Layout and Flexbox

Today's schedule

Today:

- Wrap up box model
- Debugging with Chrome Inspector
- **Case study**: Squarespace Layout
  - Flex box
  - Misc helpful CSS

Quick review

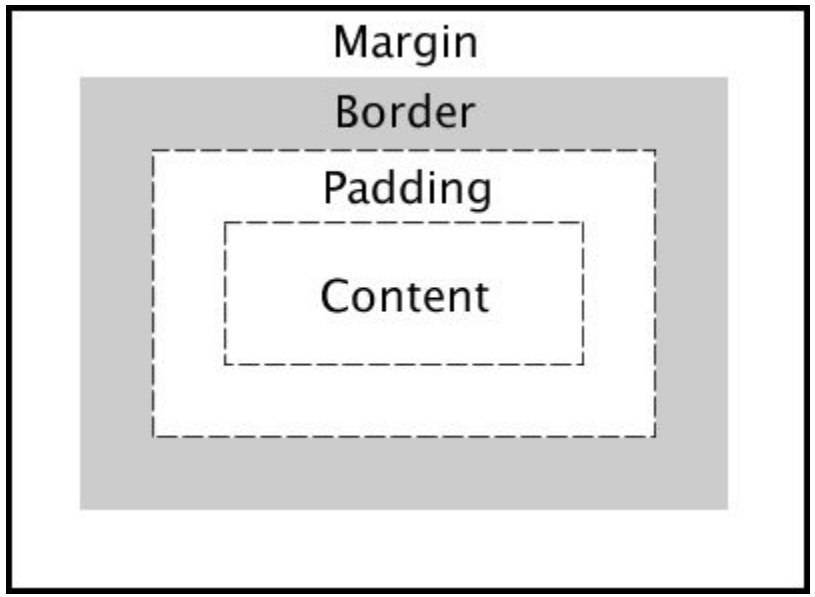
Selector summary

Example	Description
`p`	All `<p>` elements
`.abc`	All elements with the `abc class`, i.e. `class="abc"`
`#abc`	Element with the `abc id`, i.e. `id="abc"`
`p.abc`	`<p>` elements with `abc class`
`p#abc`	`<p>` element with `abc id` (`p` is redundant)
`div strong`	`<strong>` elements that are descendants of a `<div>`
`h2, div`	`<h2>` elements and `<div>`s

The CSS Box Model

Every element is composed of 4 layers:

- the element's content
- the **`border`** around the element's content
- **`padding`** space between the content and border (inside)
- a **`margin`** clears the area around border (outside)



`<div>`s look a little squished

When we add a border to multiple divs, they sit flush against each other:

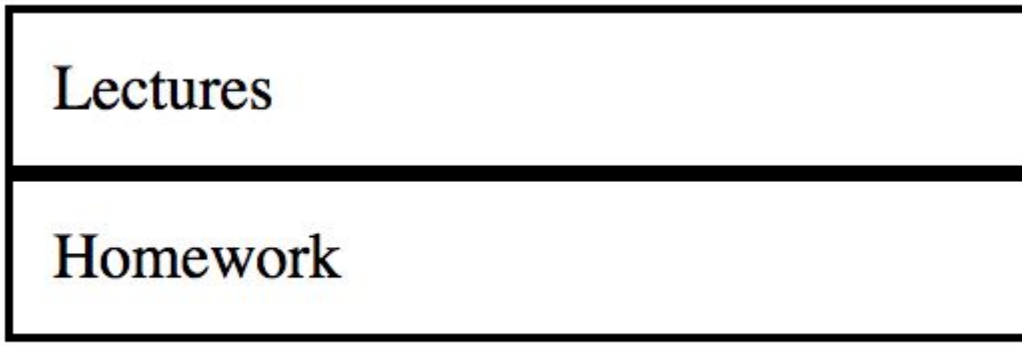
HTML

```
<div>
  Lectures
</div>
<div>
  Homework
</div>
```

CSS

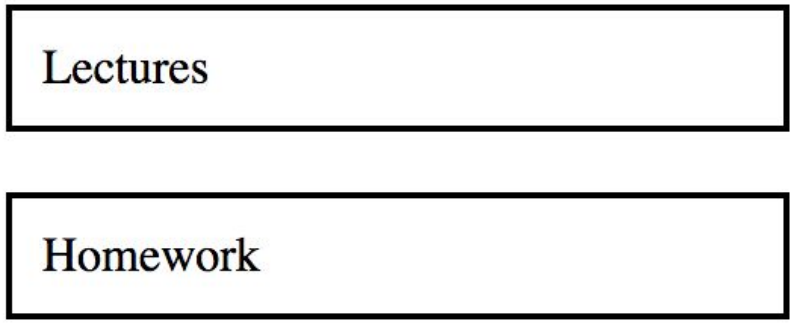
```
div {
  border: 2px solid black;
  padding: 10px;
}
```

Q: How do we add space between multiple elements?



margin

```
div {
  margin: 20px;
  padding: 10px;
  border: 2px solid black;
}
```



`margin` is the space between the border and other elements.

- Can specify `margin-top`, `margin-bottom`, `margin-left`, `margin-right`
- There's also a shorthand:

```
margin: 2px 4px 3px 1px;  <- top|right|bottom|left
margin: 10px 2px;         <- top+bottom|left+right
```

Back where we left off!

margin

Actually, why doesn't this:

```
div {
  margin: 20px;
  padding: 10px;
  border: 2px solid black;
}
```

Lectures

Homework

Look more like this?

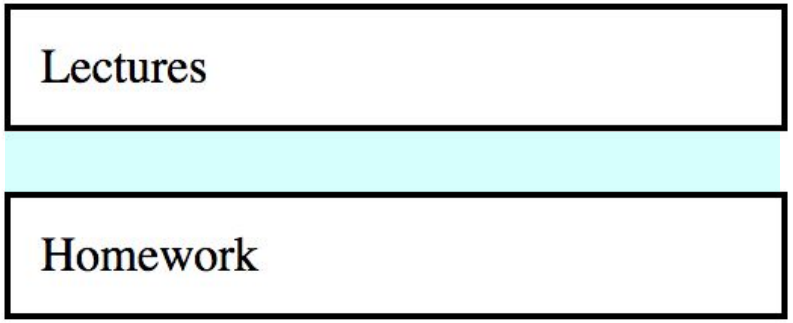
Lectures

Homework

margin

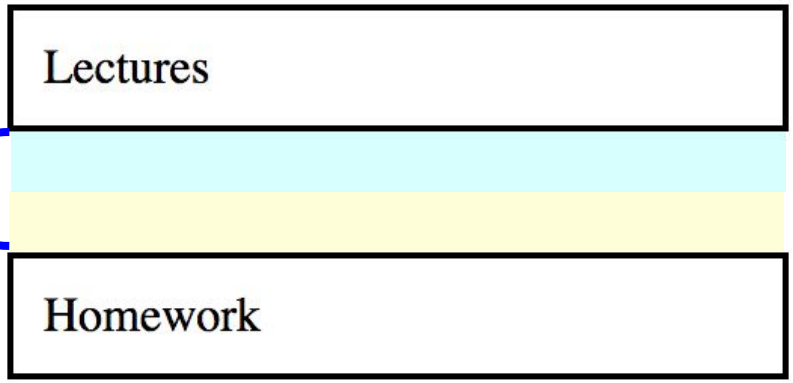
Actually, why doesn't this:

```
div {
  margin: 20px;
  padding: 10px;
  border: 2px solid black;
}
```



...look more like this?

20px margin-bottom +
20px margin top =
40px margin?



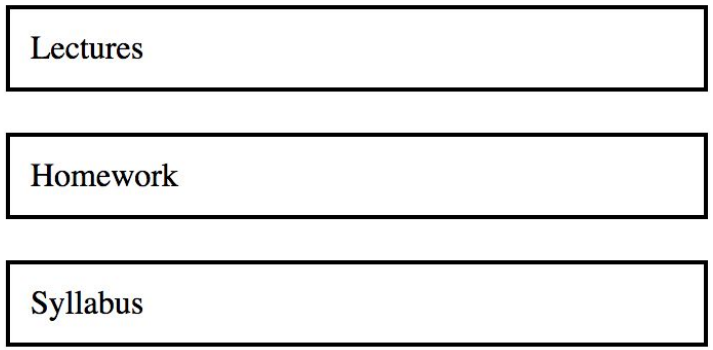
`margin` collapsing

Sometimes the top and bottom margins of block elements are combined ("collapsed") into a single margin.

- This is called **margin collapsing**

Generally if:

- The elements are siblings
- The elements are block-level (***not*** *inline-block*)



then they collapse into **max**(*Bottom Margin*, *Top Margin*).

(There are some exceptions to this, but when in doubt, use the Page Inspector to see what's going on.)

Negative margin

Margins can be negative as well.

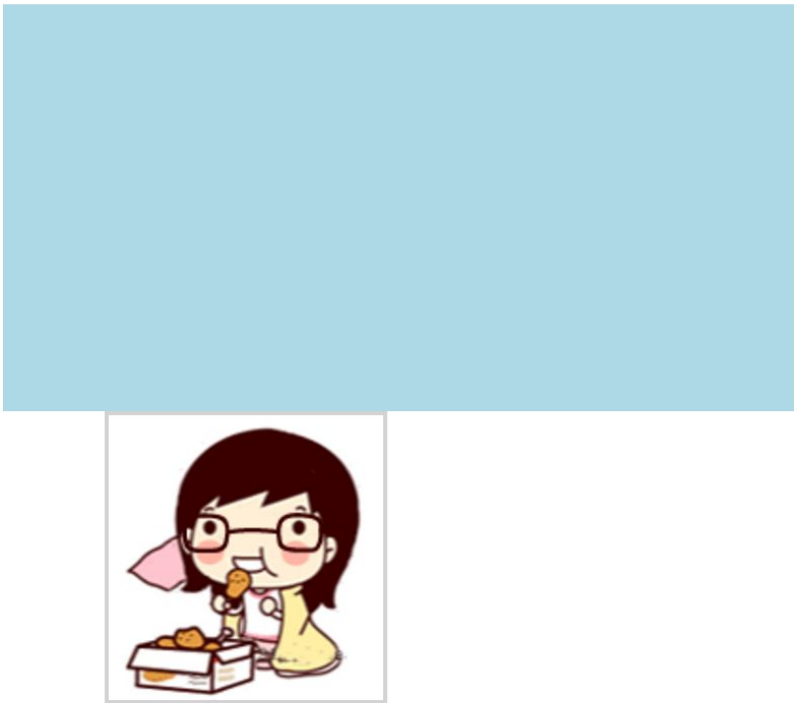
- **No negative margin on image:**

HTML

```
<div id="header"></div>
<div id="profile">
  <img src="https://s3-us-west-2.amazonaws.com/
</div>
```

CSS

```
#header {
  background-color: lightblue;
  height: 200px;
}

img {
  margin-left: 50px;
  height: 140px;
  border: 2px solid LIGHTGRAY;
}
```



Negative margin

Margins can be negative as well. (CodePen)

- `img { margin-top: -50px; }`

HTML

```
<div id="header"></div>
<div id="profile">
  <img src="https://s3-us-west-2.amazonaws.com/
</div>
```

CSS

```
#header {
  background-color: lightblue;
  height: 200px;
}

img {
  margin-top: -50px;
  margin-left: 50px;
  height: 140px;
  border: 2px solid LIGHTGRAY;
}
```

auto margins

If you set `margin-left` and `margin-right` to `auto`, you can center a block-level element (CodePen):

HTML

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Auto Margins</title>
  </head>
  <body>
    <div>
      This is a box of text.
    </div>
  </body>
</html>
```

CSS

```
div {
  margin-left: auto;
  margin-right: auto;

  border: 2px solid black;
  padding: 10px;
  width: 300px;
}
```

Box model for inline elements?

Q: Does the box model apply to inline elements as well?

Box model for inline elements?

Q: Does the box model apply to inline elements as well?

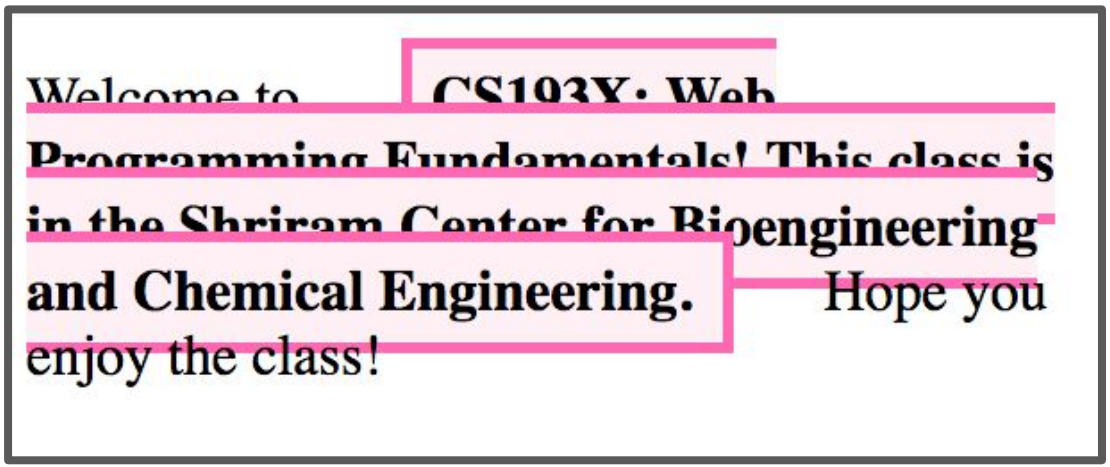
A: Yes, but the box is shaped differently.

CSS

```
strong {
  border: 3px solid hotpink;
  padding: 5px;
  margin: 25px;

  background-color: lavenderblush;
}
```

HTML

```
<p>
  Welcome to
  <strong>
    CS193X: Web Programming
  </strong>
  Hope you enjoy the class!
</p>
```



Box model for inline elements?

Q: Does the box model apply to inline elements as well?

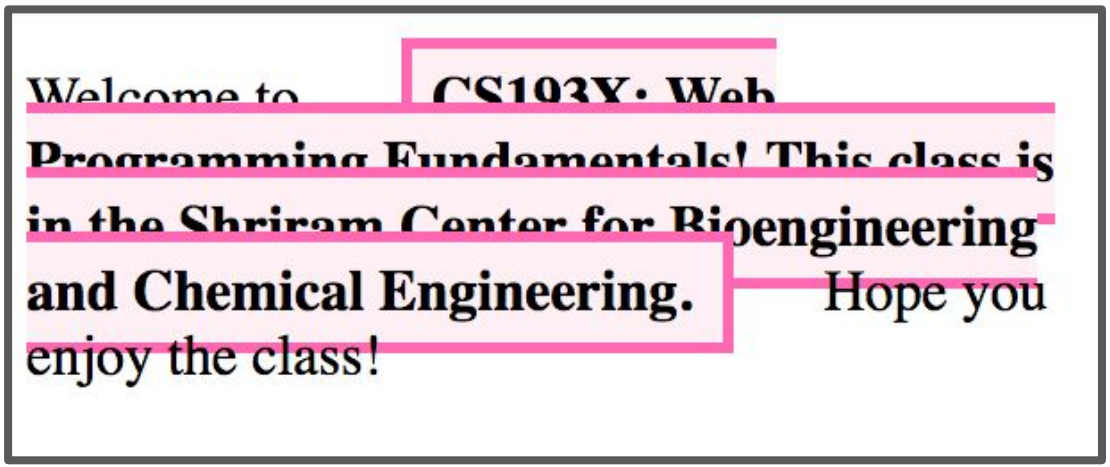
A: Yes, but the box is shaped differently.

CSS

```
strong {
  border: 3px solid hotpink;
  padding: 5px;
  margin: 25px;

  background-color: lavenderblush;
}
```

HTML

```
<p>
  Welcome to
  <strong>
    CS193X: Web Programming
  </strong>
  Hope you enjoy the class!
</p>
```

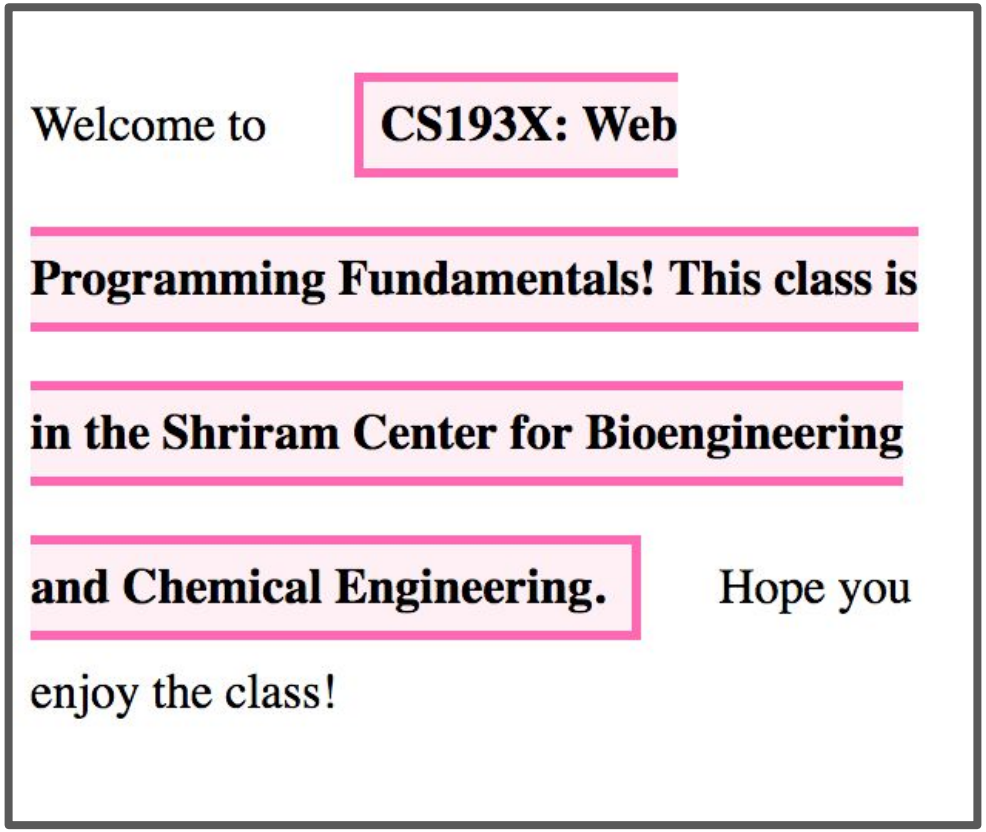


Let's change the line height to view this more clearly...

Inline element box model

CSS

```css
p {
  width: 300px;
  line-height: 50px;
}

strong {
  border: 3px solid hotpink;
  padding: 5px;
  margin: 25px;
  background-color: lavenderblush;
}
```

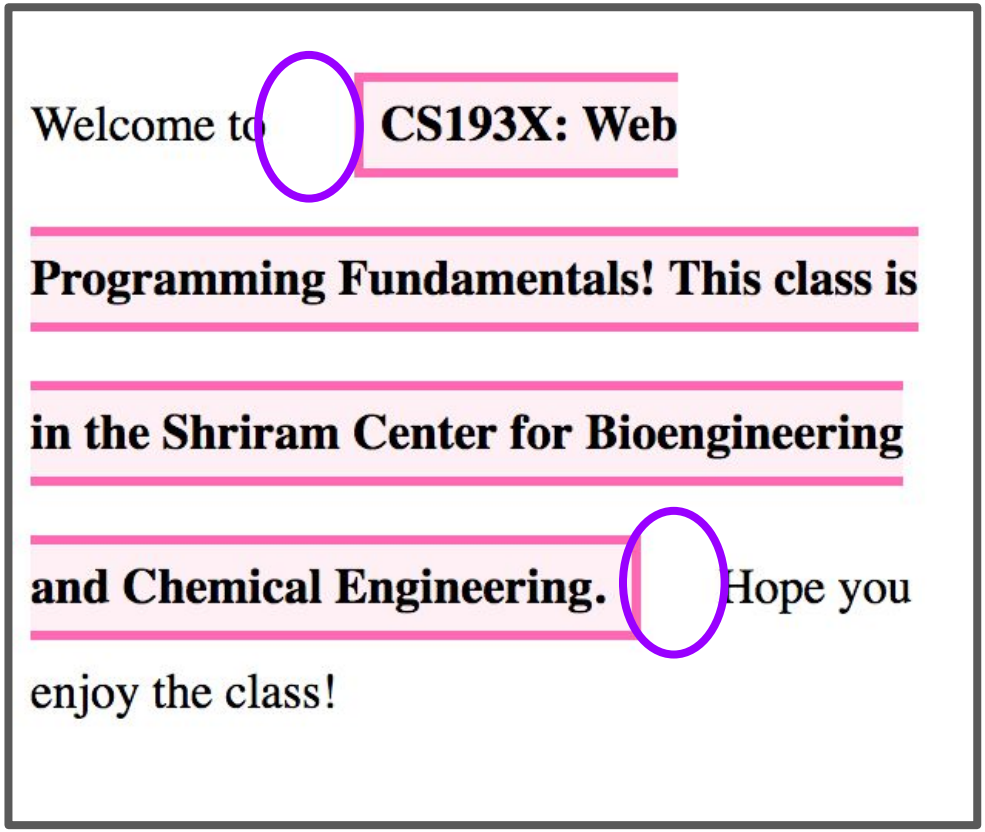


(Codepen)

Inline element box model

CSS

```
strong {
  border: 3px solid hotpink;
  padding: 5px;
  margin: 25px;
  line-height: 50px;
  background-color: lavenderblush;
}
```

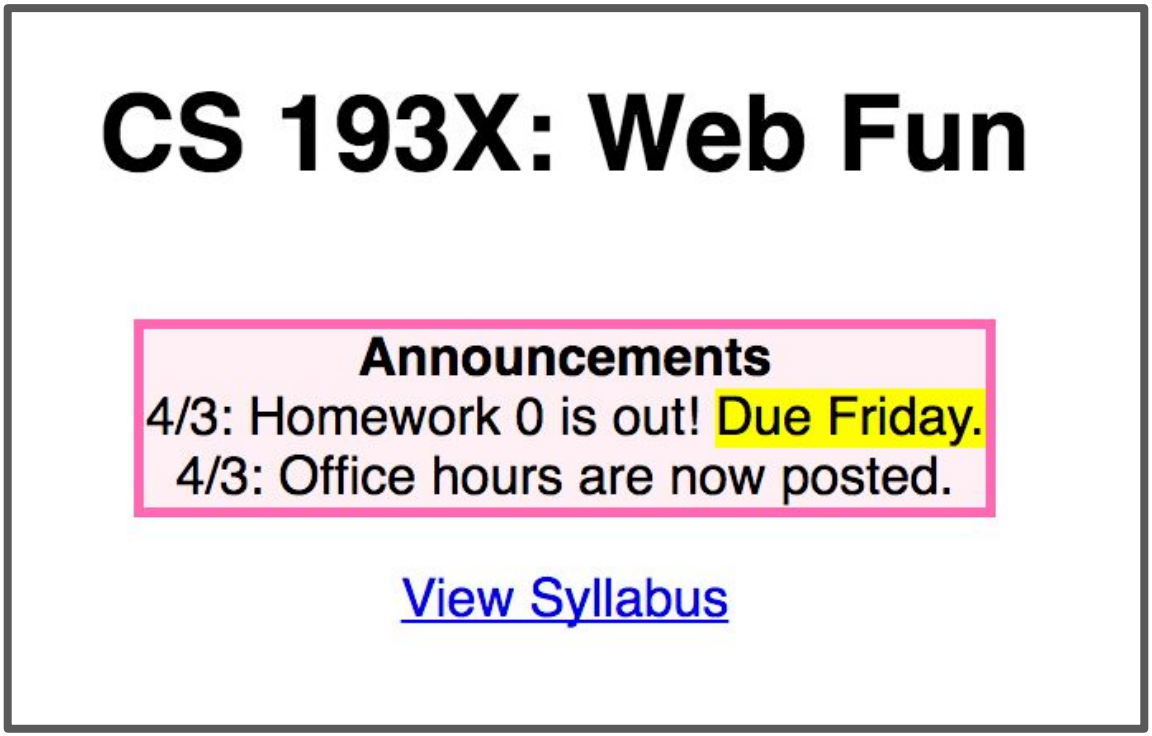


- **margin** is to the left and right of the inline element
  - `margin-top` and `margin-bottom` are **ignored**
- use **`line-height`** to manage space between lines

(Codepen)

The CSS Box Model

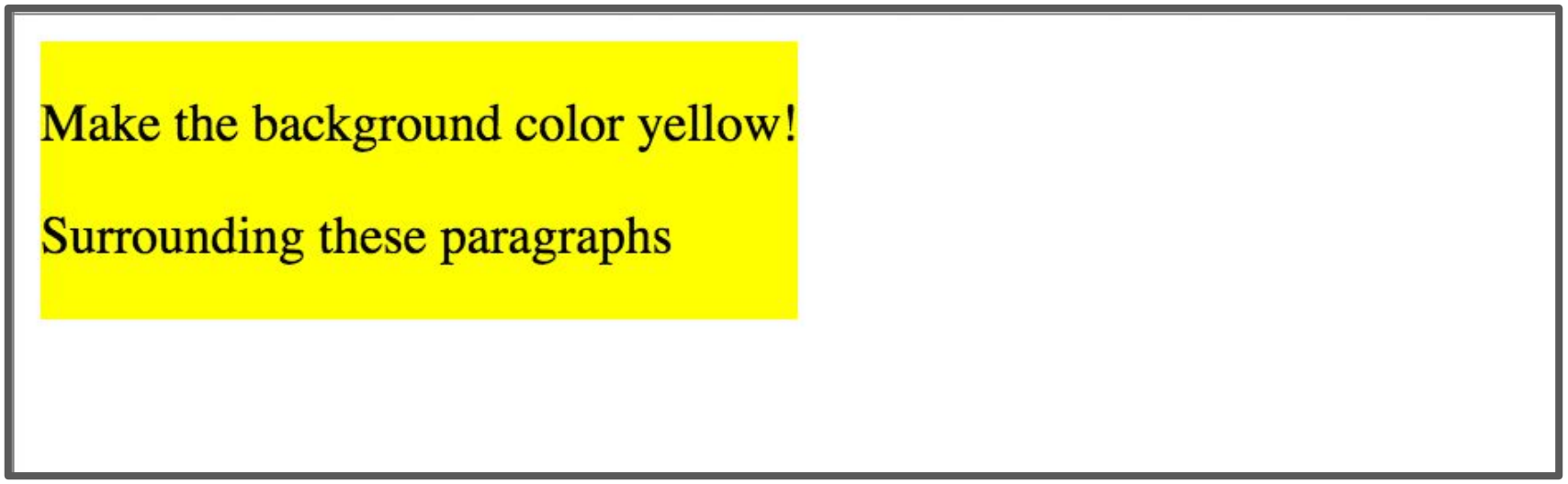
Let's revisit our Course web page example:



Q: What does this look like in the browser?

```
div {
  display: inline-block;
  background-color: yellow;
}
```

```
<body>
 <div>
  <p>Make the background color yellow!</p>
  <p>Surrounding these paragraphs</p>
 </div>
</body>
```



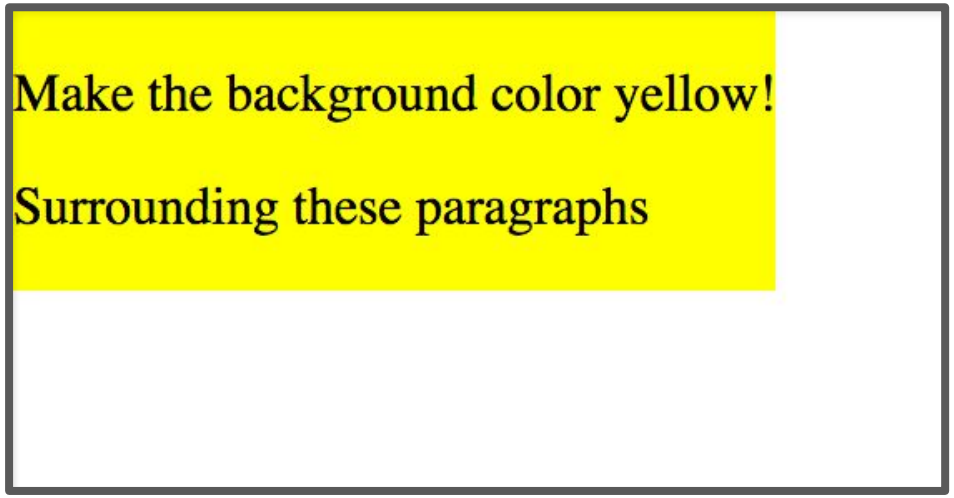
Q: Why is there a white space around the box?

We can use the browser's Page Inspector to help us figure it out!

`body` has a default `margin`

Set `body { margin: 0; }` to make your elements lay flush to the page.

```
body {
  margin: 0;
}

div {
  display: inline-block;
  background-color: yellow;
}
```



Recap so far...

We've talked about:

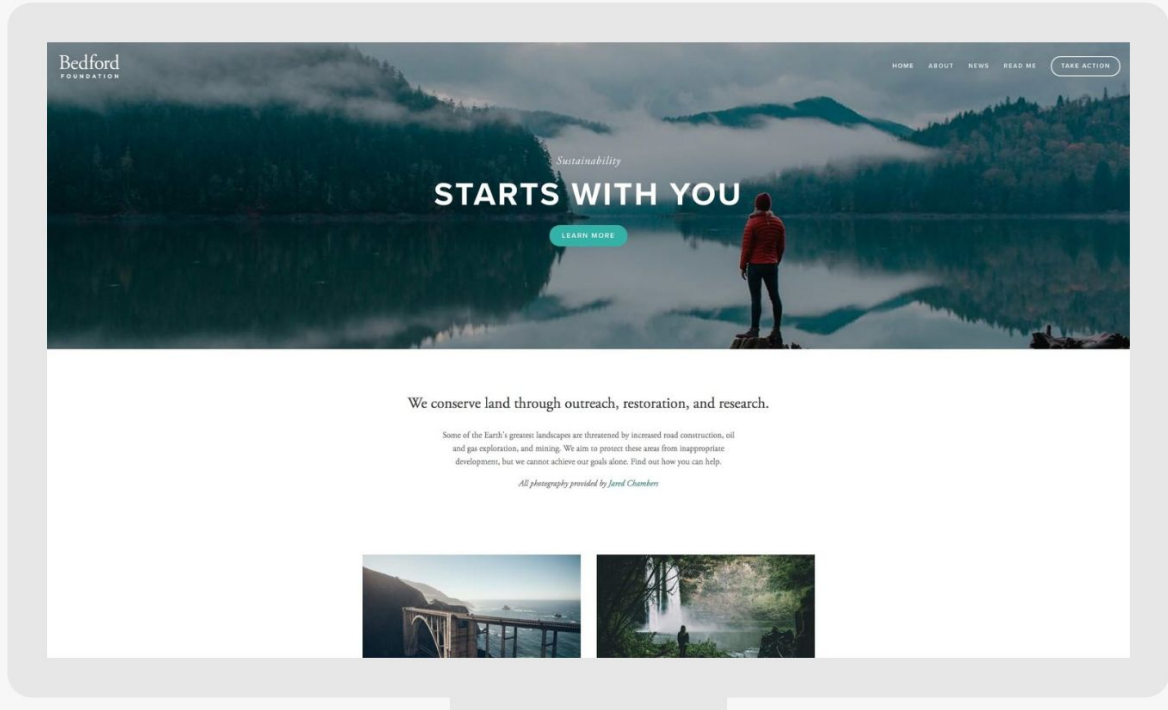
- **block vs inline** and the "natural" layout of the page, depending on the element type
- **`classes` and `ids`** and how to specify specific elements and groups of elements
- **`div` and `span`** and how to create generic elements
- **The CSS box model** and how every element is shaped like a box, with content -> padding -> border -> margin

Let's try making a "real" looking page!

Layout exercise

Squarespace template

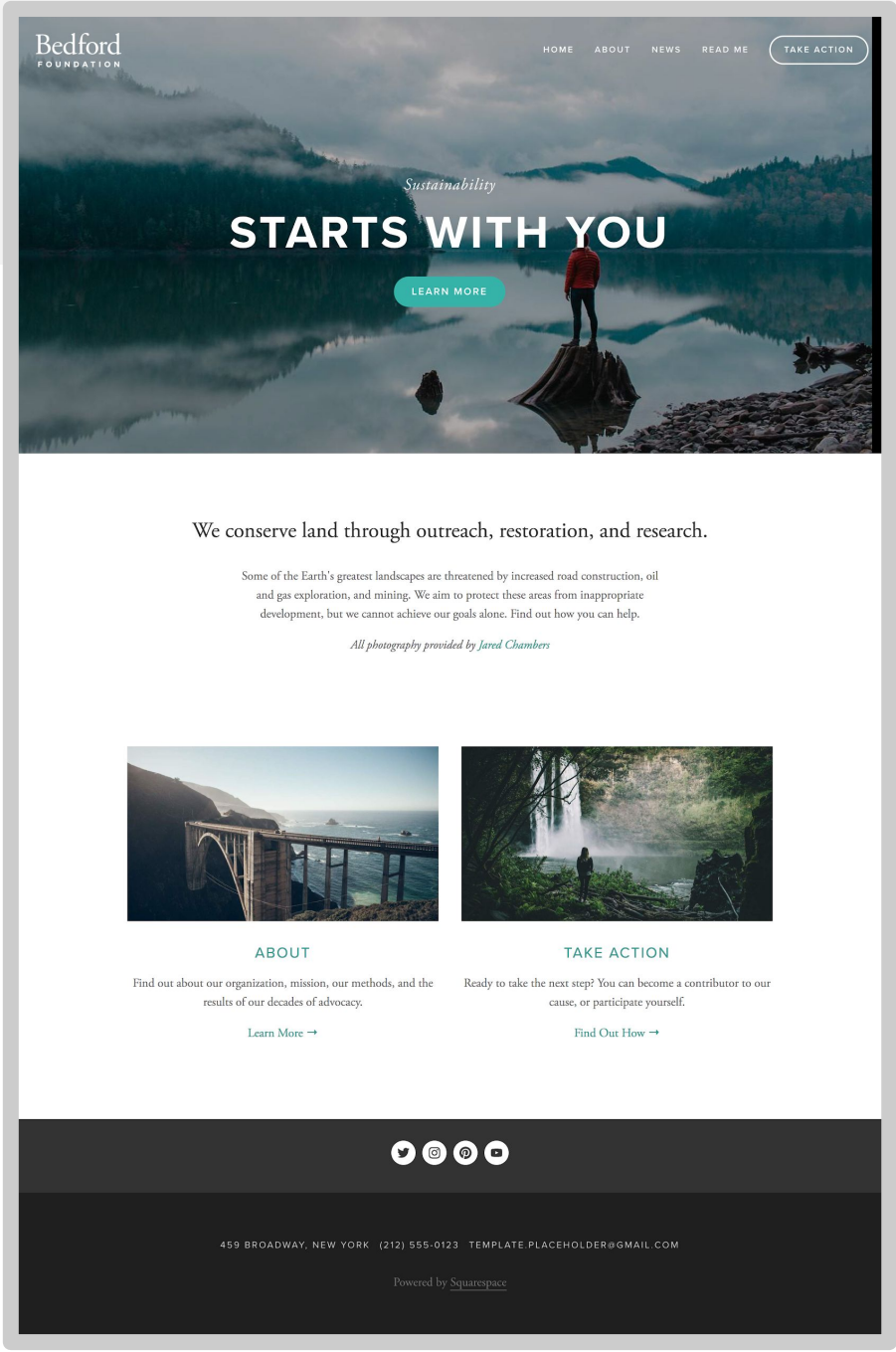
Squarespace's most popular template looks like this:



Q: Do we know enough to make something like that?

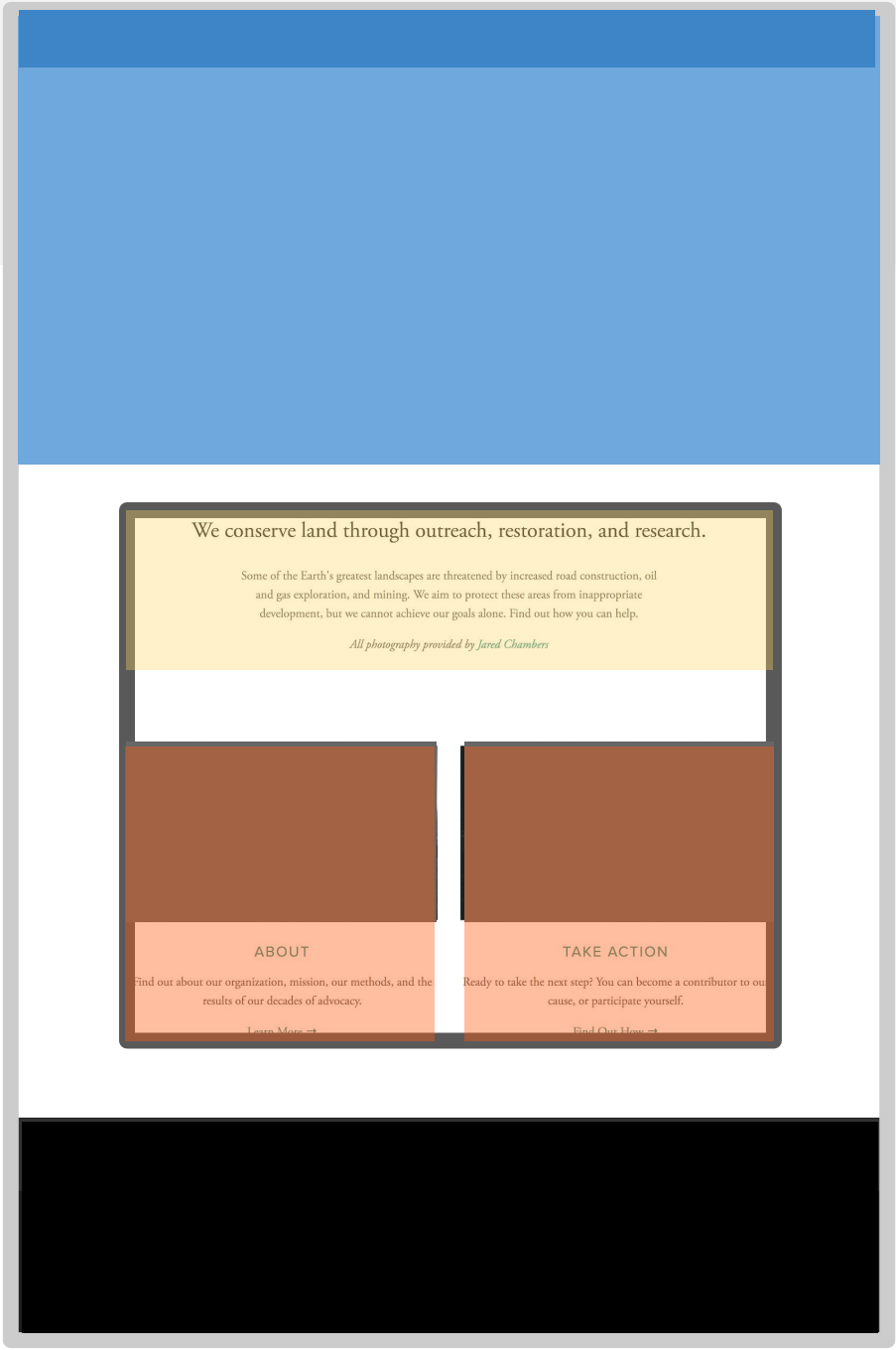
Basic shape

Begin visualizing the layout in terms of boxes:



Basic shape

Begin visualizing the layout in terms of boxes:

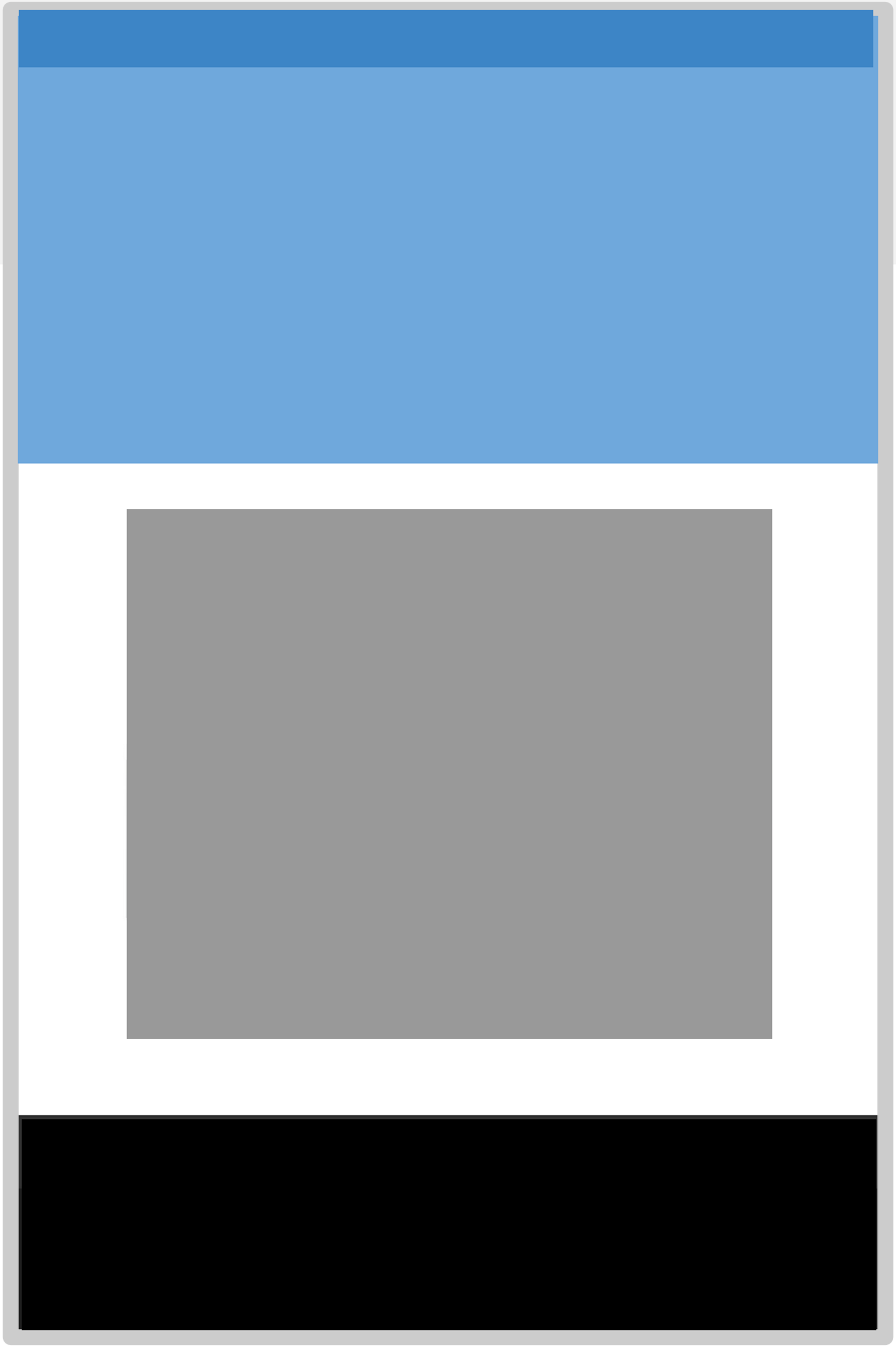


Basic shape

Begin visualizing the layout in terms of boxes:

Let's first try making this layout!

Codepen Link



Content Sectioning elements

Name	Description
`<p>`	Paragraph (mdn)
`<h1>-<h6>`	Section headings (mdn)
`<article>`	A document, page, or site (mdn) This is usually a root container element after `body`.
`<section>`	Generic section of a document (mdn)
`<header>`	Introductory section of a document (mdn)
`<footer>`	Footer at end of a document or section (mdn)
`<nav>`	Navigational section (mdn)

These elements do not "do" anything; they are basically more descriptive `<div>`s. Makes your HTML more readable. See MDN for more info.

Content Sectioning elements

Name	Description
`<p>`	Paragraph (mdn)
`<h1>-<h6>`	Section headings (mdn)
`<article>`	A document, page, or site (mdn) This is usu
`<section>`	Generic
`<header>`	Introduc
`<footer>`	Footer a
`<nav>`	Navigati

Prefer these elements to `<div>` when it makes sense!

These elements do not "do" anything, they are basically more descriptive `<div>`s. Makes your HTML more readable. See MDN for more info.

Header

Navbar:

- Height: 75px
- Background: royalblue
- <nav>

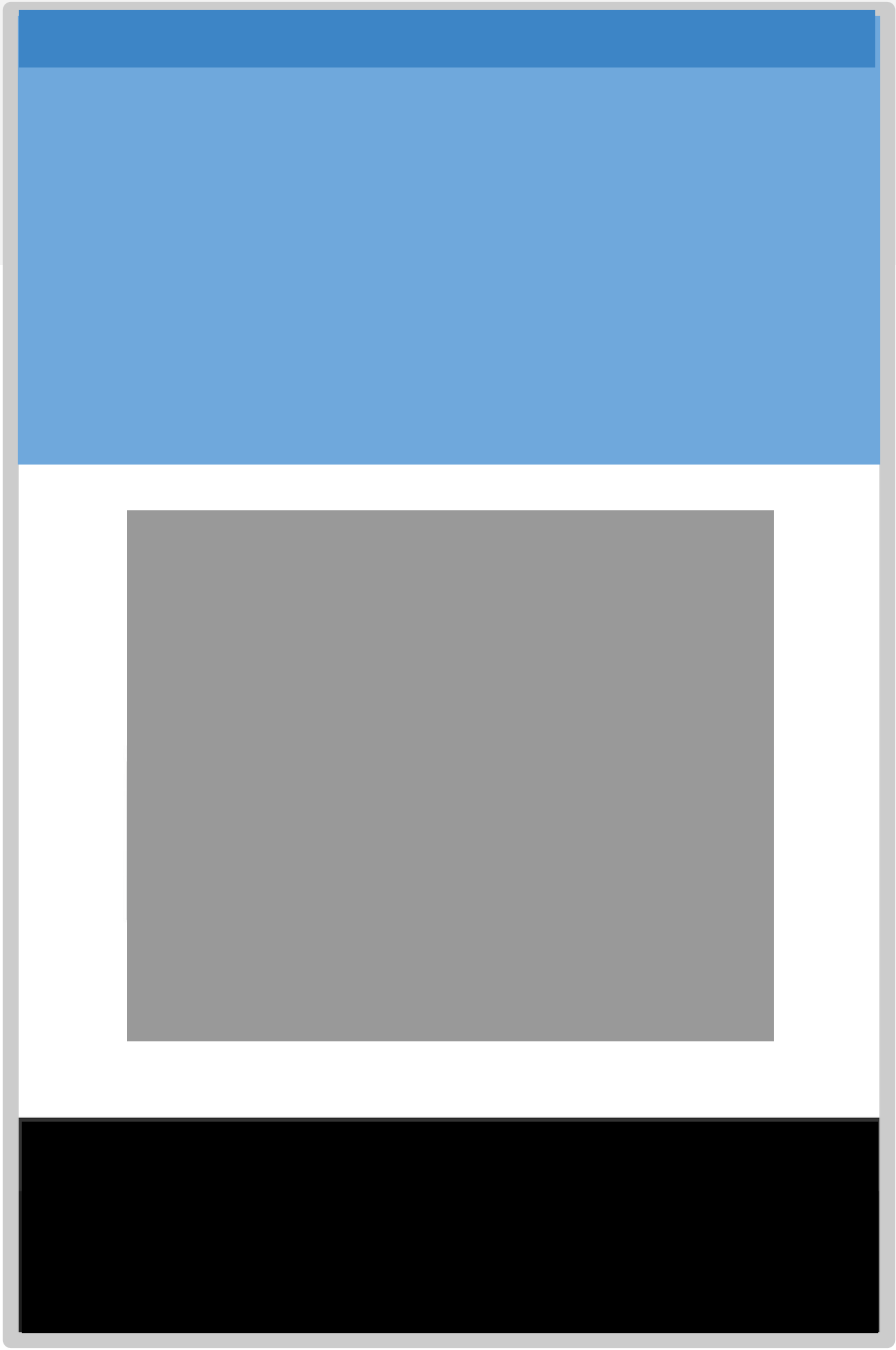
Header:

- Height: 400px;
- Background: lightskyblue
- <header>

Main section

Gray box:

- Surrounding space: 75px above and below; 100px on each side
- Height: 500px
- Background: gray
- `<section>`



Footer

Footer:

- Height: 100px
- Background: Black
- `<footer>`

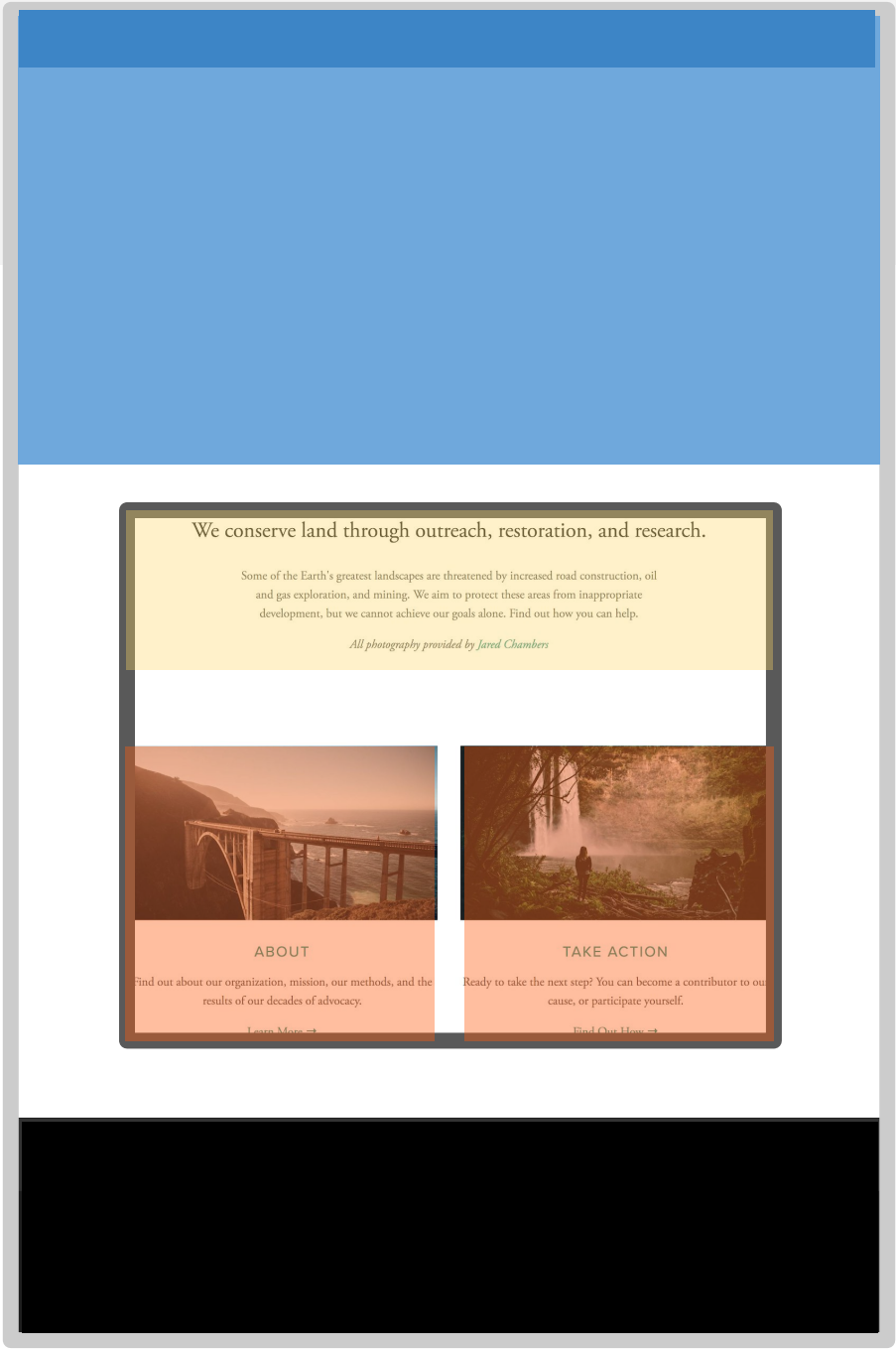
Main contents

Yellow paragraph:

- Height: 200px
- Background: khaki
- Space beneath: 75px
- `<p>`

Orange box:

- Height: 400px;
- Width: 48% of the parent's width, with space in between
- Background: tomato
- `<div>`

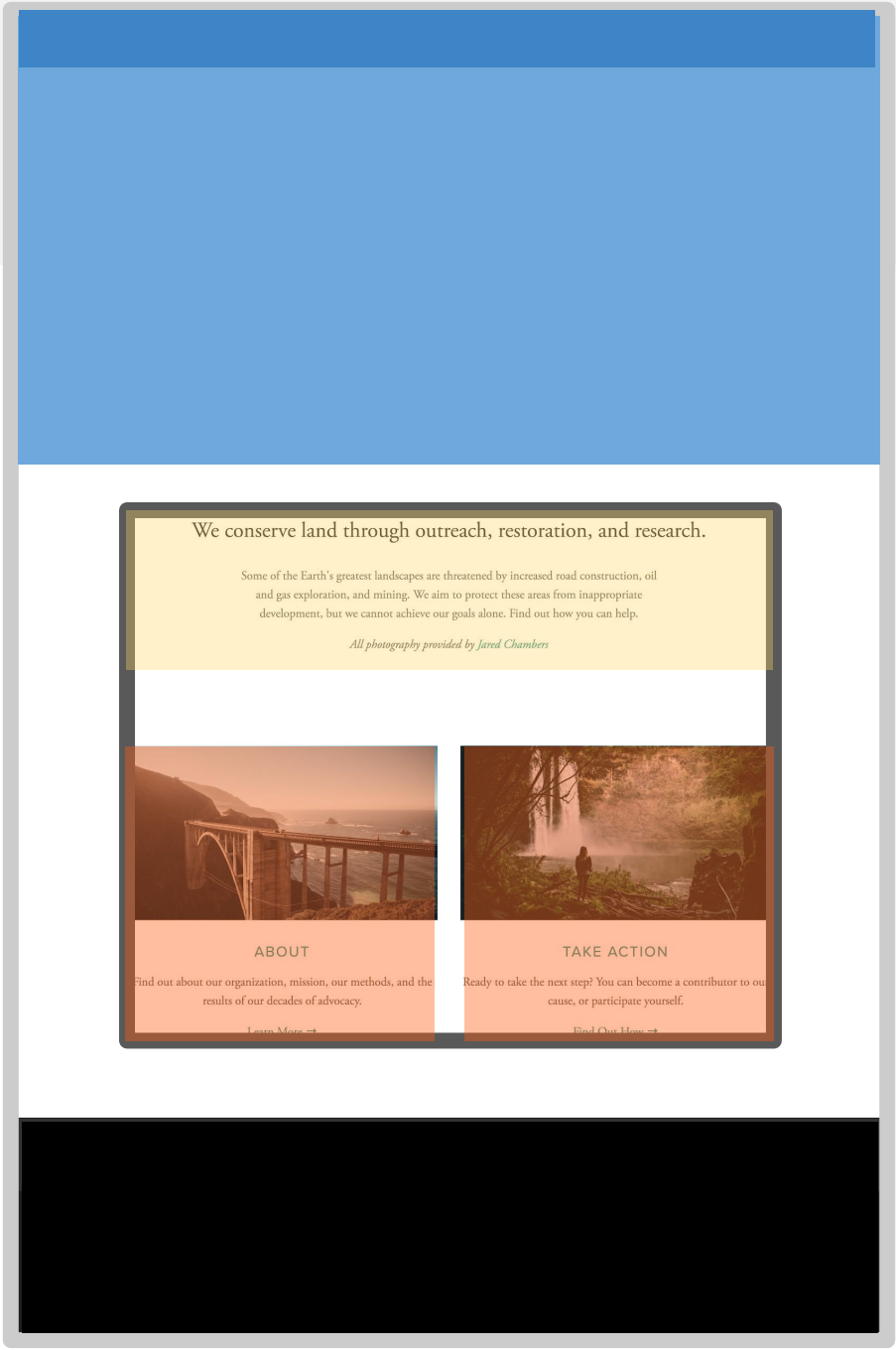


Main contents

Orange box:

- Height: 400px;
- Width: 48% of the parent's width, with space in between
- Background: tomato
- `<div>`

This is where we get stuck.



Flexbox

CSS layout so far

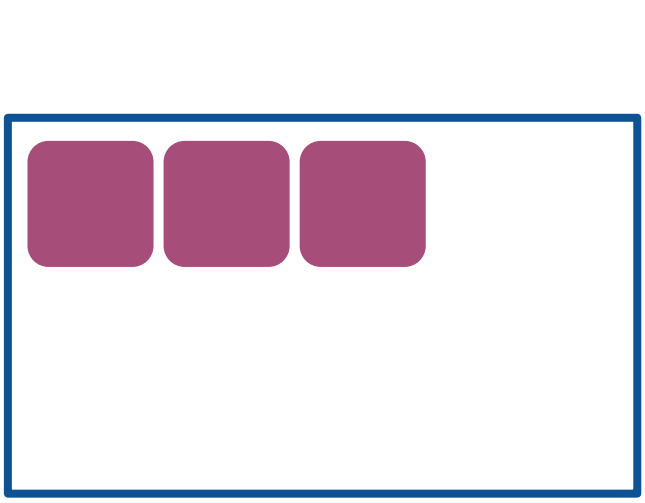
Block layout:
Laying out large sections of a page

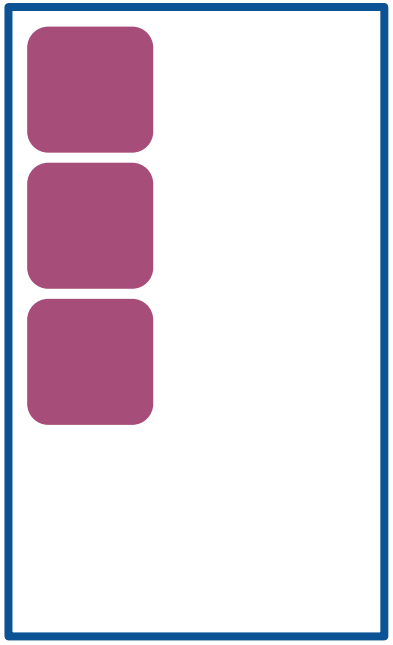
Inline layout:
Laying out text and other inline content within a section

Flex layout

To achieve more complicated layouts, we can enable a different kind of CSS layout rendering mode: **Flex layout.**

Flex layout defines a special set of rules for laying out items in rows or columns.

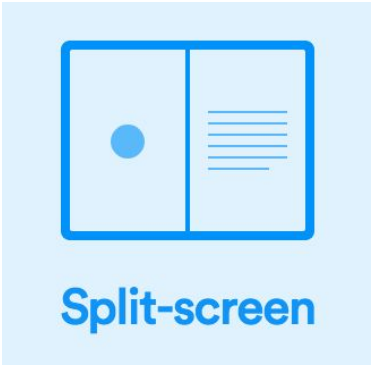


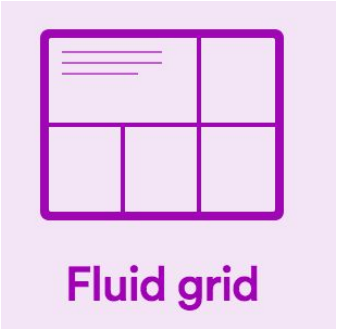


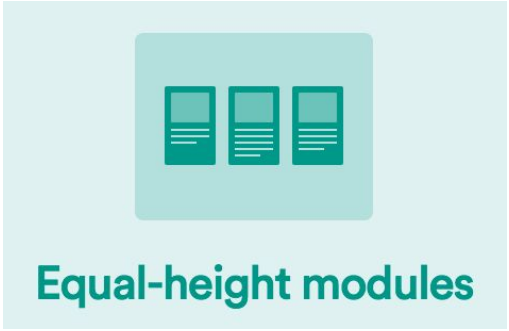
Flex layout

Flex layout solves all sorts of problems.

- Here are some examples of layouts that are easy to create with flex layout (and really difficult otherwise):





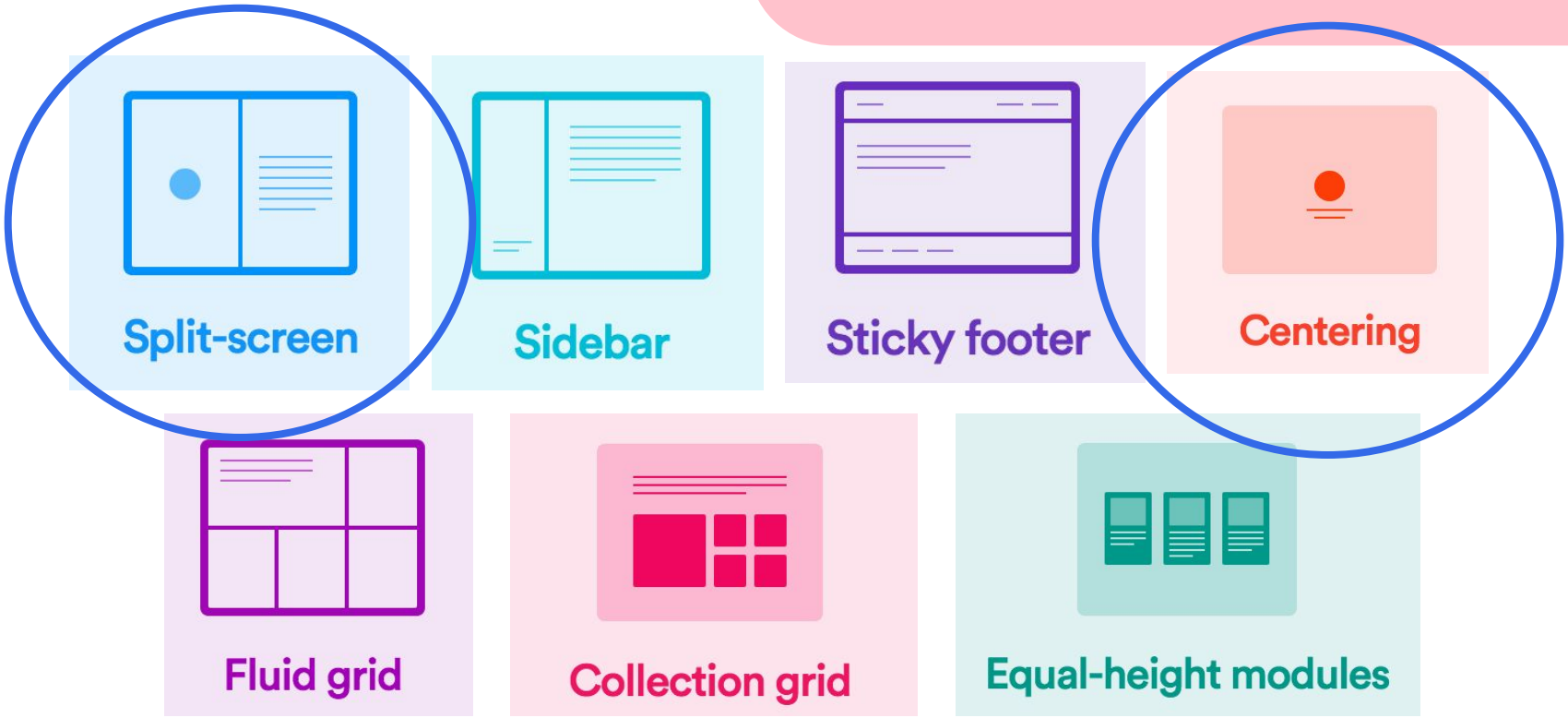


Flex layout

Flex layout solves all sorts o

- Here are some examples of la
  layout (and really difficult oth

But today we're only covering the basics!

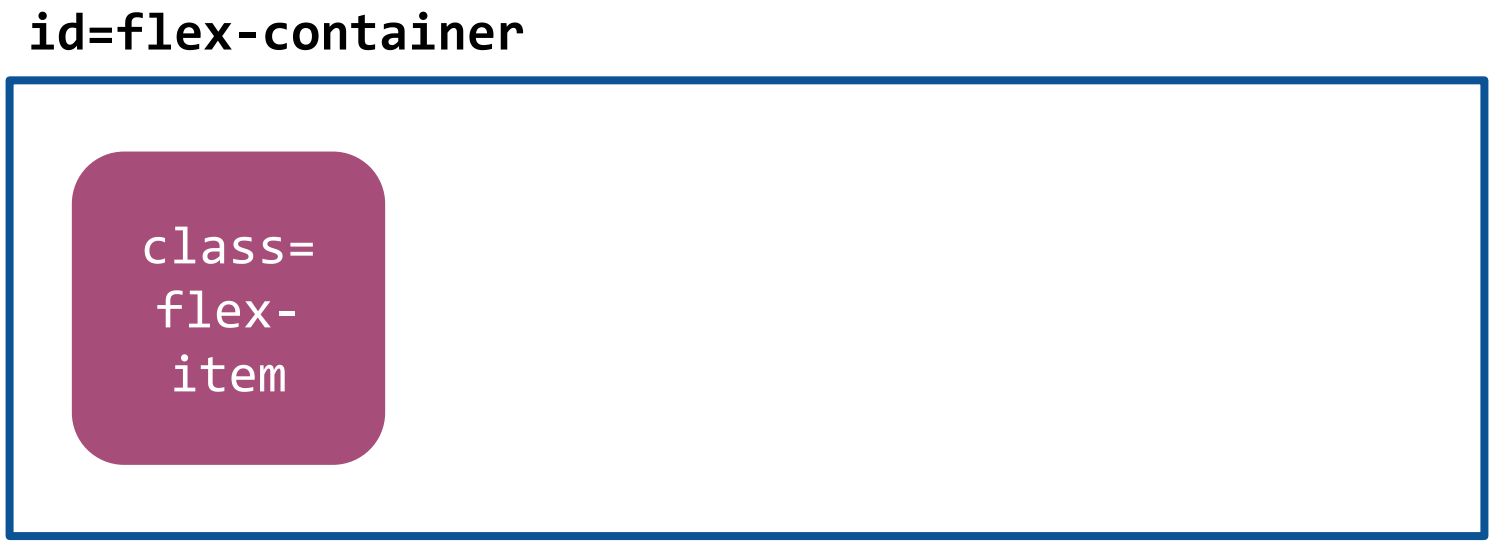


Flex basics

Flex layouts are composed of:

- A **Flex container**, which contains one or more:
  - **Flex item**(s)

You can then apply CSS properties on the **flex container** to dictate how the flex items are displayed.



Flex basics

To make an element a flex container, change `display`:

- Block container: `display: flex;` or
- Inline container: `display: inline-flex;`

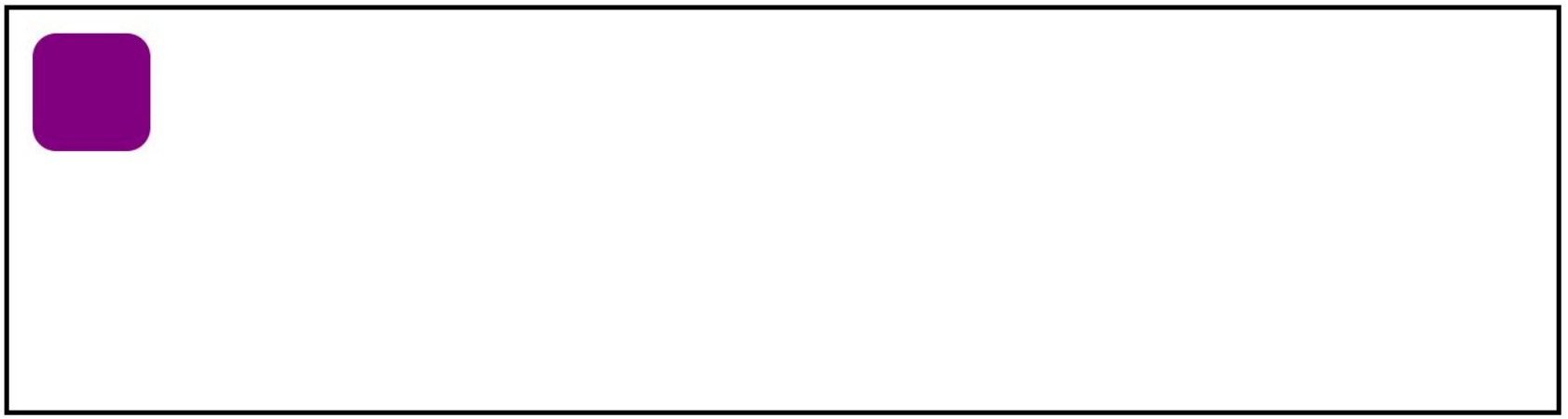
Follow along in Codepen

HTML

```html
<html>
  <head>
    <meta charset="utf-8">
    <title>Flexbox example</title>
  </head>
  <body>

    <div id="flex-container">
      <div class="flex-item"></div>
    </div>

  </body>
</html>
```

CSS

```css
#flex-container {
  display: flex;
  border: 2px solid black;
  padding: 10px;
  height: 150px;
}

.flex-item {
  border-radius: 10px;
  background-color: purple;
  height: 50px;
  width: 50px;
}
```

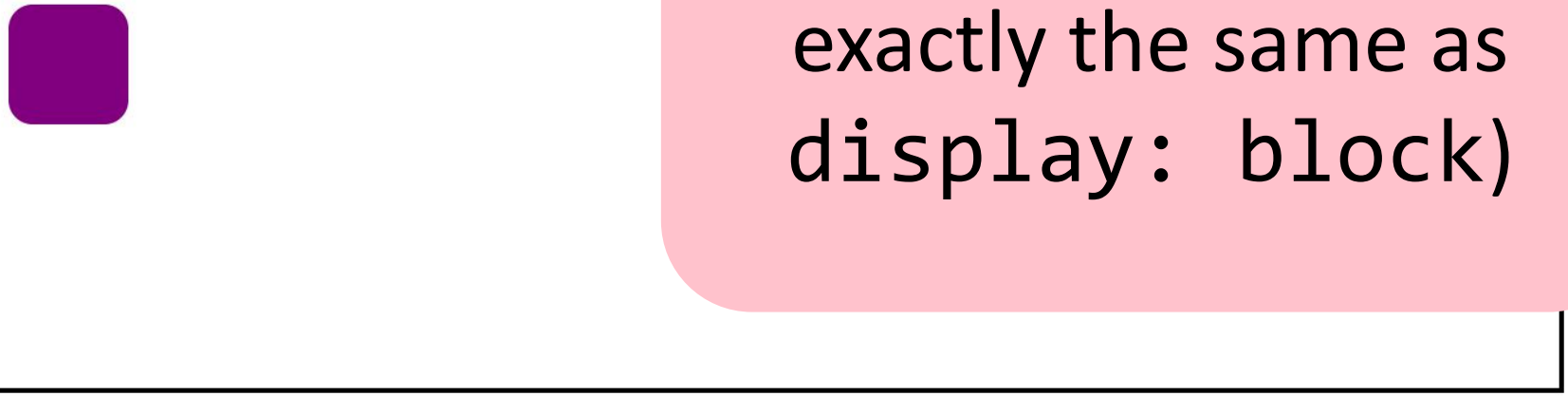


HTML

```html
<html>
  <head>
    <meta charset="utf-8">
    <title>Flexbox example</title>
  </head>
  <body>

    <div id="flex-container">
      <div class="flex-item"></div>
    </div>

  </body>
</html>
```

CSS

```css
#flex-container {
  display: flex;
  border: 2px solid black;
  padding: 10px;
  height: 150px;
}

.flex-item {
  border-radius: 10px;
  background-color: purple;
  height: 50px;
```



(So far, this looks exactly the same as `display: block`)

Flex basics: `justify-content`

You can control where the item is horizontally* in the box by setting `justify-content` on the flex container:

```
#flex-container {
  display: flex;
  justify-content: flex-start;
}
```

*when flex direction is `row`. We'll get to what "flex direction" means soon.

Flex basics: `justify-content`

You can control where the item is horizontally* in the box by setting `justify-content` on the flex container:

```
#flex-container {
  display: flex;
  justify-content: flex-end;
}
```

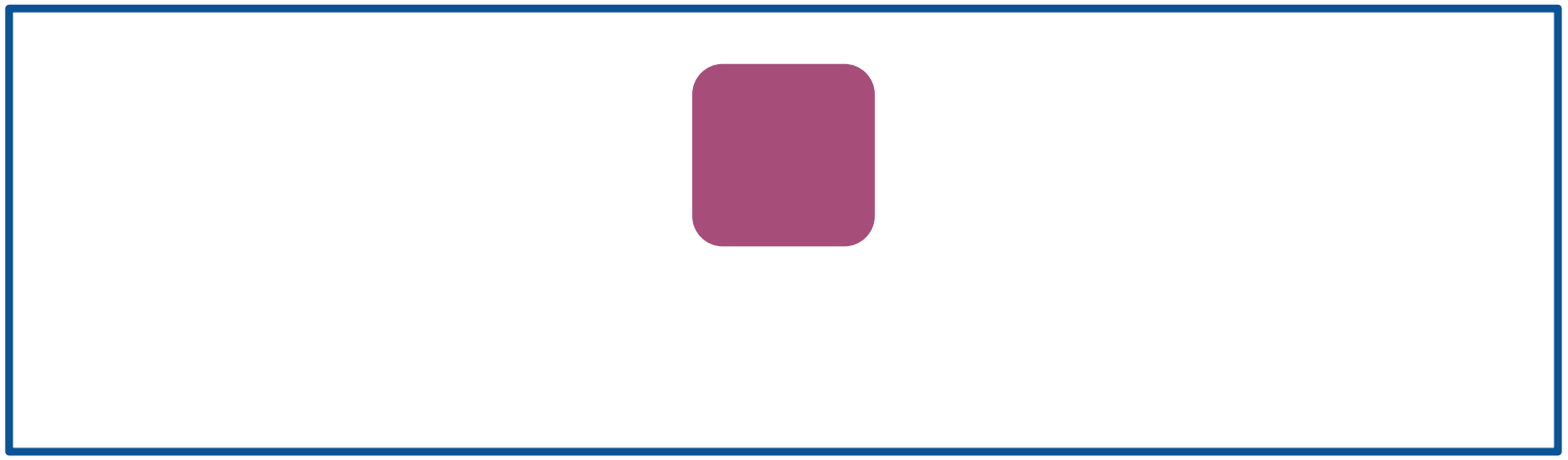
*when flex direction is `row`. We'll get to what "flex direction" means soon.

Flex basics: `justify-content`

You can control where the item is horizontally* in the box by setting `justify-content` on the flex container:

```
#flex-container {
  display: flex;
  justify-content: center;
}
```

*when flex direction is `row`. We'll get to what "flex direction" means soon.



Flex basics: `align-items`

You can control where the item is vertically* in the box by setting `align-items` on the flex container:

```
#flex-container {
  display: flex;
  align-items: flex-start;
}
```

*when flex direction is `row`. We'll get to what "flex direction" means soon.

Flex basics: `align-items`

You can control where the item is vertically* in the box by setting `align-items` on the flex container:

```
#flex-container {
  display: flex;
  align-items: flex-end;
}
```

*when flex direction is `row`. We'll get to what "flex direction" means soon.

Flex basics: `align-items`

You can control where the item is vertically* in the box by setting `align-items` on the flex container:

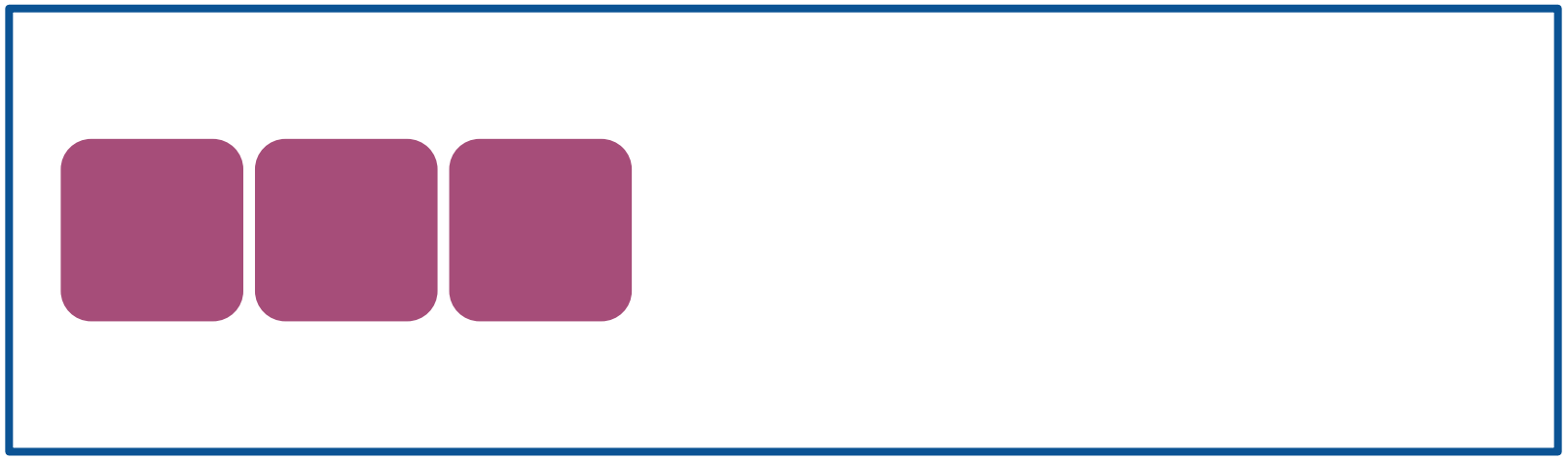
```
#flex-container {
  display: flex;
  align-items: center;
}
```

*when flex direction is `row`. We'll get to what "flex direction" means soon.

Multiple items

Same rules apply with multiple flex items:

```
#flex-container {
  display: flex;
  justify-content: flex-start;
  align-items: center;
}
```



Multiple items

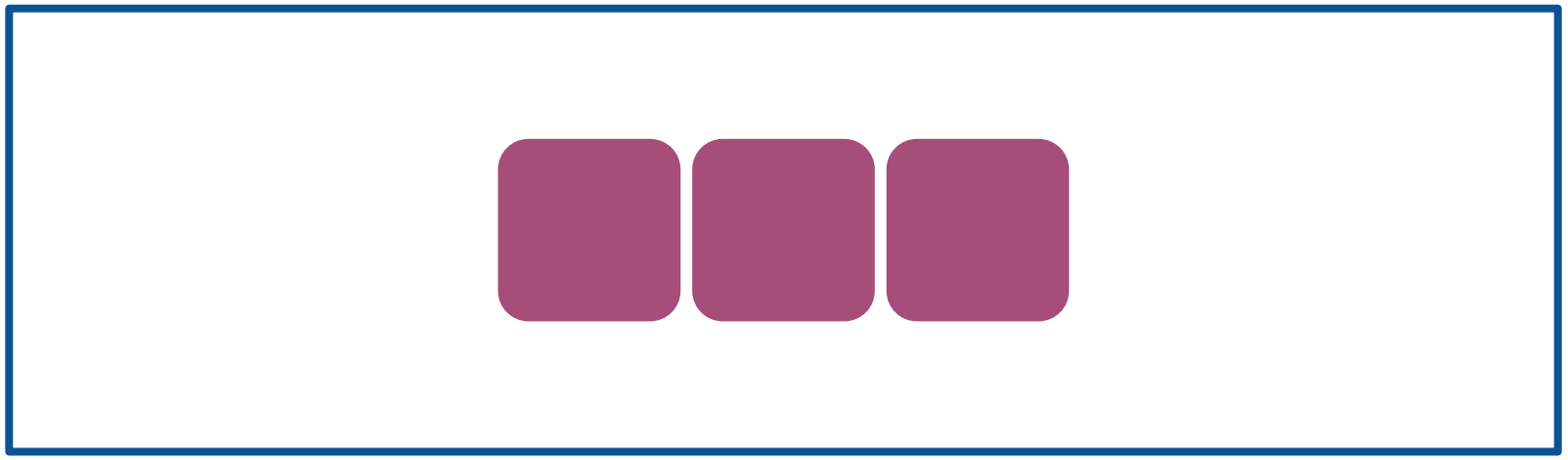
Same rules apply with multiple flex items:

```
#flex-container {
  display: flex;
  justify-content: flex-end;
  align-items: center;
}
```

Multiple items

Same rules apply with multiple flex items:

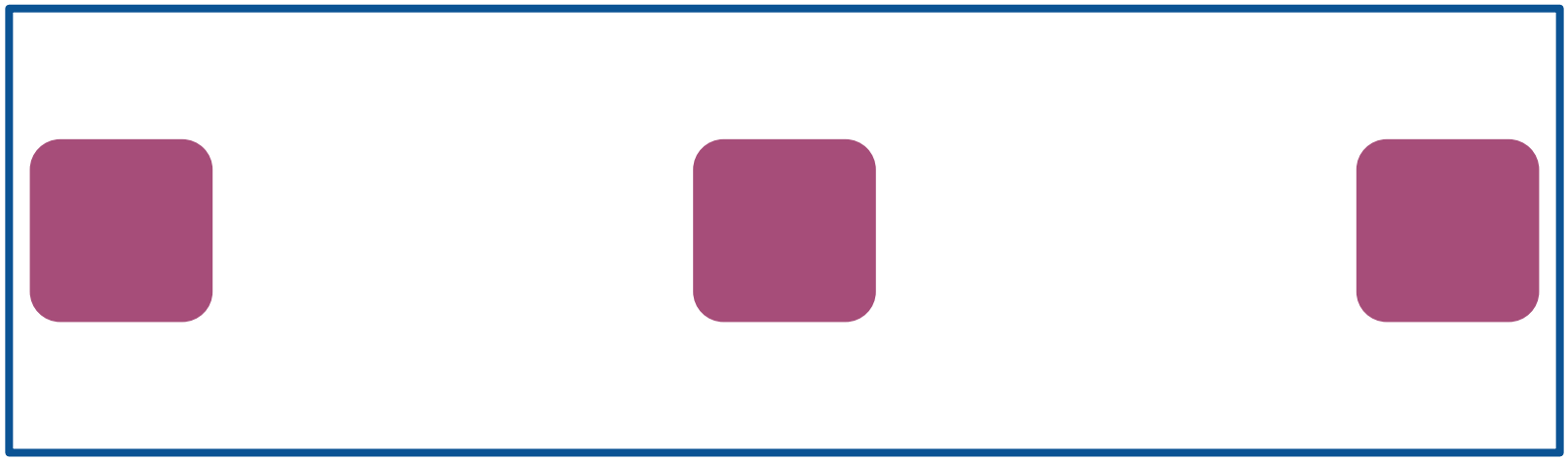
```
#flex-container {
  display: flex;
  Justify-content: center;
  align-items: center;
}
```



Multiple items

And there is also **space-between** and **space-around**:

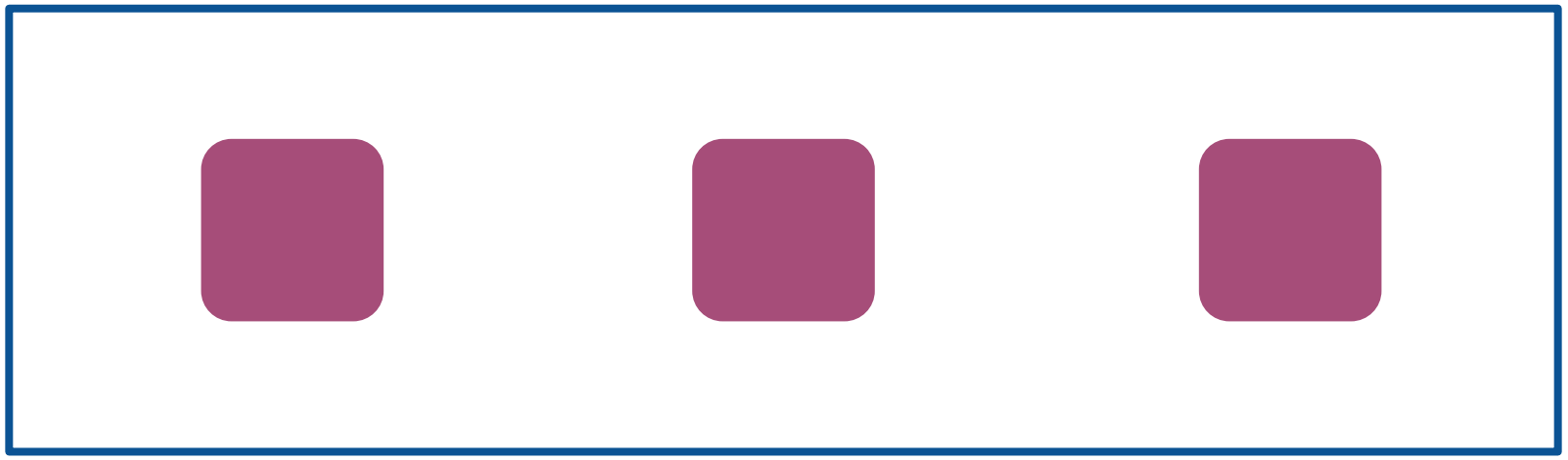
```
#flex-container {
  display: flex;
  Justify-content: space-between;
  align-items: center;
}
```



Multiple items

And there is also **space-between** and **space-around**:

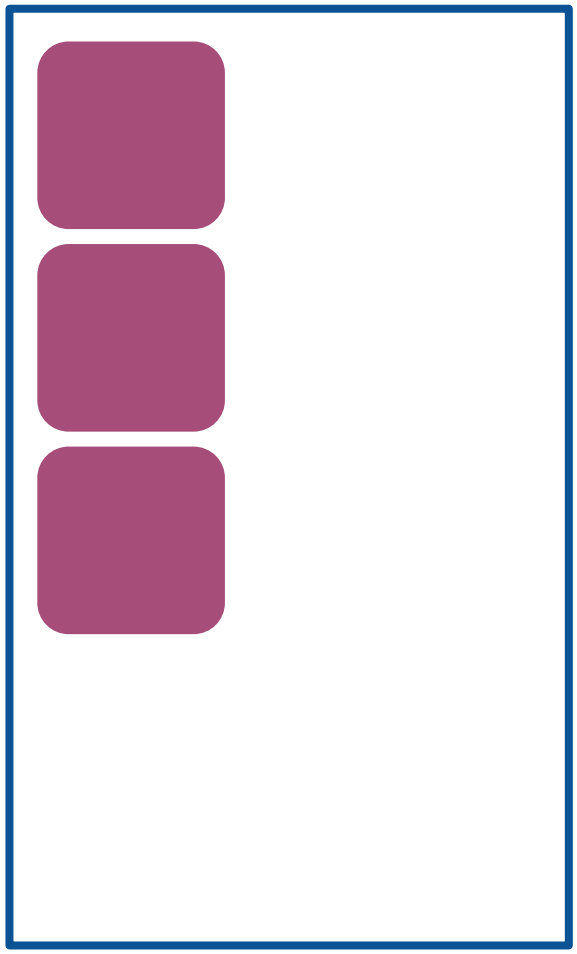
```
#flex-container {
  display: flex;
  Justify-content: space-around;
  align-items: center;
}
```



flex-direction

And you can also lay out columns instead of rows:

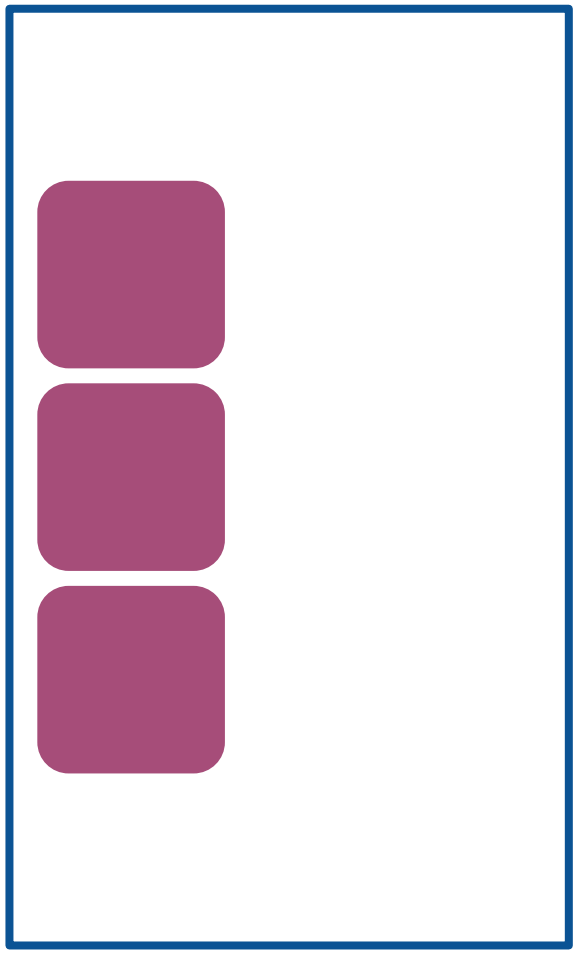
```
#flex-container {
  display: flex;
  flex-direction: column;
}
```



flex-direction

And you can also lay out columns instead of rows:

```
#flex-container {
  display: flex;
  flex-direction: column;
  justify-content: center;
}
```

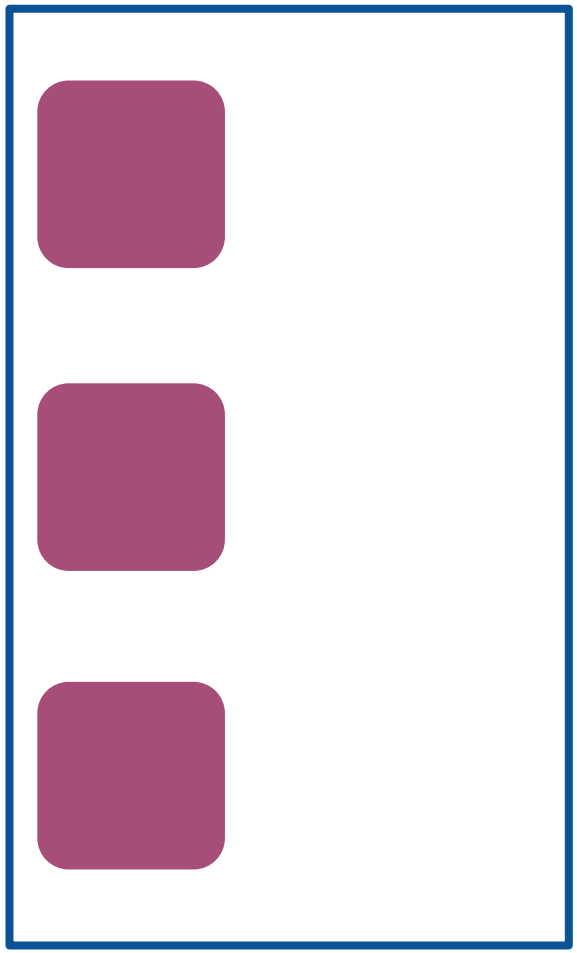


Now **`justify-content`** controls where the column is vertically in the box

flex-direction

And you can also lay out columns instead of rows:

```
#flex-container {
  display: flex;
  flex-direction: column;
  justify-content: space-around;
}
```

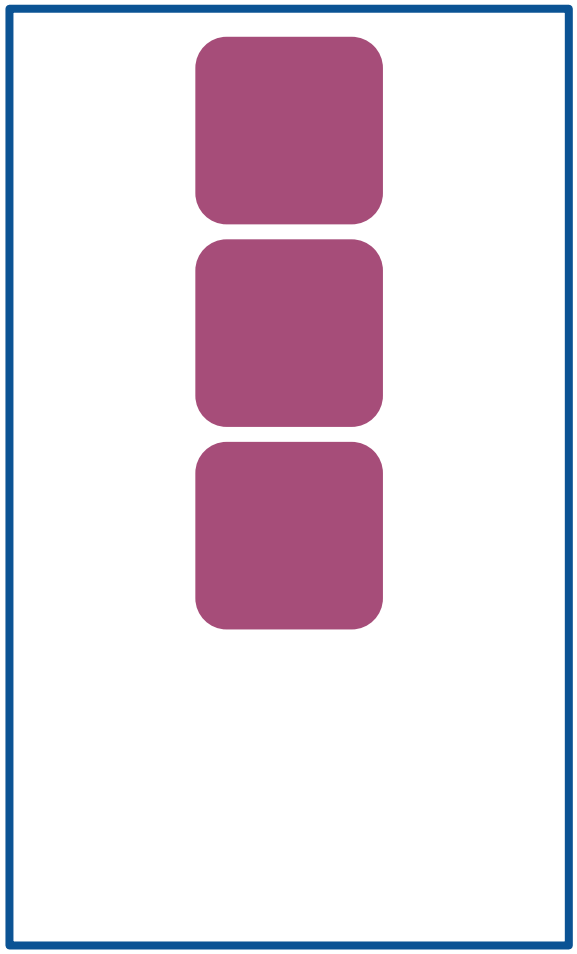


Now **`justify-content`** controls where the column is vertically in the box

flex-direction

And you can also lay out columns instead of rows:

```
#flex-container {
  display: flex;
  flex-direction: column;
  align-items: center;
}
```

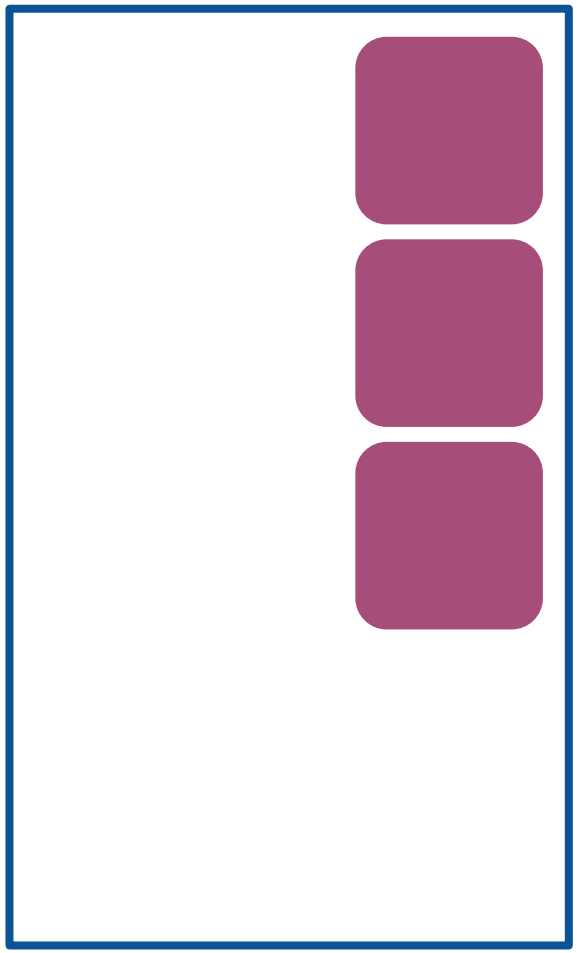


Now **`align-items`** controls where the column is horizontally in the box

flex-direction

And you can also lay out columns instead of rows:

```
#flex-container {
  display: flex;
  flex-direction: column;
  align-items: flex-end;
}
```



Now **`align-items`** controls where the column is horizontally in the box

Before we move on...

What happens if the flex item is an inline element?

HTML

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Flexbox example</title>
  </head>
  <body>

    <div id="flex-container">
      <span class="flex-item"></span>
      <span class="flex-item"></span>
      <span class="flex-item"></span>
    </div>

  </body>
```

CSS

```
#flex-container {
  display: flex;
  border: 2px solid black;
  height: 150px;
}

.flex-item {
  border-radius: 10px;
  background-color: purple;
  height: 50px;
  width: 50px;
  margin: 5px;
}
```

JS

???

HTML

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Flexbox example</title>
  </head>
  <body>

    <div id="flex-container">
      <span class="flex-item"></span>
      <span class="flex-item"></span>
      <span class="flex-item"></span>
    </div>

  </body>
```

CSS

```
#flex-container {
  display: flex;
  border: 2px solid black;
  height: 150px;
}

.flex-item {
  border-radius: 10px;
  background-color: purple;
  height: 50px;
  width: 50px;
  margin: 5px;
}
```

JS

Recall: block layouts

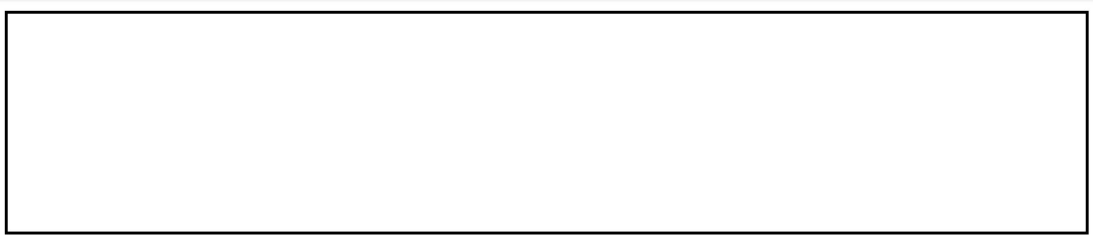
If `#flex-container` was **not** `display: flex;`

HTML

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Flexbox example</title>
  </head>
  <body>

    <div id="flex-container">
      <span class="flex-item"></span>
      <span class="flex-item"></span>
      <span class="flex-item"></span>
    </div>

  </body>
</html>
```

CSS

```
#flex-container {
  border: 2px solid black;
  height: 150px;
}

.flex-item {
  border-radius: 10px;
  background-color: purple;
  height: 50px;
  width: 50px;
  margin: 5px;
}
```



Then the `span` `flex-items` would not show up because `span` elements are inline, which don't have a height and width

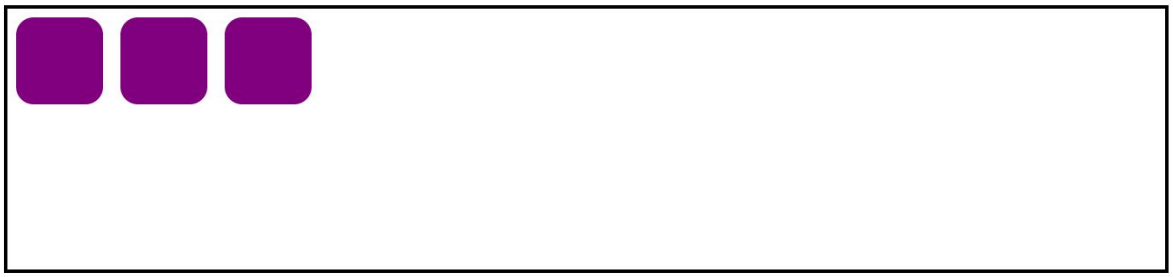
Flex layouts

HTML

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Flexbox example</title>
  </head>
  <body>

    <div id="flex-container">
      <span class="flex-item"></span>
      <span class="flex-item"></span>
      <span class="flex-item"></span>
    </div>

  </body>
```

CSS

```
#flex-container {
  display: flex;
  border: 2px solid black;
  height: 150px;
}

.flex-item {
  border-radius: 10px;
  background-color: purple;
  height: 50px;
  width: 50px;
  margin: 5px;
}
```



Why does this change when `display: flex`?

Why do inline elements suddenly seem to have height and width?

Homework

Flexbox

Theory and video

https://htmlandcssguidebook.com/css/flexbox/

Interactive game

https://flexboxfroggy.com/

Try to implement the layout for squarespace from lecture with flexbox!