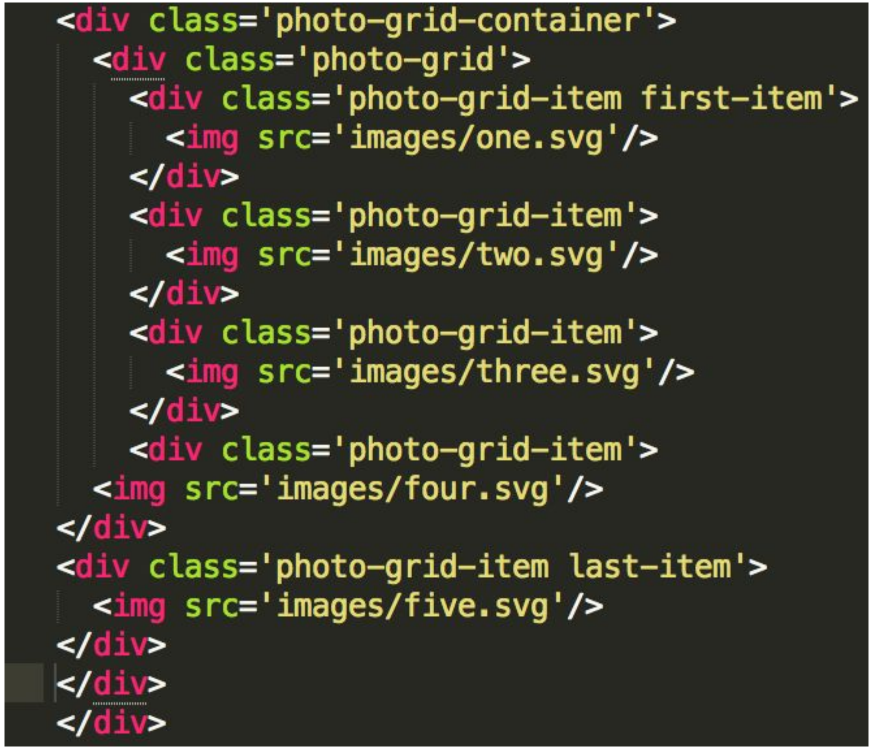

EXAMPLE.HTML
Aug 14, 2016
Sign Up
Login
Subscribe
AWESOME LOGO

## Wrapping flex items as a grid

Not only can it render items as a grid—it can change their alignment, direction, order, and size, too. To create a grid, we need the `flex-wrap` property.

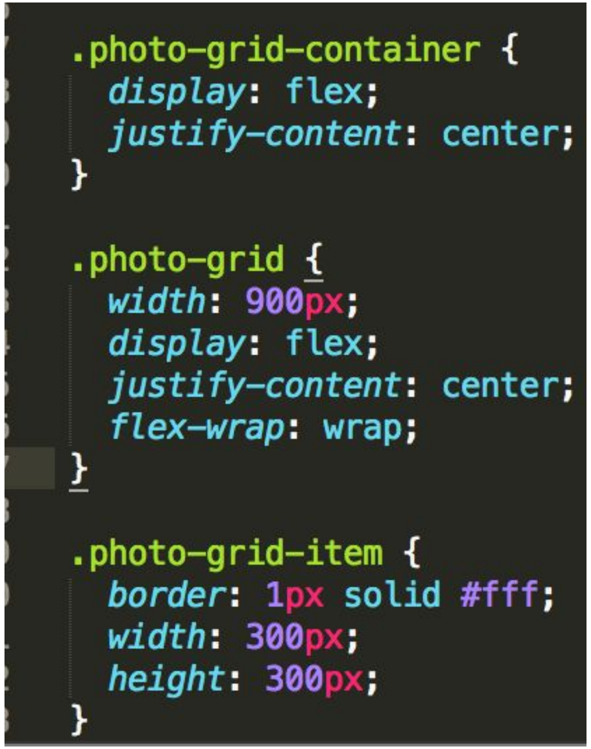


.html

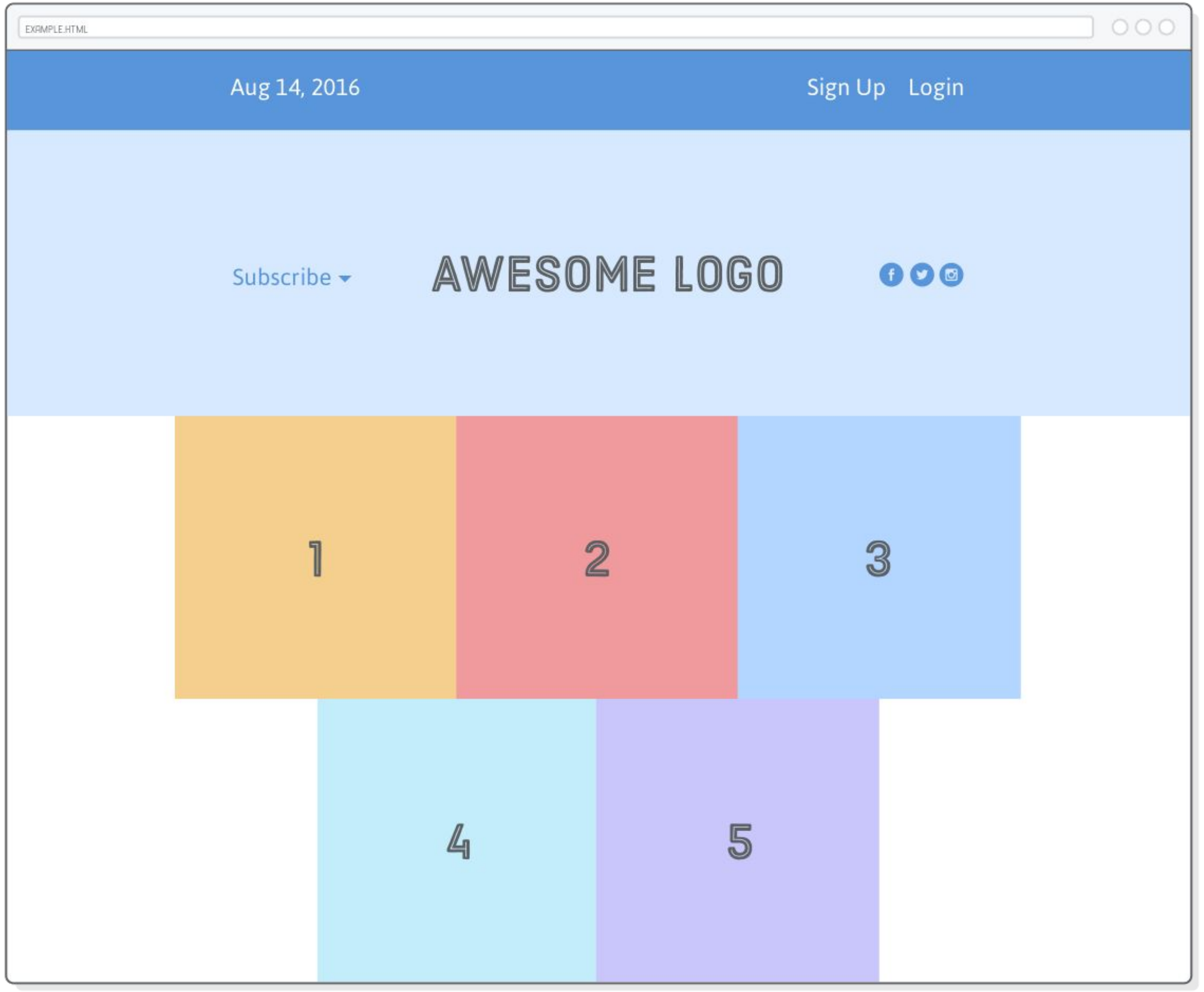


```html
<div class='photo-grid-container'>
  <div class='photo-grid'>
    <div class='photo-grid-item first-item'>
      <img src='images/one.svg'/>
    </div>
    <div class='photo-grid-item'>
      <img src='images/two.svg'/>
    </div>
    <div class='photo-grid-item'>
      <img src='images/three.svg'/>
    </div>
    <div class='photo-grid-item'>
  <img src='images/four.svg'/>
</div>
<div class='photo-grid-item last-item'>
  <img src='images/five.svg'/>
</div>
</div>
</div>
```

.css



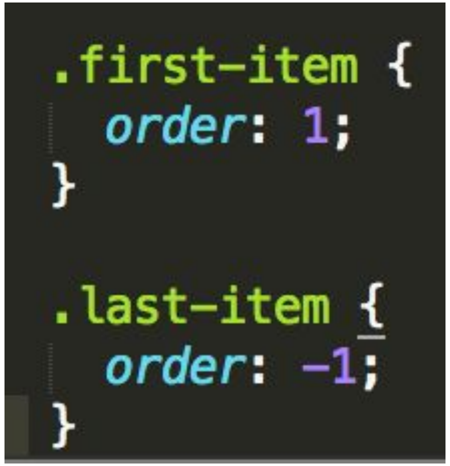
```css
.photo-grid-container {
  display: flex;
  justify-content: center;
}

.photo-grid {
  width: 900px;
  display: flex;
  justify-content: center;
  flex-wrap: wrap;
}

.photo-grid-item {
  border: 1px solid #fff;
  width: 300px;
  height: 300px;
}
```


EXAMPLE.HTML
Aug 14, 2016
Sign Up
Login
Subscribe
AWESOME LOGO
1
2
3
4
5

## Individual items - order

Adding an order property to a flex item defines its order in the container without affecting surrounding items. Its default value is 0, and increasing or decreasing it from there moves the item to the right or left, respectively.

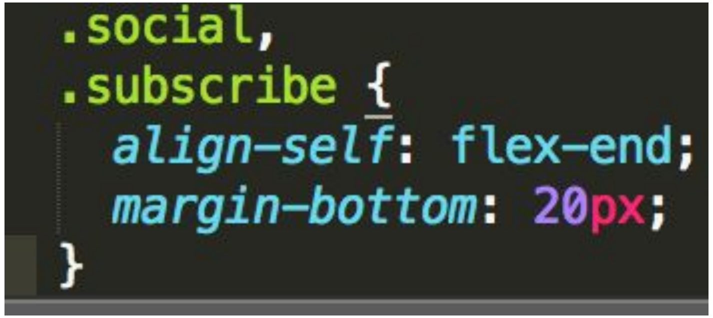
This can be used, for example, to swap order of the .first-item and .last-item elements in our grid.



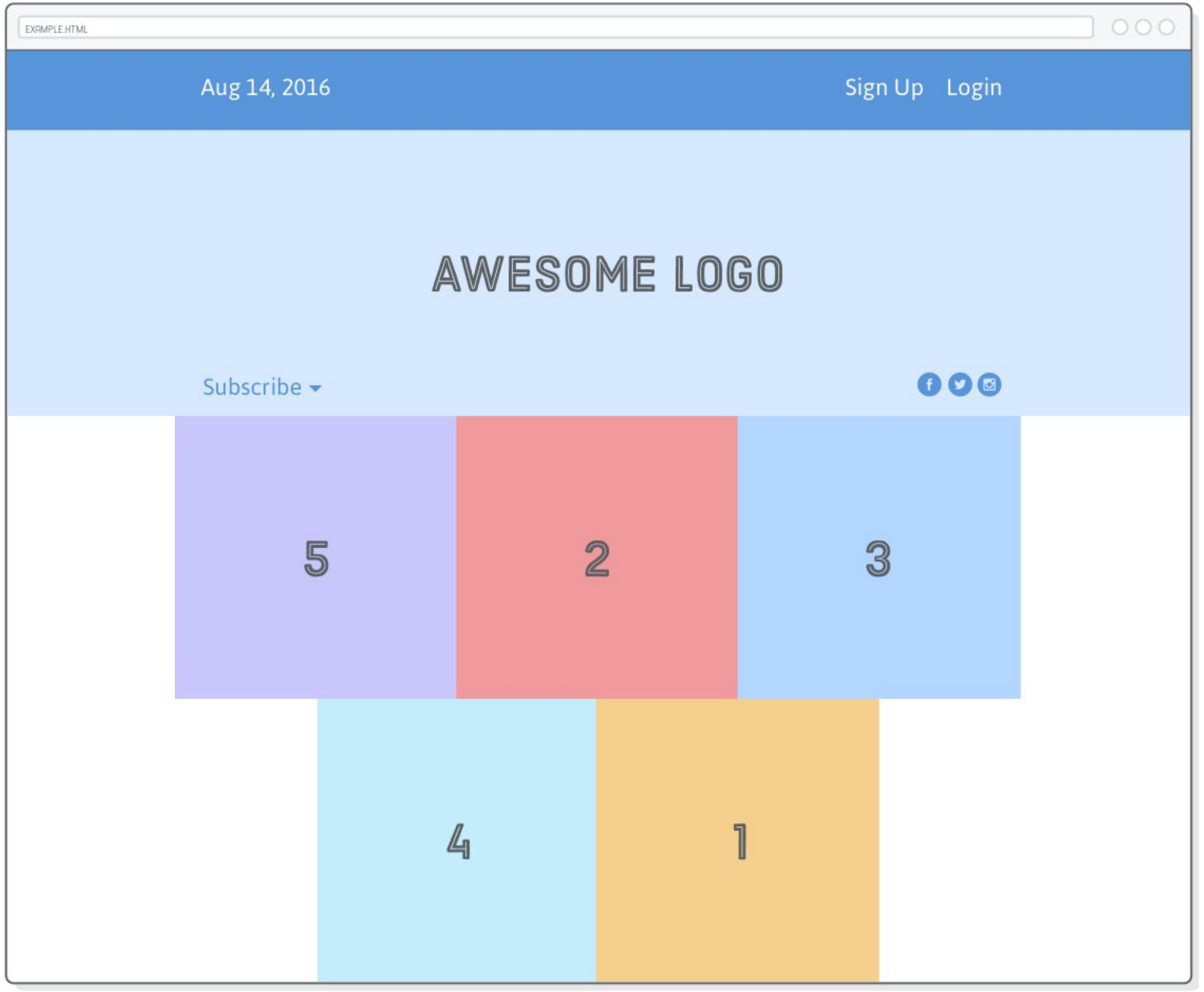
```
.first-item {
  order: 1;
}

.last-item {
  order: -1;
}
```

## Flex item alignment

We can do the same thing with vertical alignment. What if we want that **Subscribe** link and those social icons to go at the bottom of the header instead of the center? Align them individually! This is where the align-self property comes in. Adding this to a flex item overrides the align-items value from its container:

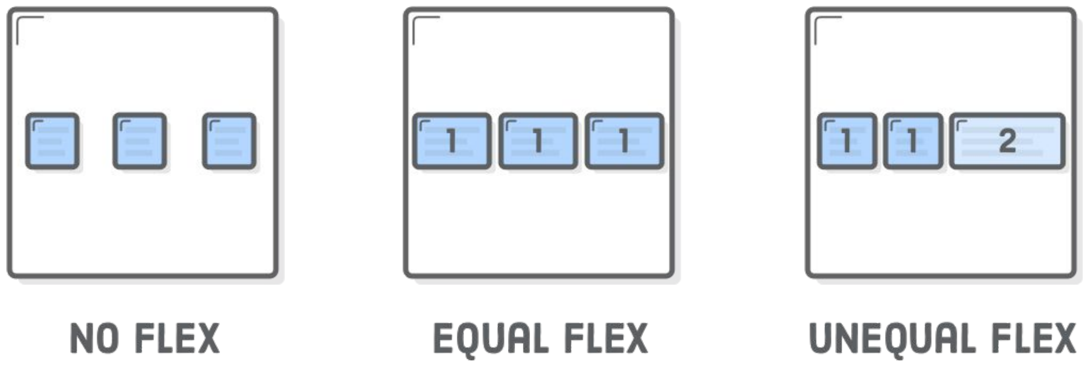


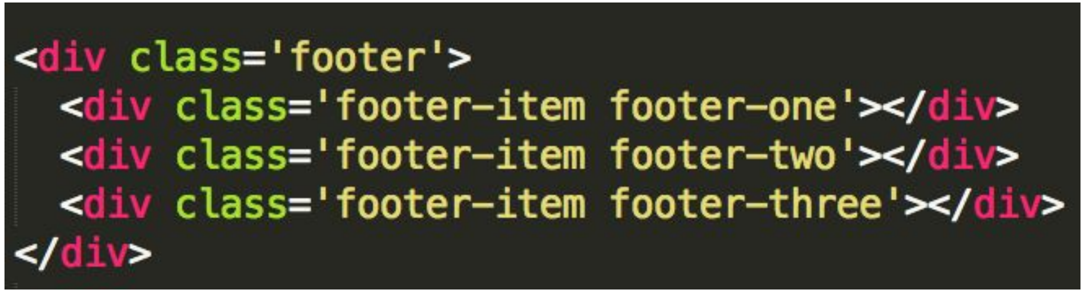
```
.social,
.subscribe {
  align-self: flex-end;
  margin-bottom: 20px;
}
```


EXAMPLE.HTML
Aug 14, 2016
Sign Up
Login
AWESOME LOGO
Subscribe
5
2
3
4
1

## Flexible items

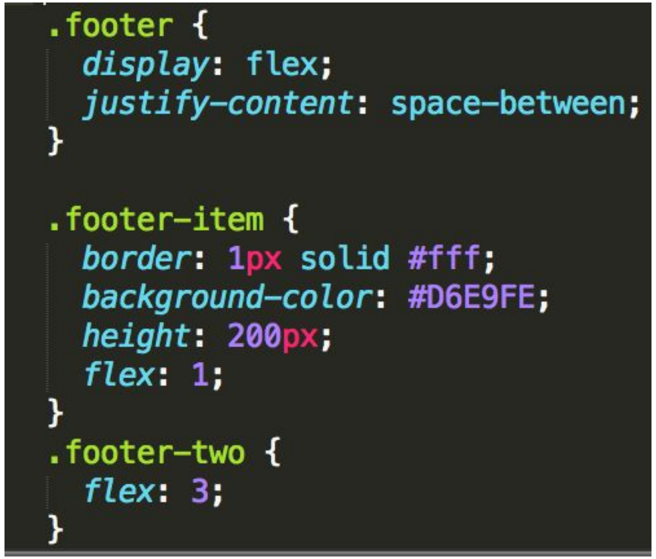
The flex property defines the width of individual items in a flex container. Or, more accurately, it allows them to have flexible widths. It works as a weight that tells the flex container how to distribute extra space to each item. For example, an item with a flex value of 2 will grow twice as fast as items with the default value of 1.





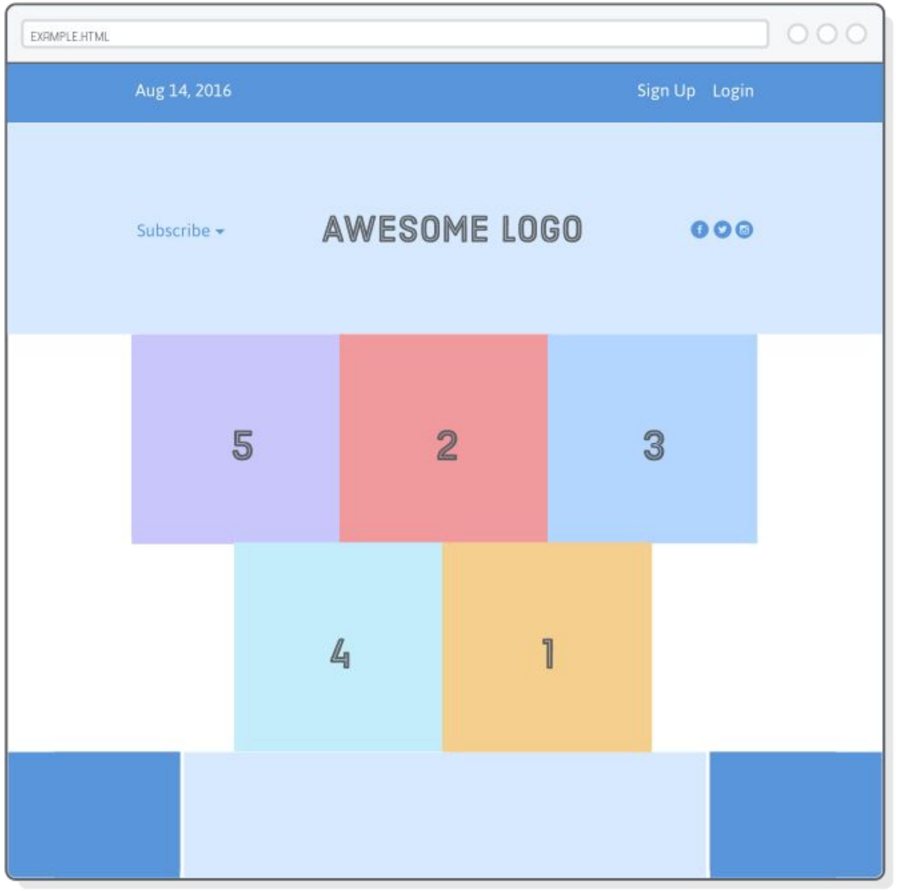
```html
<div class='footer'>
  <div class='footer-item footer-one'></div>
  <div class='footer-item footer-two'></div>
  <div class='footer-item footer-three'></div>
</div>
```

html



```css
.footer {
  display: flex;
  justify-content: space-between;
}

.footer-item {
  border: 1px solid #fff;
  background-color: #D6E9FE;
  height: 200px;
  flex: 1;
}
.footer-two {
  flex: 3;
}
```

css


EXAMPLE.HTML
Aug 14, 2016
Sign Up
Login
Subscribe
AWESOME LOGO
5
2
3
4
1