Interactive Web Programming

1st semester of 2021

Murilo Camargos
(**murilo.filho@fgv.br**)

Heavily based on **Victoria Kirst** slides

Today's schedule

Schedule:

- Box model
- Debugging with Chrome Inspector
- **Case study**: Squarespace Layout
- Flexbox

Announcements:

- Homework 1 is out! Due **Mar 18**.

Next week:

- Intro to JavaScript

Last class

Quick Review

Block vs Inline

1. **block:** flows **top-to-bottom**; **has** **`height`** and **`width`**

   `<p>, <h1>, <blockquote>, <ol>, <ul>, <table>`

2. **inline:** flows **left-to-right**; **does not have** **`height`** and **`width`**

   `<a>, <em>, <strong>,<br>`

   a. **inline block:** flows **left-to-right**; **has** **`height`** and **`width`**

      equal to size of the content

      `<img>`

CSS Selectors

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

Generic elements **div** vs **span**

Two generic tags with no intended purpose or style:

- `<div>`: a generic **block** element
- `<span>`: a generic **inline** element

Technically, you can define your entire web page using `<div>` and the `class` attribute.

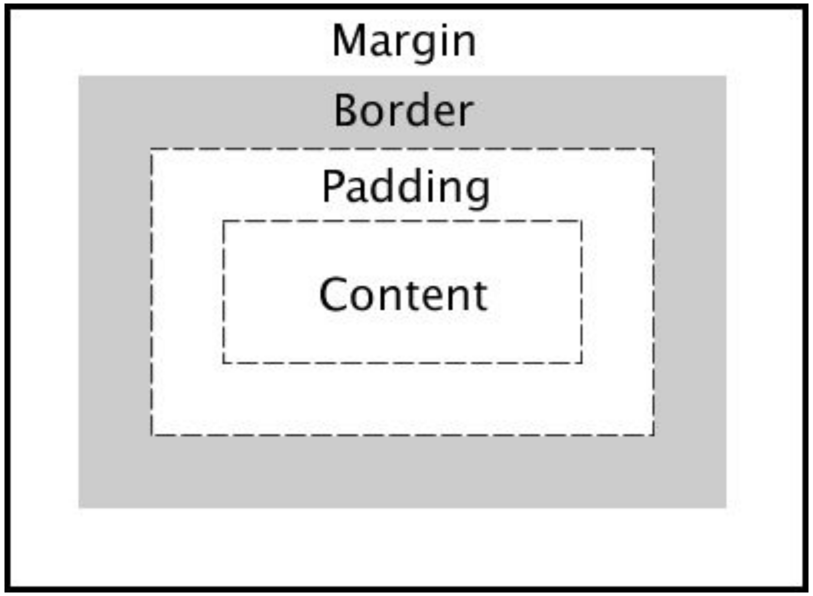
- Is this a good idea?
- Why does HTML have `id`s when you have `class`es?
- Why does HTML have `<p>`, `<h1>`, `<strong>`, etc. when you have `<div>`, `<span>`, `class`, and `id`?

CSS Box Model

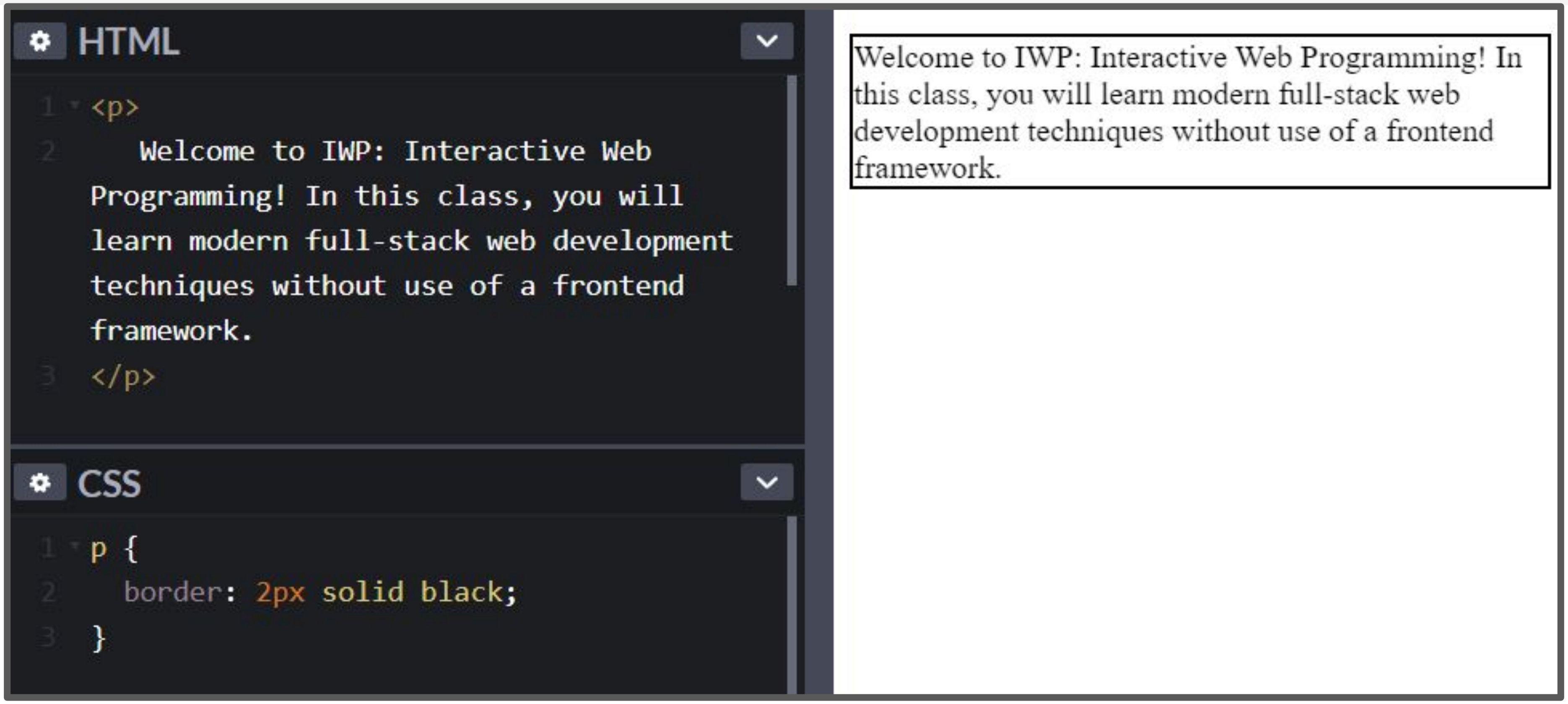
The CSS Box Model

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)



border


HTML

```html
<p>
   Welcome to IWP: Interactive Web Programming! In this class, you will learn modern full-stack web development techniques without use of a frontend framework.
</p>
```

CSS

```css
p {
  border: 2px solid black;
}
```

Welcome to IWP: Interactive Web Programming! In this class, you will learn modern full-stack web development techniques without use of a frontend framework.

We've used the shorthand:

`border:` *width style color*`;`

border

Can also specify each border individually:

```
border-top
border-bottom
border-left
border-right
```

And can set each property individually:

```
border-style: dotted;
border-width: 3px;
border-color: purple;
```

(all styles)

border

Can also specify each border individually:

```
border-top
border-bottom
border-left
border-right
```

And can set each property individually:

```
border-style: dotted;
border-width: 3px;
border-color: purple;
```

(all styles)

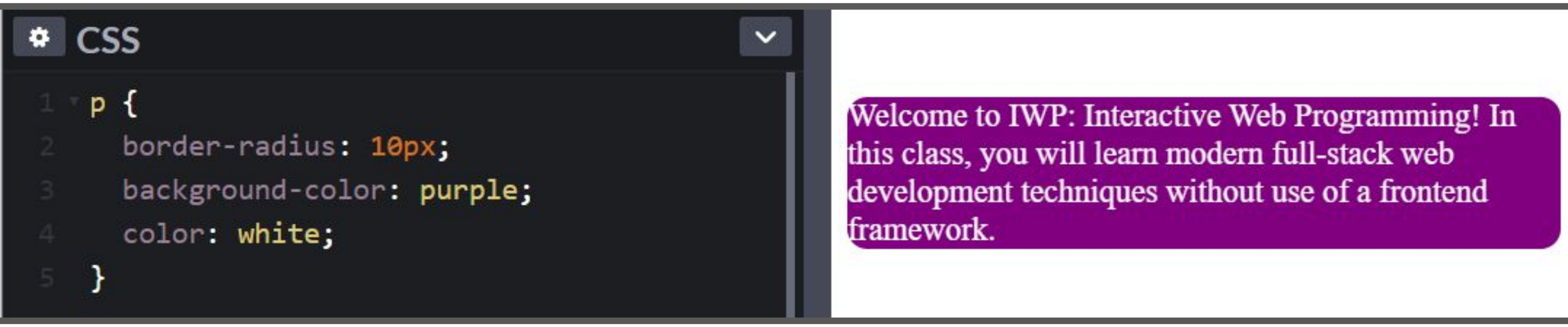
There are other units besides pixels (px) but we will address them latter.

Rounded border

Can specify the `border-radius` to make rounded corners:

```
border-radius: 10px;
```

You don't actually need to set a border to use border-radius.



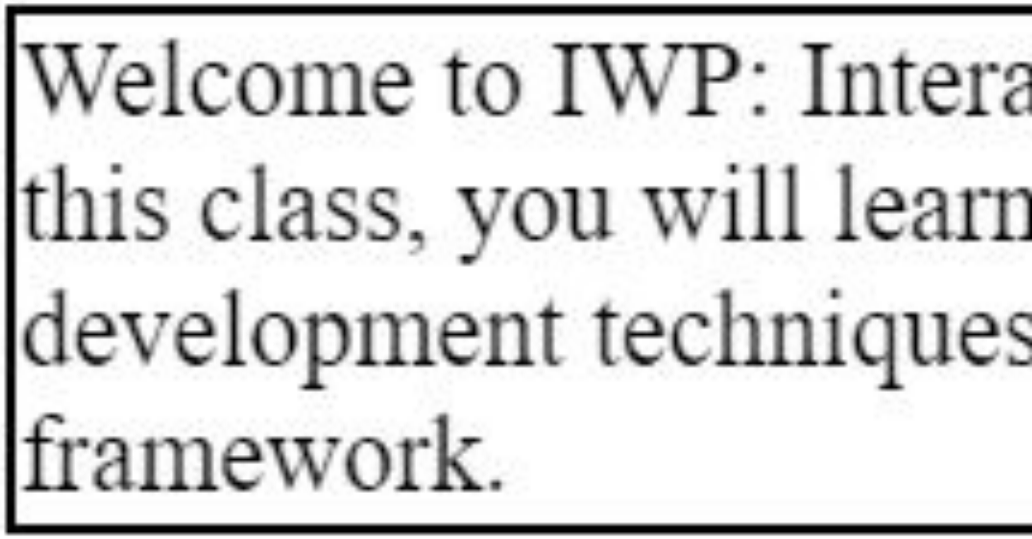
```
p {
  border-radius: 10px;
  background-color: purple;
  color: white;
}
```

Welcome to IWP: Interactive Web Programming! In this class, you will learn modern full-stack web development techniques without use of a frontend framework.

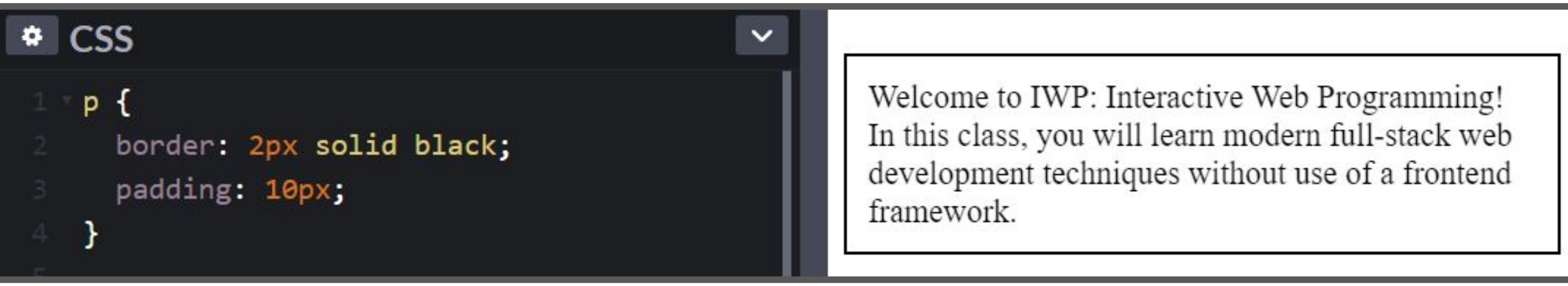
Borders look a little squished

When we add a border to an element, it sits flush against the text:

Q: How do we add space between the border and the content of the element?


Welcome to IWP: Intera
this class, you will learn
development techniques
framework.

padding



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

Welcome to IWP: Interactive Web Programming! In this class, you will learn modern full-stack web development techniques without use of a frontend framework.

`padding` is the space between the border and the content.

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

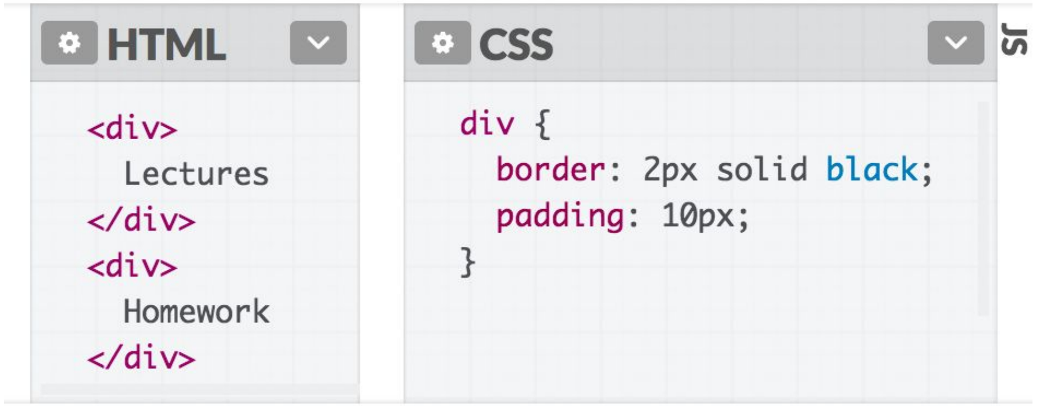
  `padding: 2px 4px 3px 1px;` <- **top|right|bottom|left**

  `padding: 10px 2px;` <- **top+bottom|left+right**

`<div>`s look a little squished

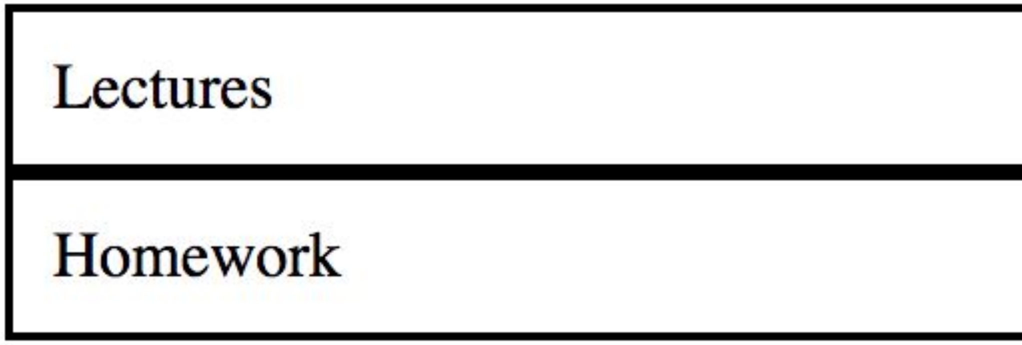
When we add a border to multiple divs, they sit flush against each other:

Q: How do we add space between multiple elements?

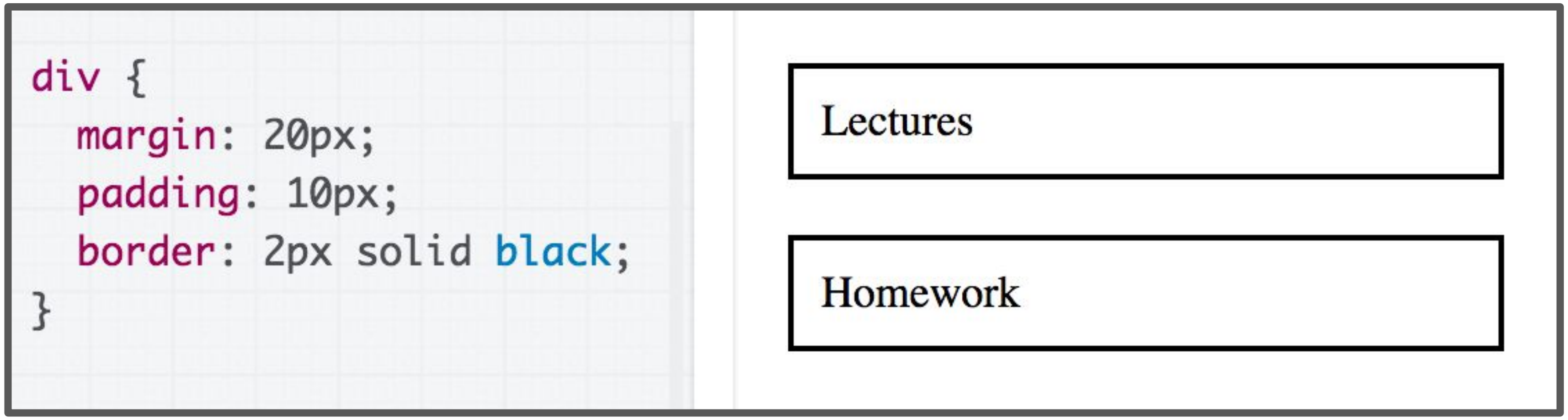


```html
<div>
  Lectures
</div>
<div>
  Homework
</div>
```

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



margin



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

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

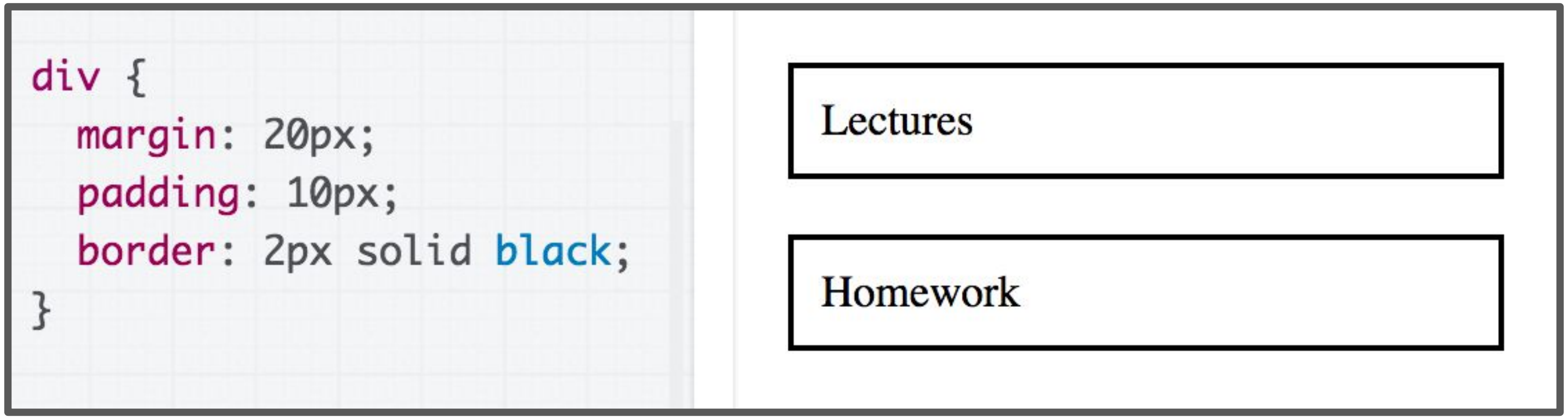
- Can specify `margin-top`, `margin-bottom`, `margin-left`, `margin-right`
- There's also a shorthand:

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

  `margin: 10px 2px;` <- **top+bottom|left+right**

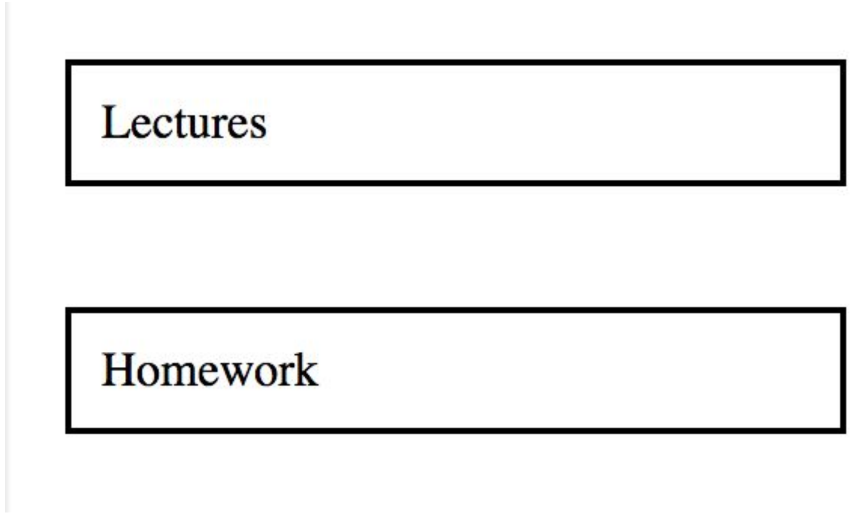
margin

Actually, why doesn't this:



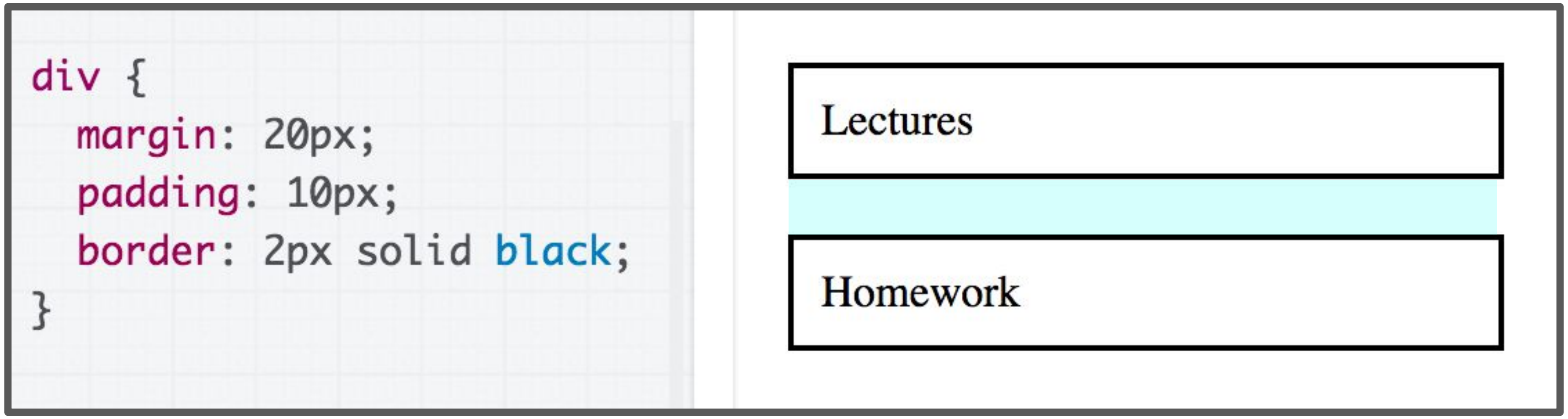
```
div {
  margin: 20px;
  padding: 10px;
  border: 2px solid black;
}
```

Look more like this?



margin

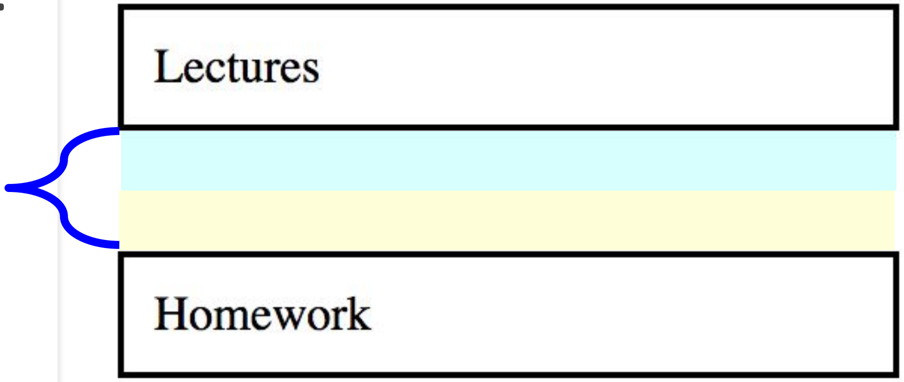
Actually, why doesn't this:



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

...look more like this?

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



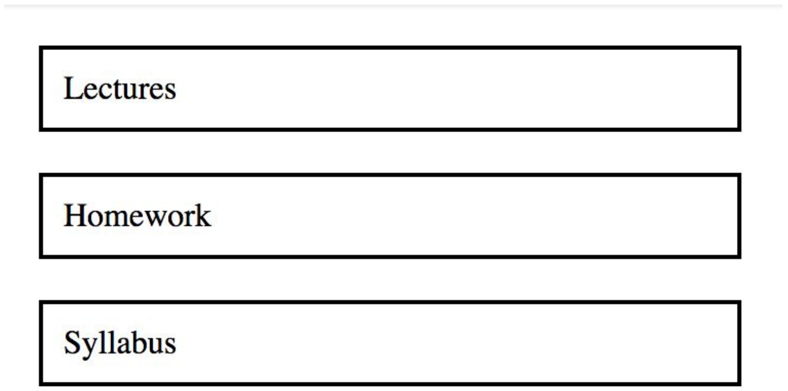
`margin collapsing`

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*)



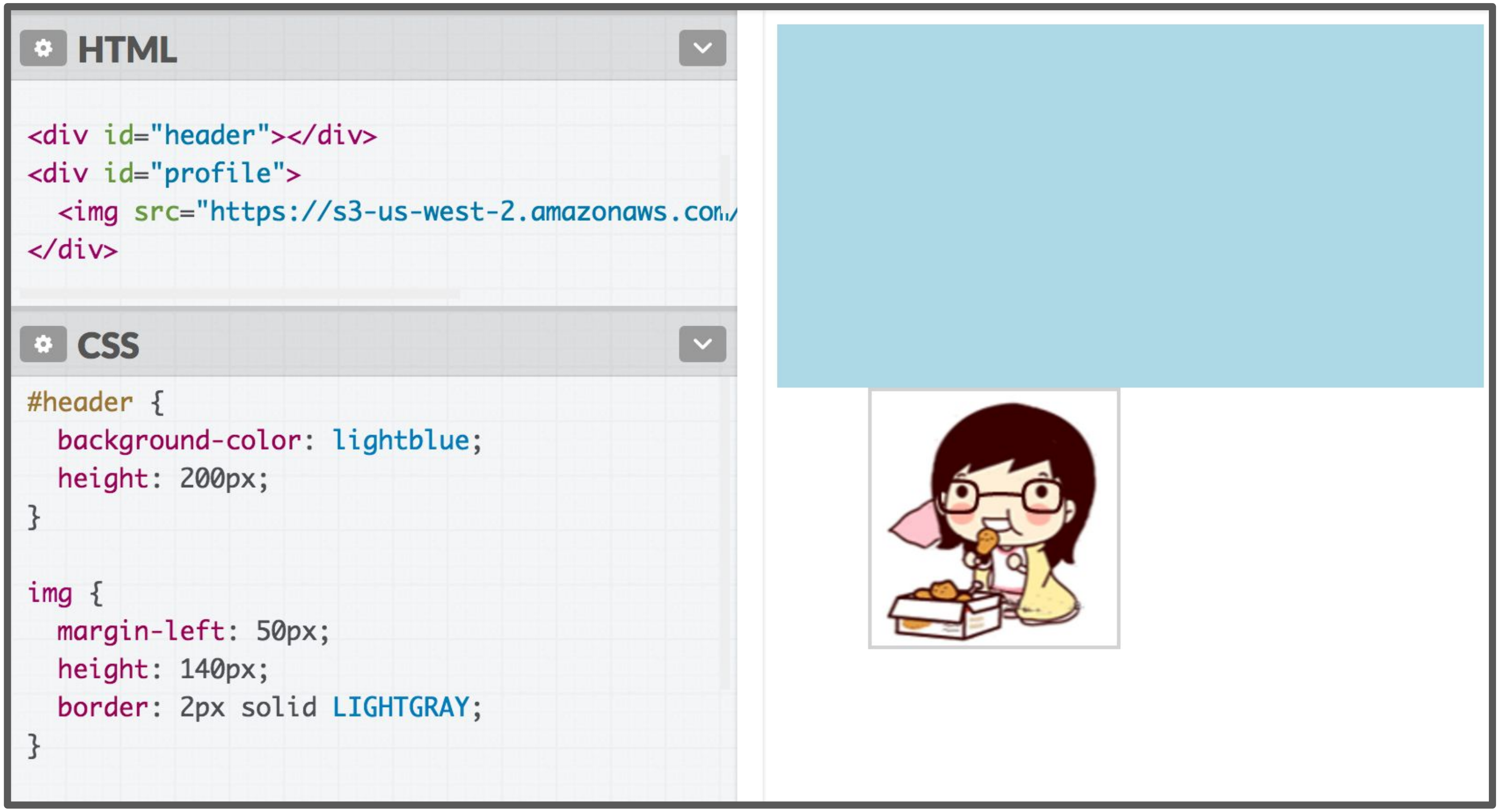
then they collapse into **max**(*Bottom Margin*, *Top Margin*).

(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:**



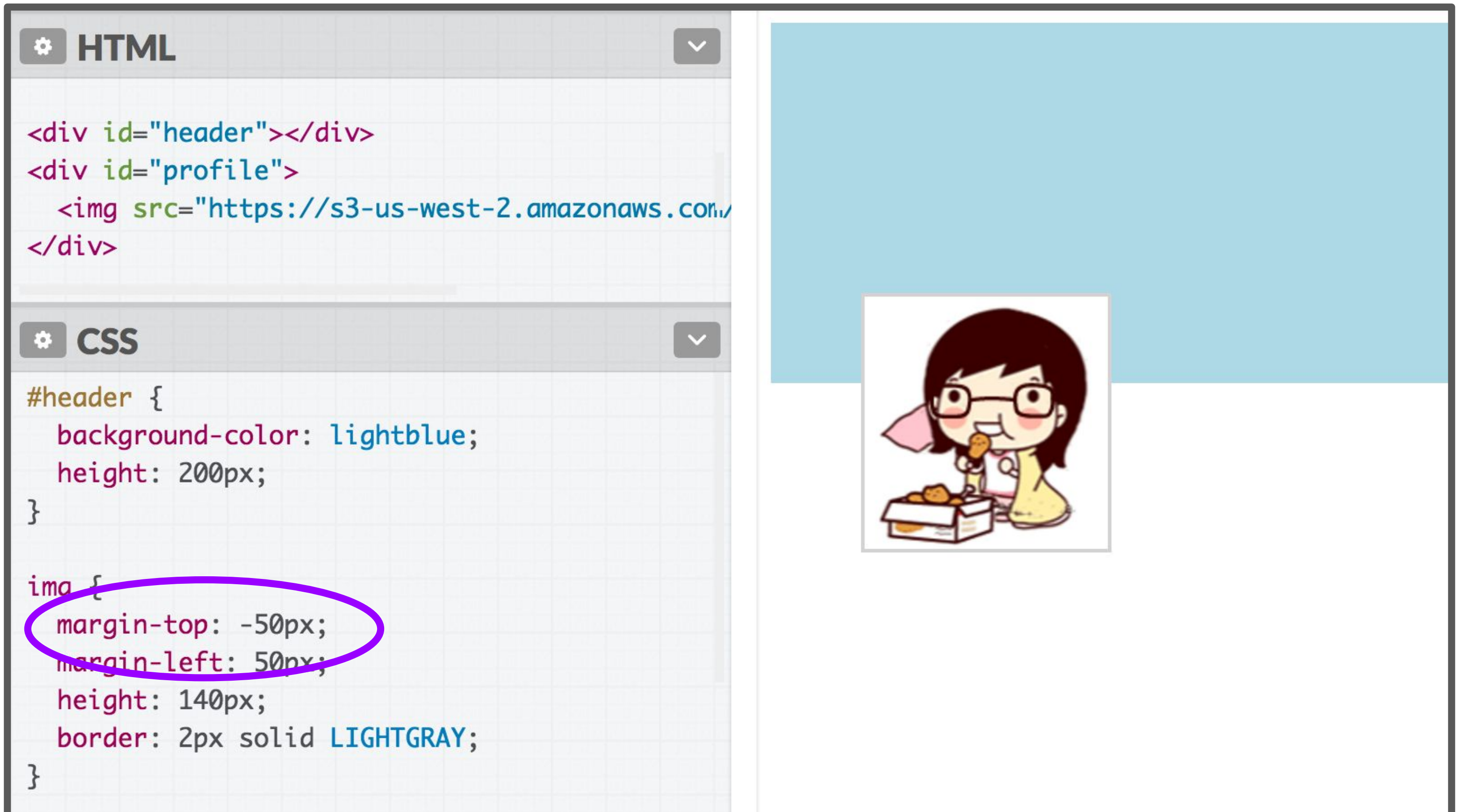
```html
<div id="header"></div>
<div id="profile">
  <img src="https://s3-us-west-2.amazonaws.com/
</div>
```

```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

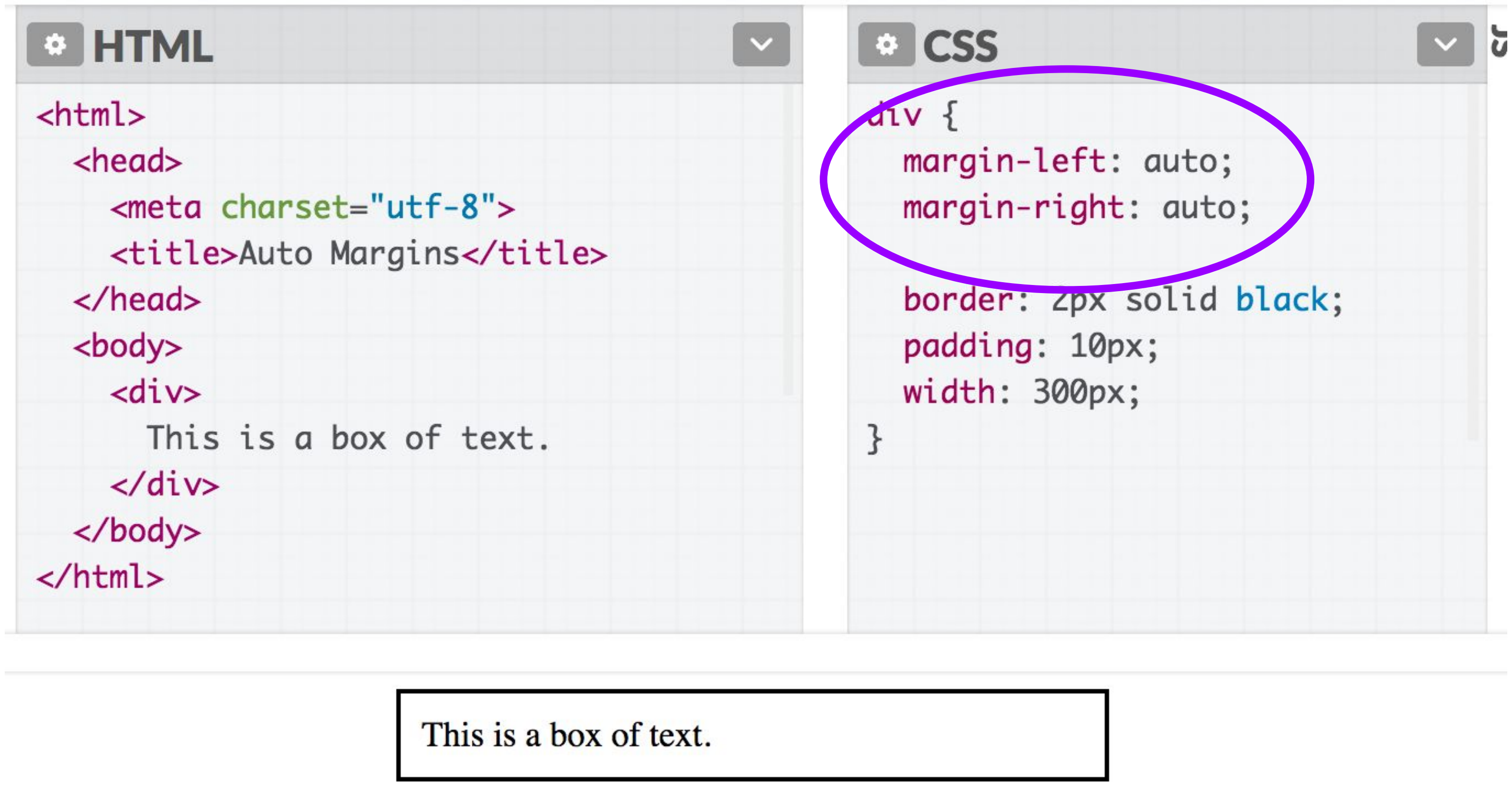
```html
<div id="header"></div>
<div id="profile">
  <img src="https://s3-us-west-2.amazonaws.com/
</div>
```

CSS

```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;
}
```

This is a box of text.

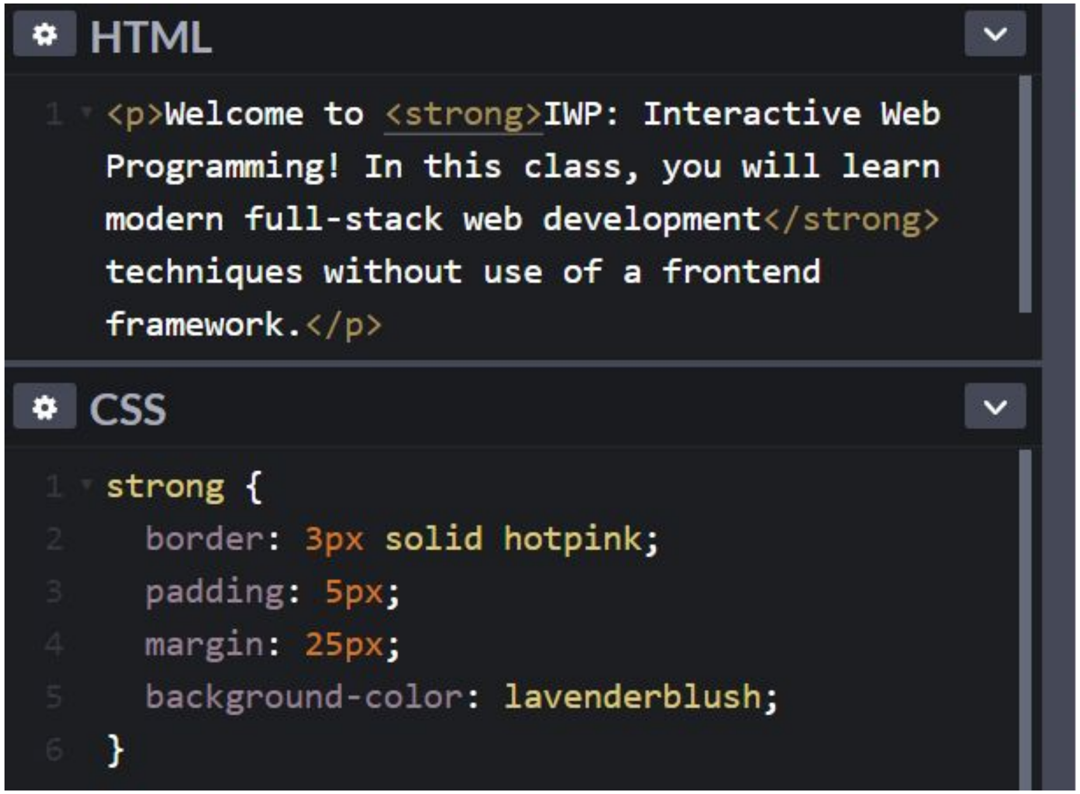
Box model for inline elements?

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

Box model for inline elements?

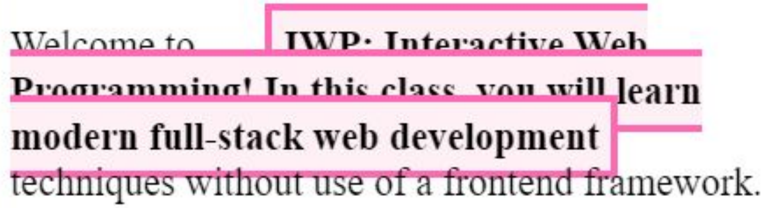
Q: Does the box model apply to inline elements as well?

A: Yes, but the box is shaped differently.



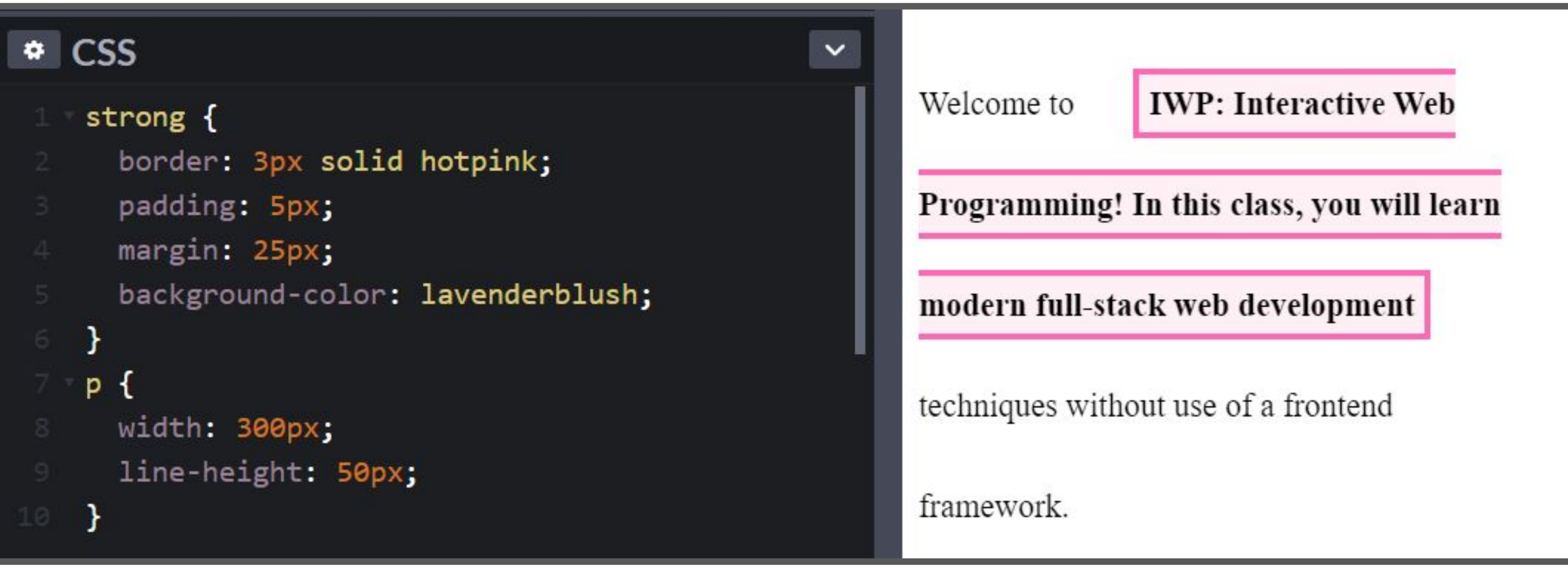
```html
<p>Welcome to <strong>IWP: Interactive Web
Programming! In this class, you will learn
modern full-stack web development</strong>
techniques without use of a frontend
framework.</p>
```

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


Welcome to **IWP: Interactive Web Programming! In this class, you will learn modern full-stack web development** techniques without use of a frontend framework.

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

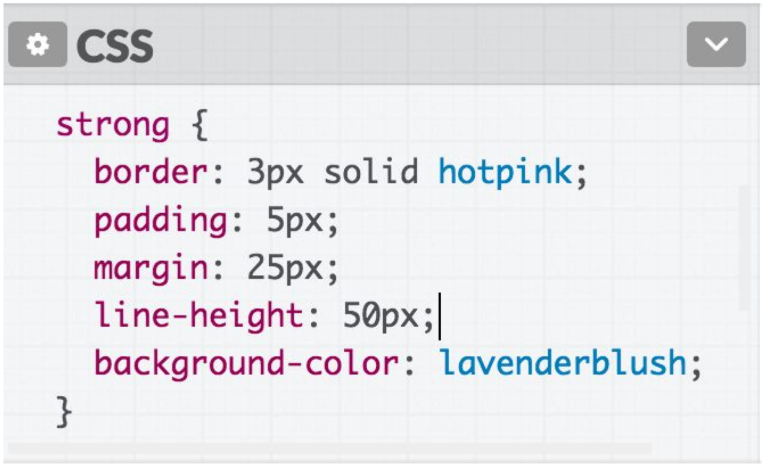
Inline element box model



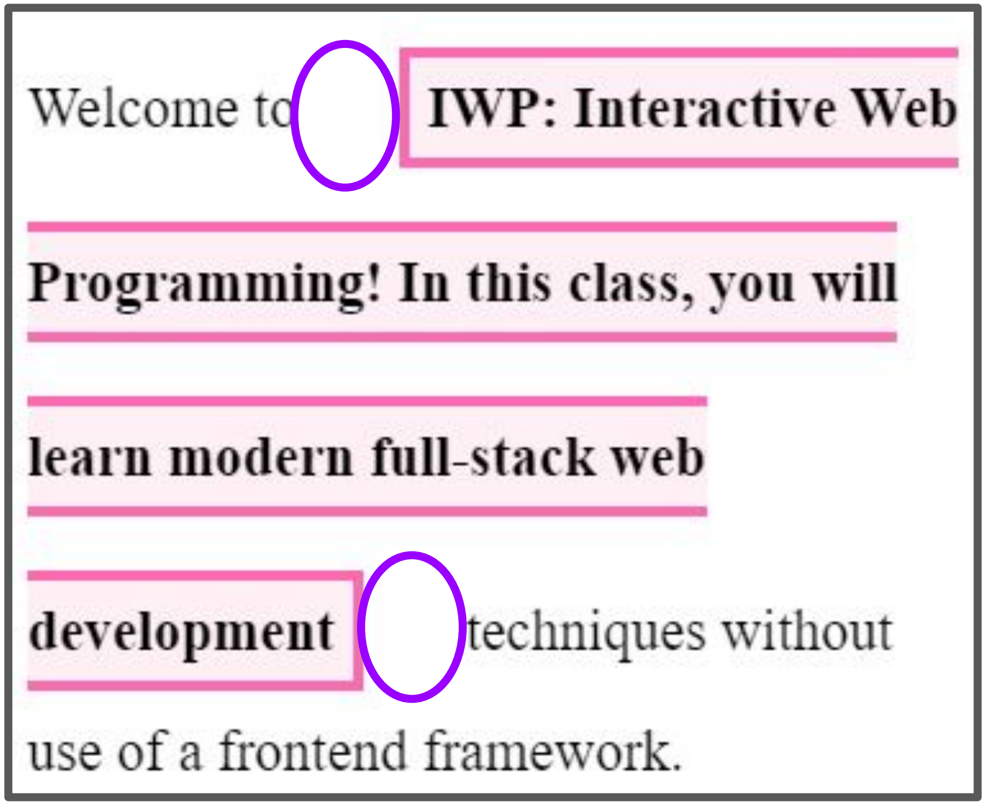
```css
strong {
  border: 3px solid hotpink;
  padding: 5px;
  margin: 25px;
  background-color: lavenderblush;
}
p {
  width: 300px;
  line-height: 50px;
}
```

Welcome to **IWP: Interactive Web Programming! In this class, you will learn modern full-stack web development** techniques without use of a frontend framework.

Inline element box model


CSS

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

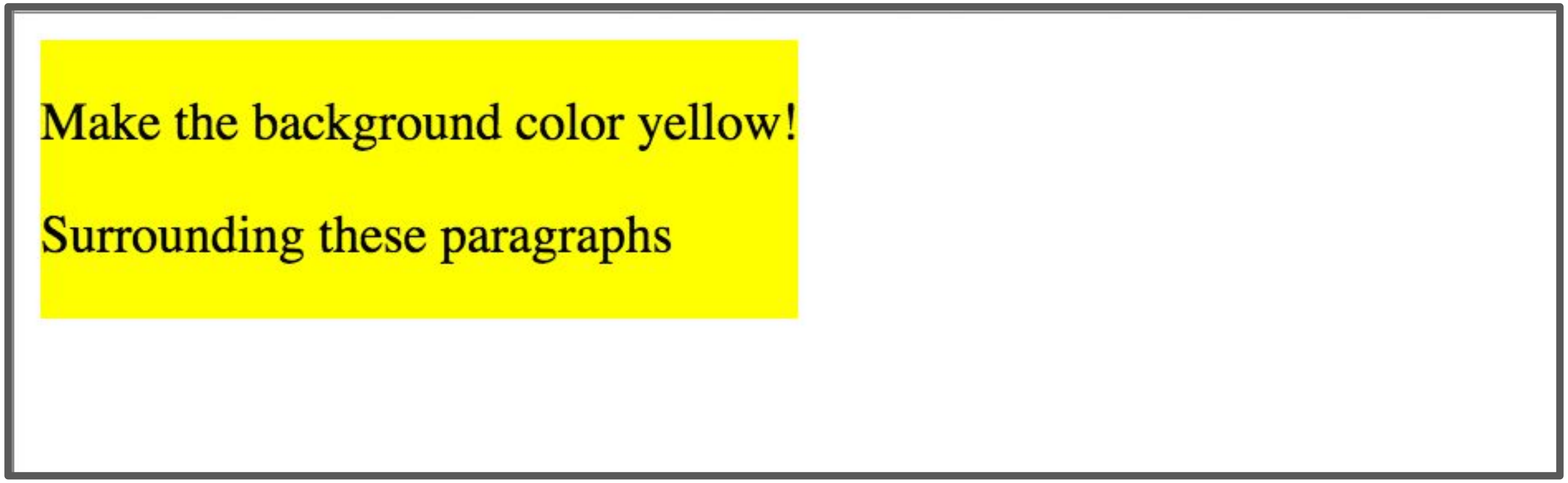

Welcome to **IWP: Interactive Web Programming! In this class, you will learn modern full-stack web development** techniques without use of a frontend framework.

- **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

Q: What does this look like in the browser?

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

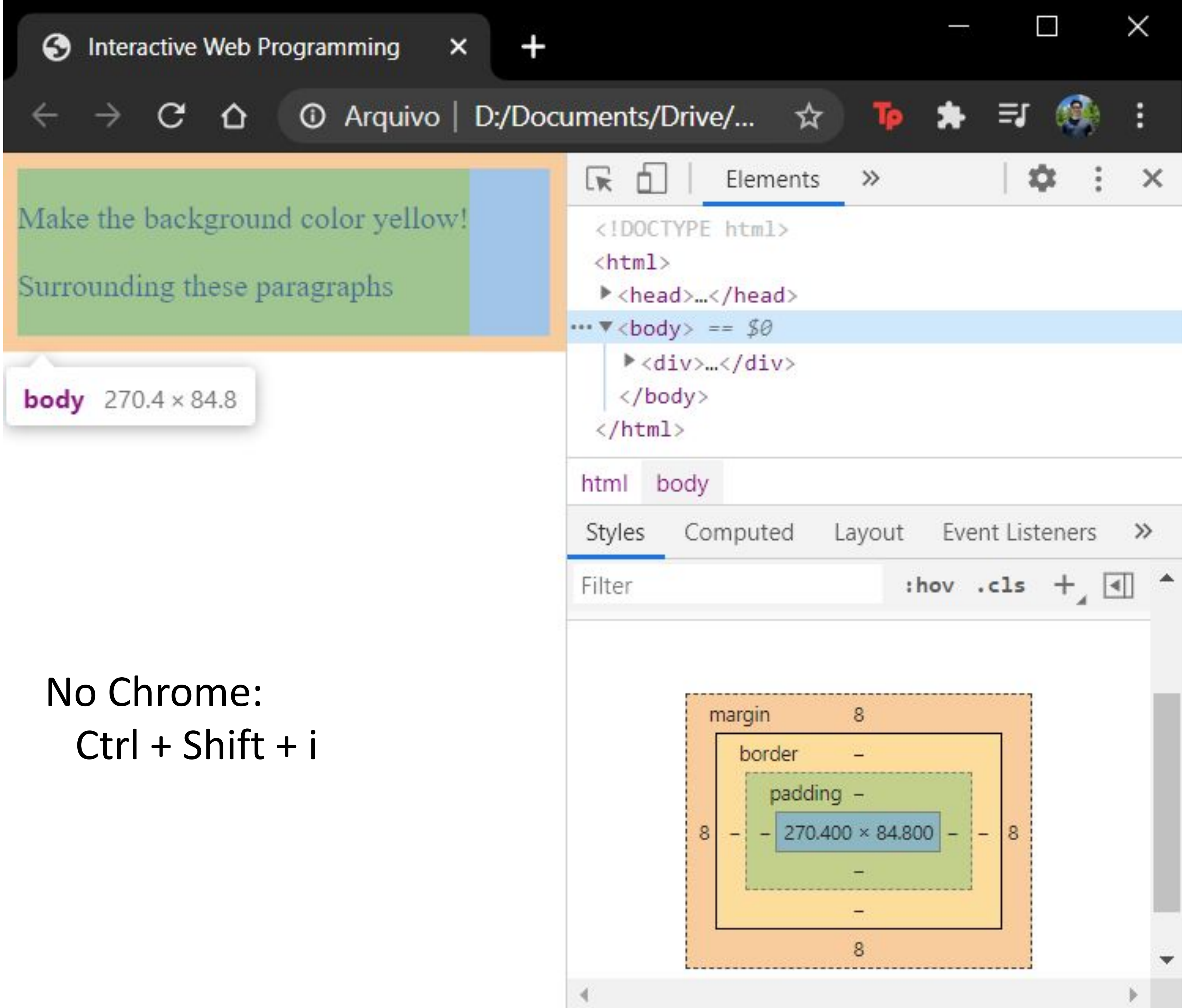
```
<body>
 <div>
  <p>Make the background color yellow!</p>
  <p>Surrounding these paragraphs</p>
 </div>
</body>
```


Make the background color yellow!

Surrounding these paragraphs

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

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



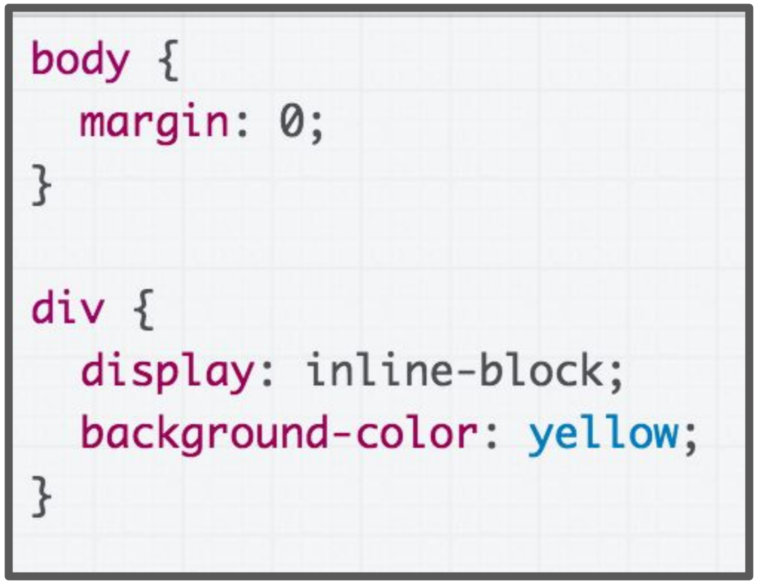
```
<!DOCTYPE html>
<html>
 ▶<head>…</head>
…▼<body> == $0
   ▶<div>…</div>
   </body>
</html>
```

No Chrome:

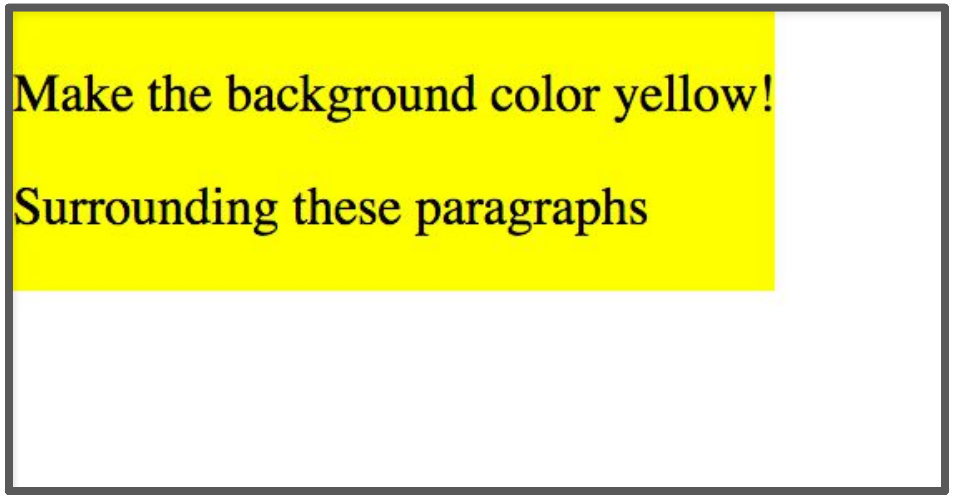
Ctrl + Shift + i

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;
}
```



Make the background color yellow!

Surrounding these paragraphs

Recap so far...

We've talked about:

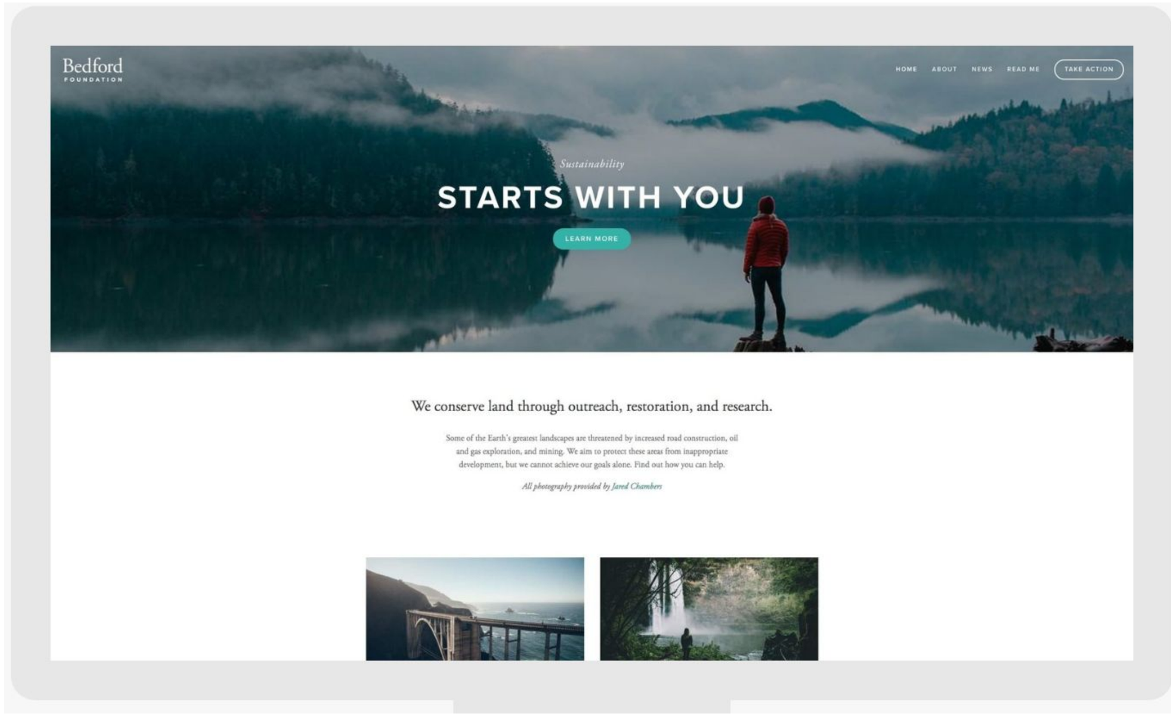
- **block vs inline** and the "natural" layout of the page, depending on the element type
- **`classes` and `ids`** and how to specify specific elements and groups of elements
- **`div` and `span`** and how to create generic elements
- **The CSS box model** and how every element is shaped like a box,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



We conserve land through outreach, restoration, and research.

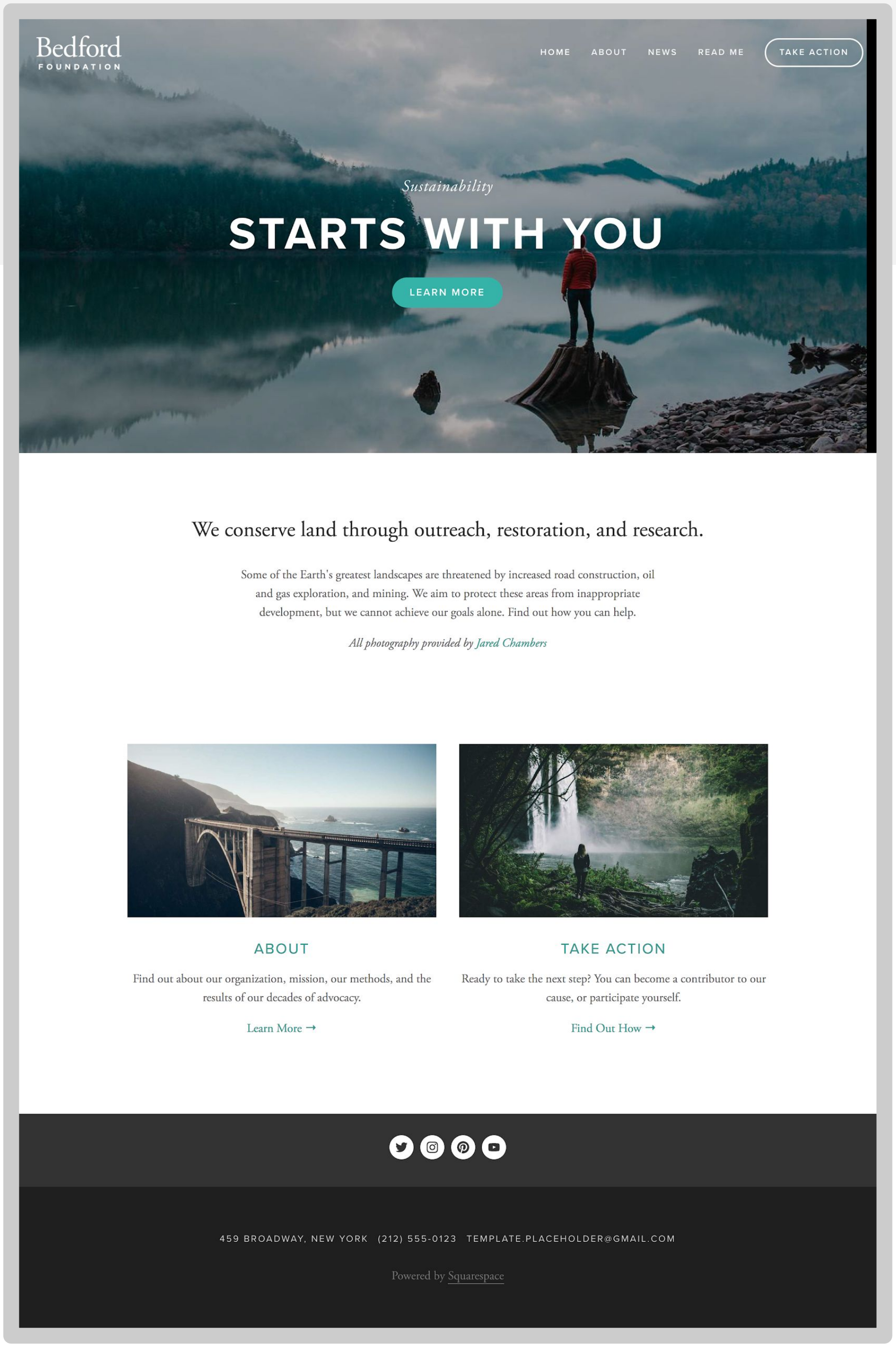
Some of the Earth's greatest landscapes are threatened by increased road construction, oil and gas exploration, and mining. We aim to protect these areas from inappropriate development, but we cannot achieve our goals alone. Find out how you can help.

All photography provided by Jared Chambers

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

Basic shape

Begin visualizing the layout in terms of boxes:



Bedford FOUNDATION

HOME ABOUT NEWS READ ME TAKE ACTION

Sustainability

STARTS WITH YOU

LEARN MORE

We conserve land through outreach, restoration, and research.

Some of the Earth's greatest landscapes are threatened by increased road construction, oil and gas exploration, and mining. We aim to protect these areas from inappropriate development, but we cannot achieve our goals alone. Find out how you can help.

All photography provided by Jared Chambers

ABOUT

Find out about our organization, mission, our methods, and the results of our decades of advocacy.

Learn More →

TAKE ACTION

Ready to take the next step? You can become a contributor to our cause, or participate yourself.

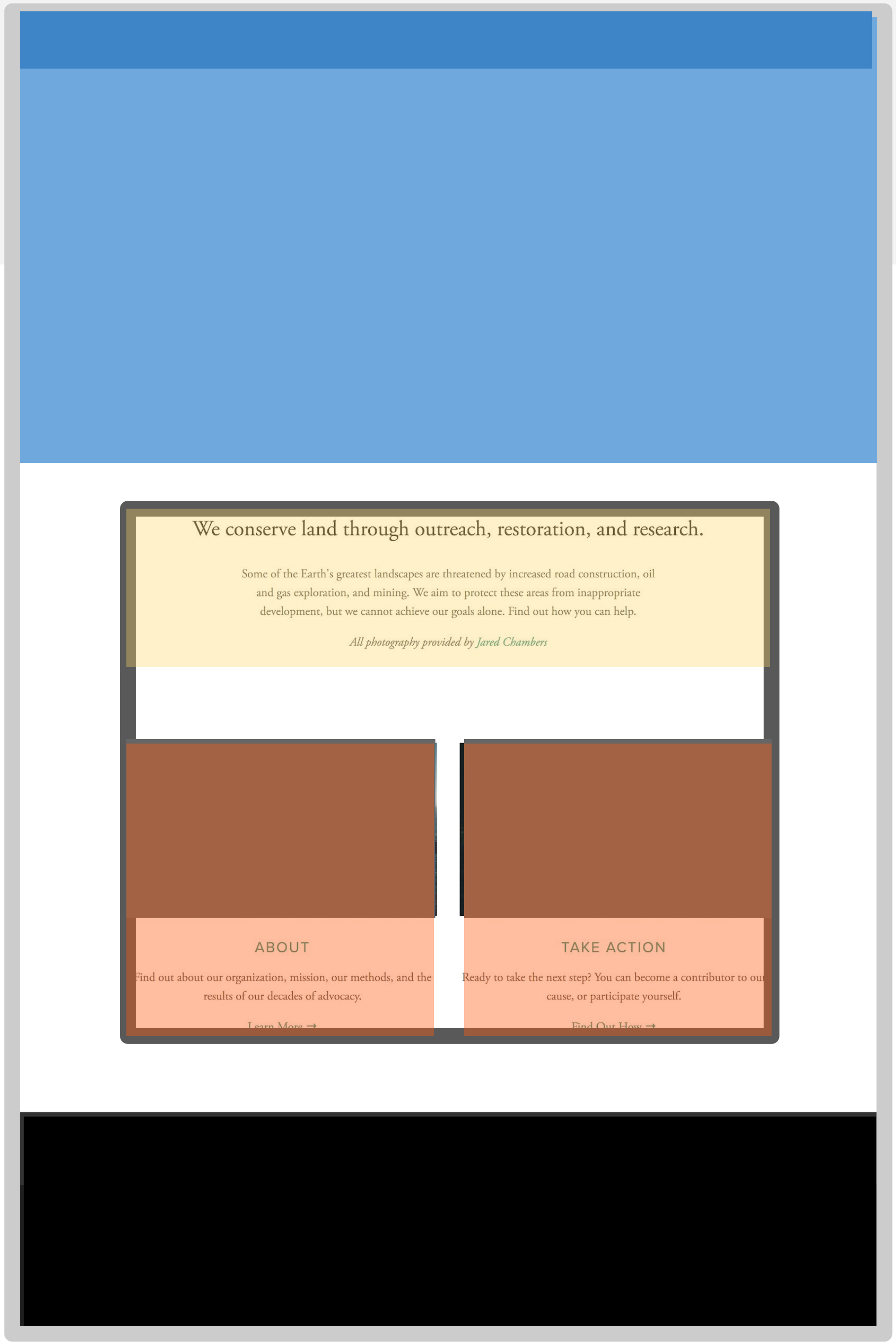
Find Out How →

459 BROADWAY, NEW YORK (212) 555-0123 TEMPLATE.PLACEHOLDER@GMAIL.COM

Powered by Squarespace

Basic shape

Begin visualizing the layout in terms of boxes:



We conserve land through outreach, restoration, and research.

Some of the Earth's greatest landscapes are threatened by increased road construction, oil and gas exploration, and mining. We aim to protect these areas from inappropriate development, but we cannot achieve our goals alone. Find out how you can help.

All photography provided by Jared Chambers

ABOUT

Find out about our organization, mission, our methods, and the results of our decades of advocacy.

Learn More →

TAKE ACTION

Ready to take the next step? You can become a contributor to ou cause, or participate yourself.

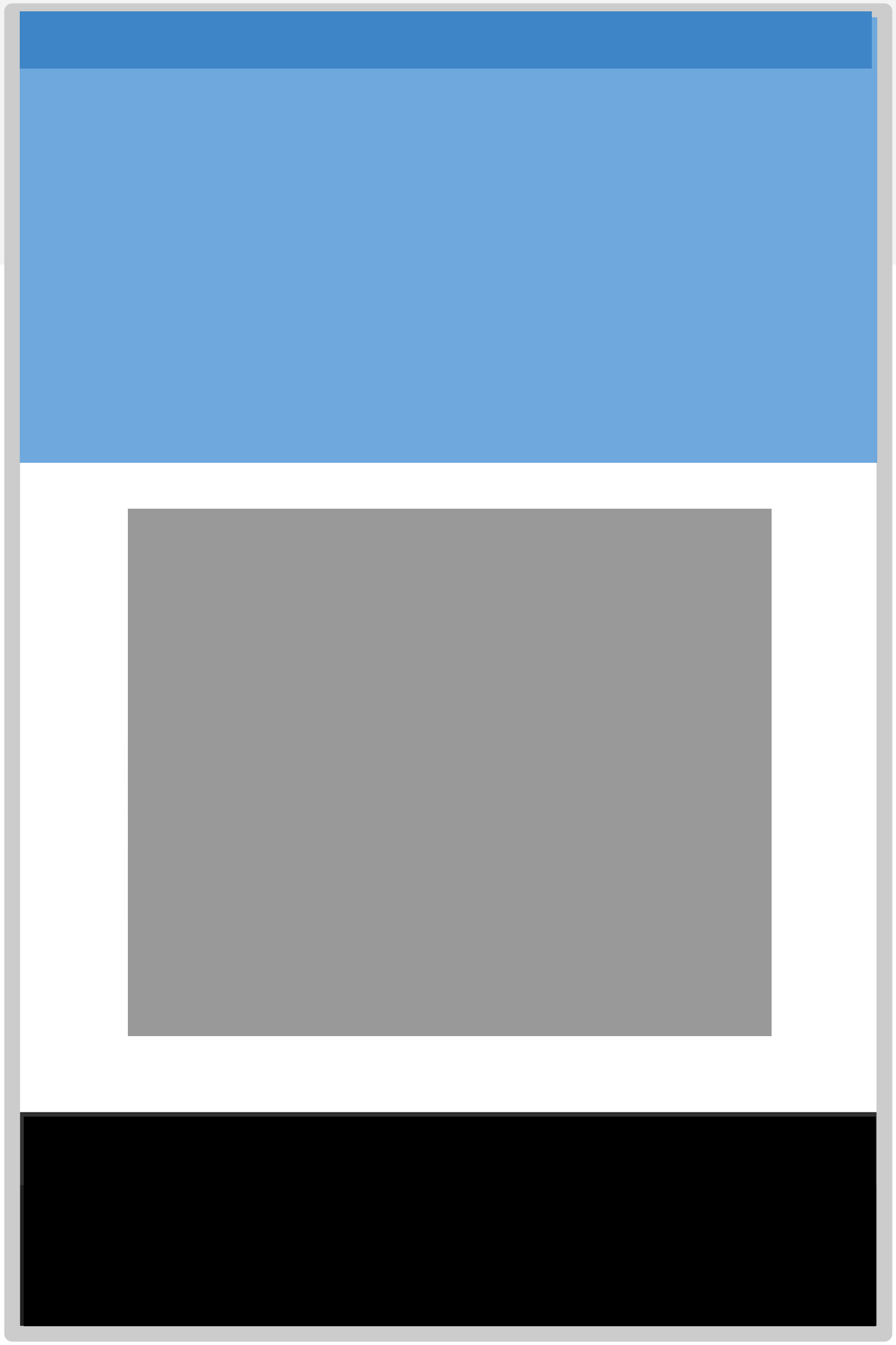
Find Out How →

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

Better SEO and more accessibility

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

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

These elements do not "do" anyth

`<div>`s. Makes your HTML more readable. See MDN for more info.

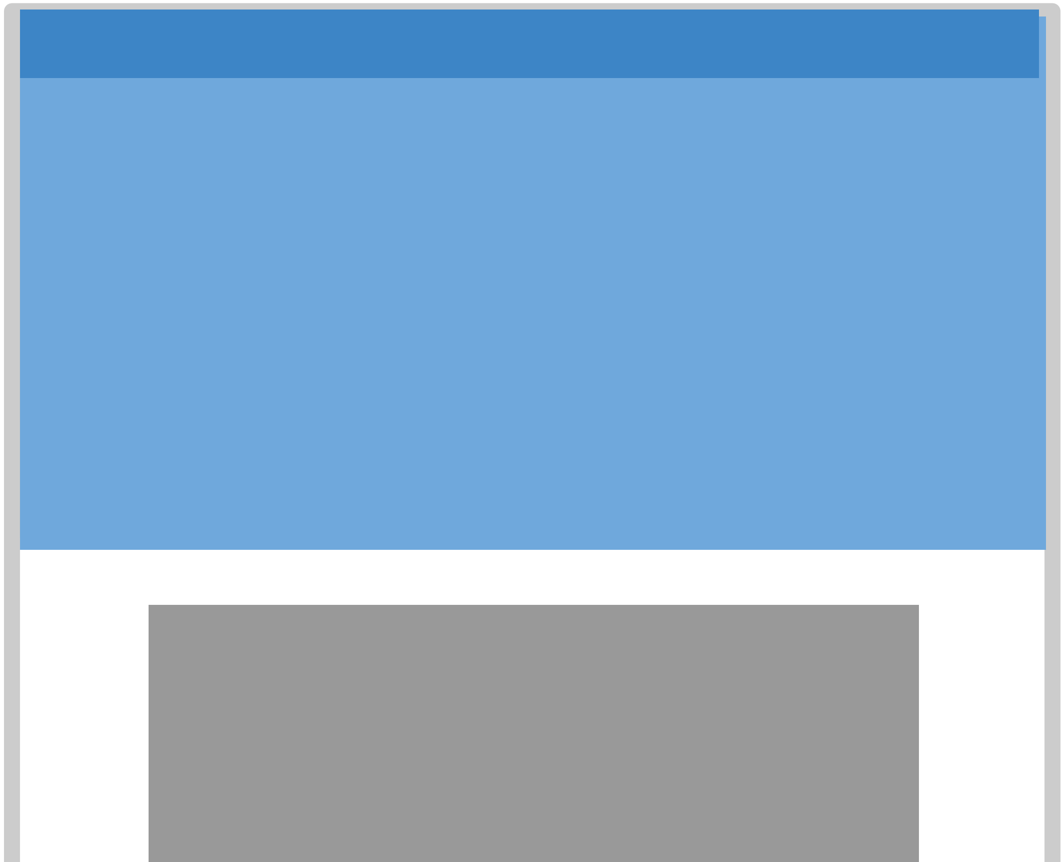
Header

Navbar:

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

Header:

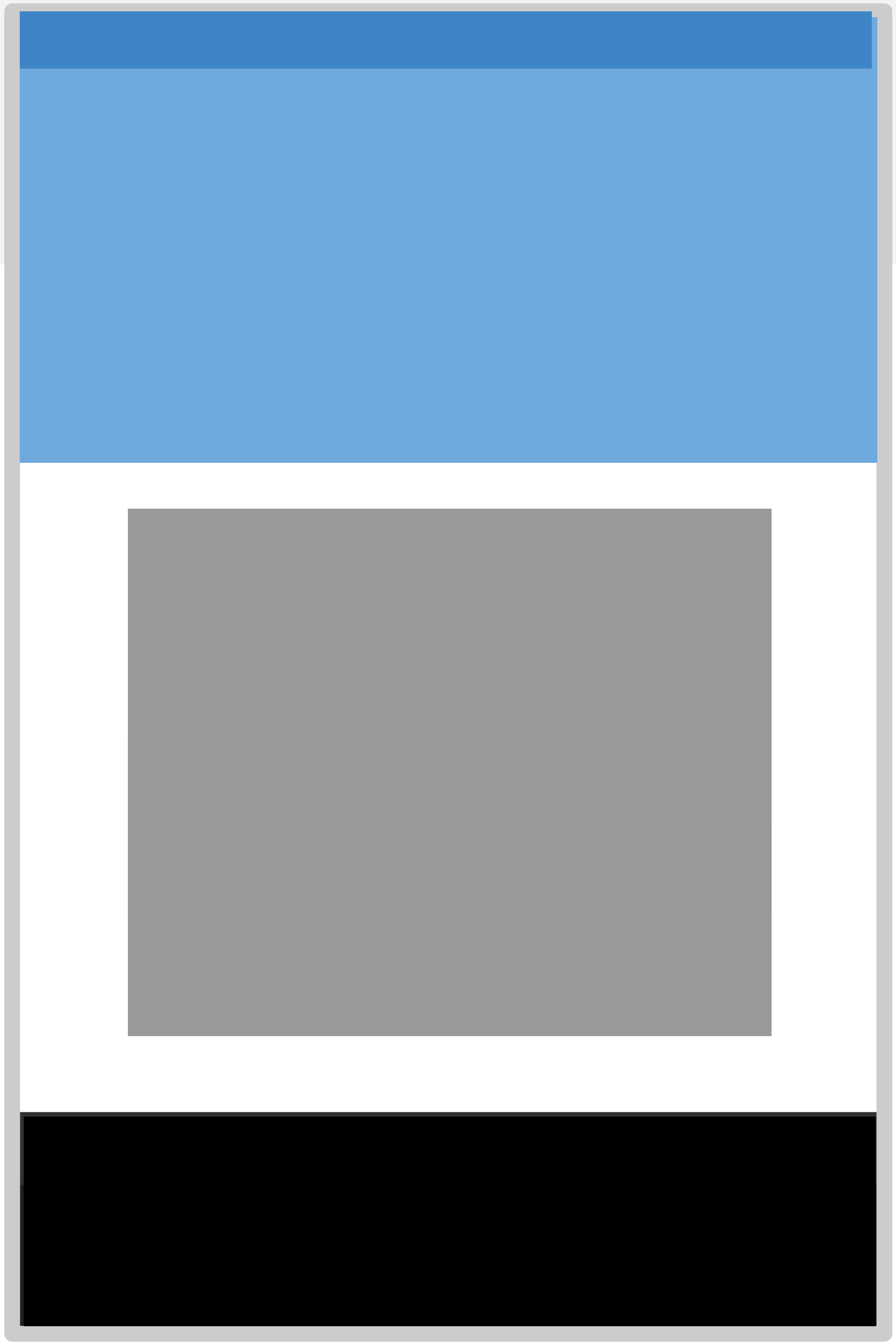
- Height: 400px;
- Background: lightskyblue
- <header>



Main section

Gray box:

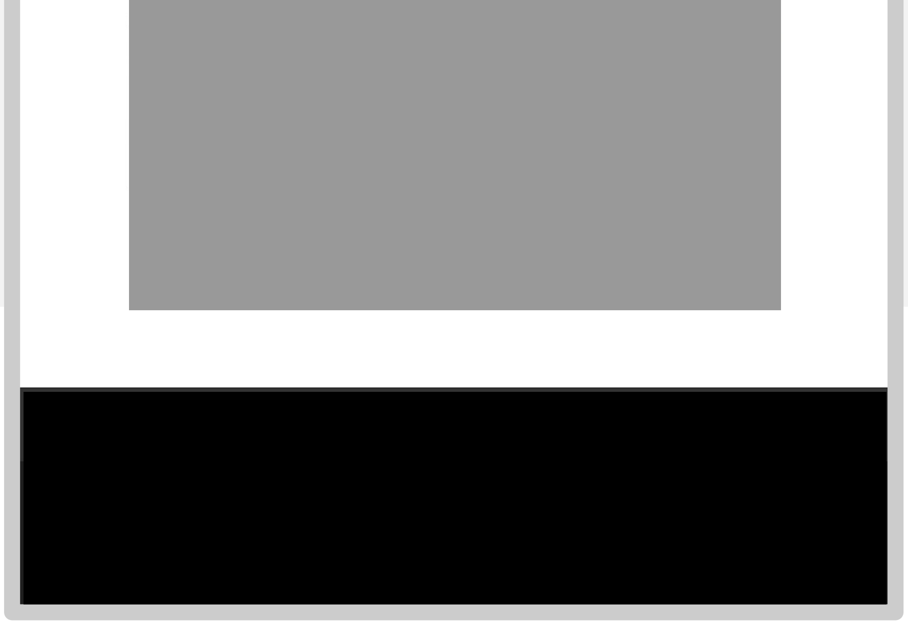
- Surrounding space: 75px above and below; 100px on each side
- Height: 500px
- Background: gray
- `<section>`



Footer

Footer:

- Height: 200px
- Background: Black
- `<footer>`



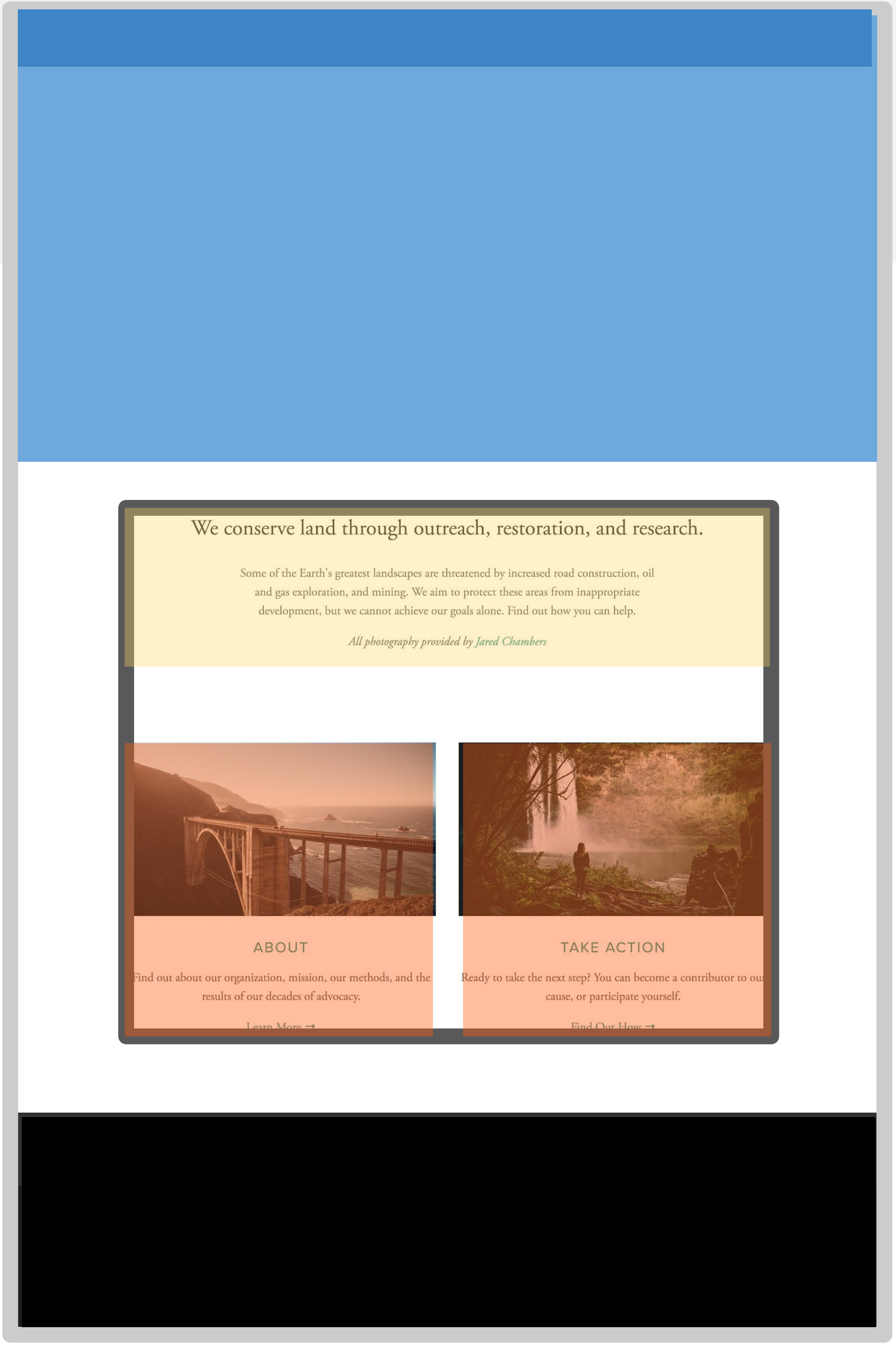
Main contents

Yellow paragraph:

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

Orange box:

- Height: 225x;
- Width: 48% of the parent's width, with space in between
- Background: tomato
- `<div>`



We conserve land through outreach, restoration, and research.

Some of the Earth's greatest landscapes are threatened by increased road construction, oil and gas exploration, and mining. We aim to protect these areas from inappropriate development, but we cannot achieve our goals alone. Find out how you can help.

All photography provided by Jared Chambers

ABOUT

Find out about our organization, mission, our methods, and the results of our decades of advocacy.

Learn More →

TAKE ACTION

Ready to take the next step? You can become a contributor to ou cause, or participate yourself.

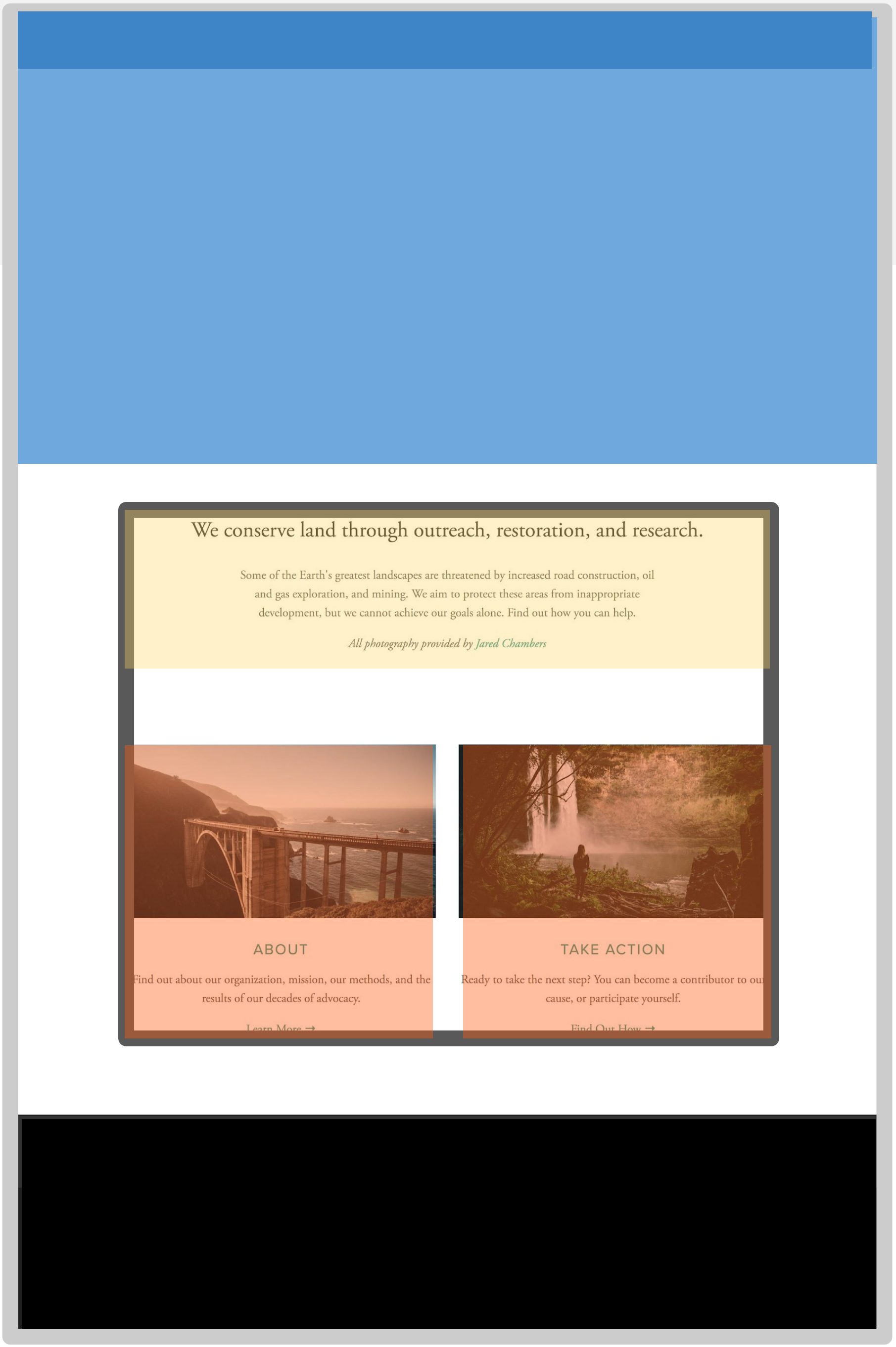
Find Out How →

Main contents

Orange box:

- Height: 225px;
- Width: 48% of the parent's width, with space in between
- Background: tomato
- `<div>`

This is where we get stuck.



We conserve land through outreach, restoration, and research.

Some of the Earth's greatest landscapes are threatened by increased road construction, oil and gas exploration, and mining. We aim to protect these areas from inappropriate development, but we cannot achieve our goals alone. Find out how you can help.

All photography provided by Jared Chambers

ABOUT

Find out about our organization, mission, our methods, and the results of our decades of advocacy.

Learn More →

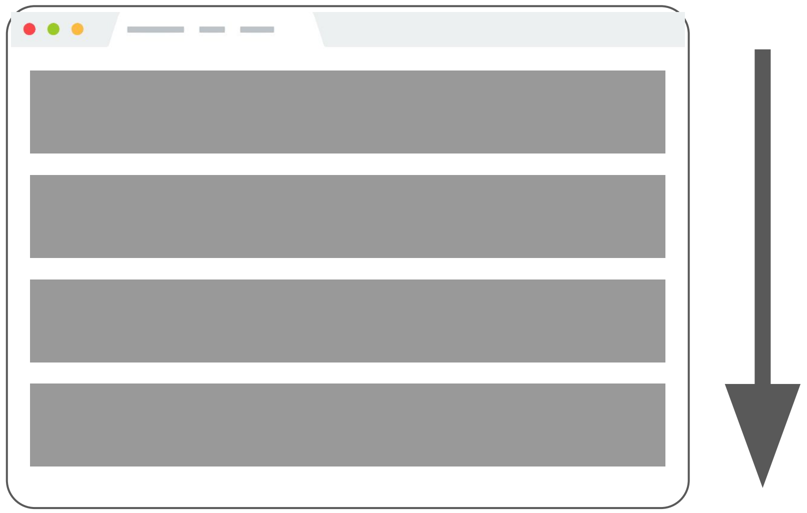
TAKE ACTION

Ready to take the next step? You can become a contributor to ou cause, or participate yourself.

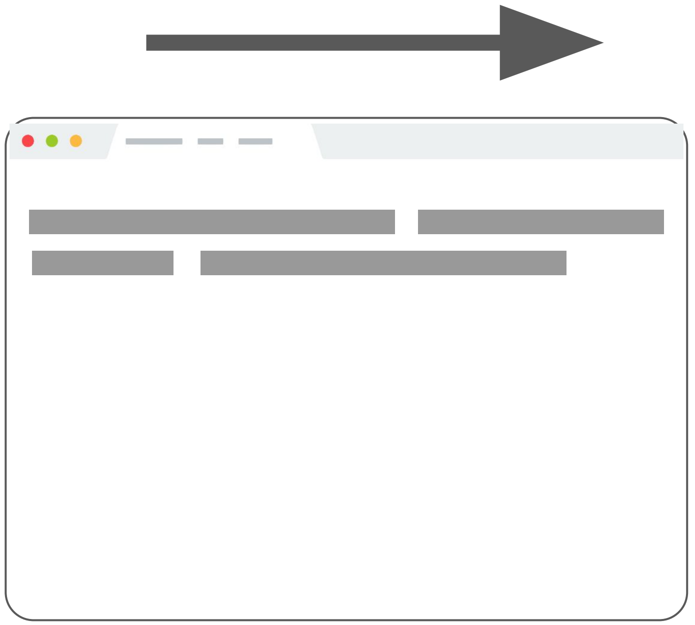
Find Out How →

Flexbox

CSS layout so far



Block layout:
Laying out large sections of a page

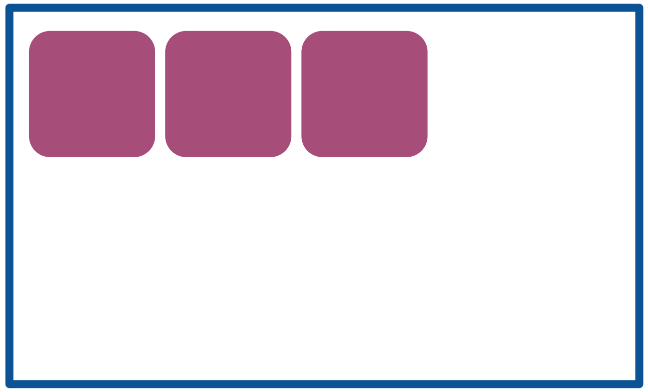


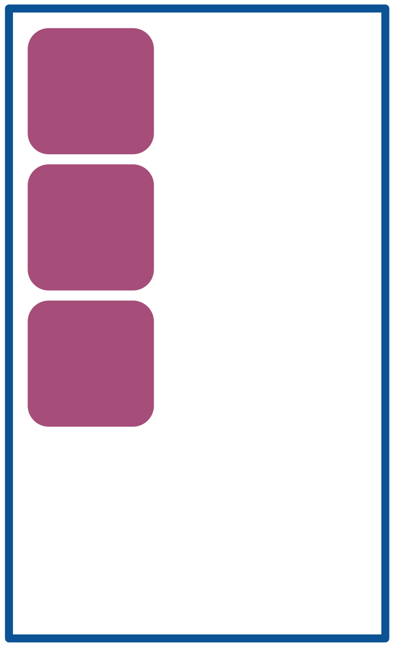
Inline layout:
Laying out text and other inline content within a section

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.

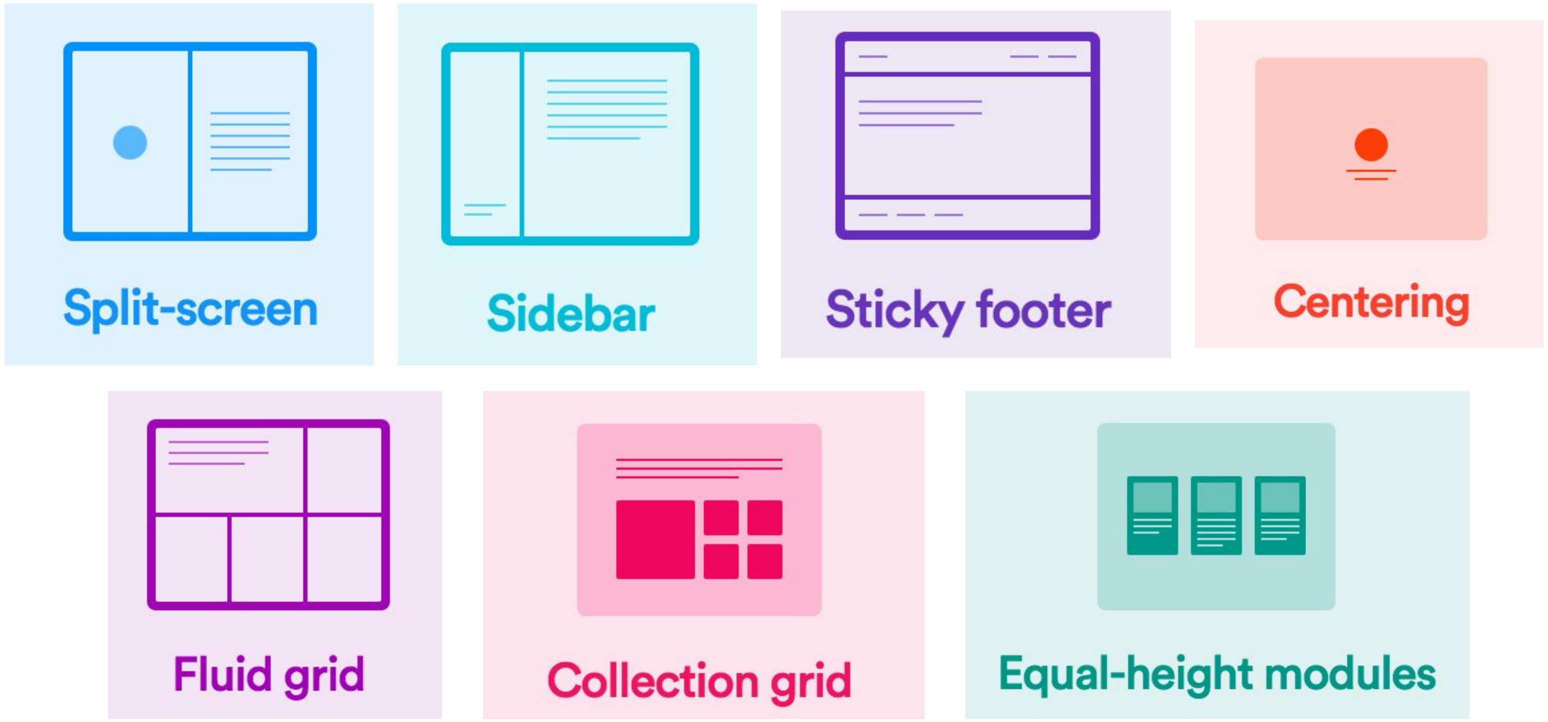




Flex layout

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):

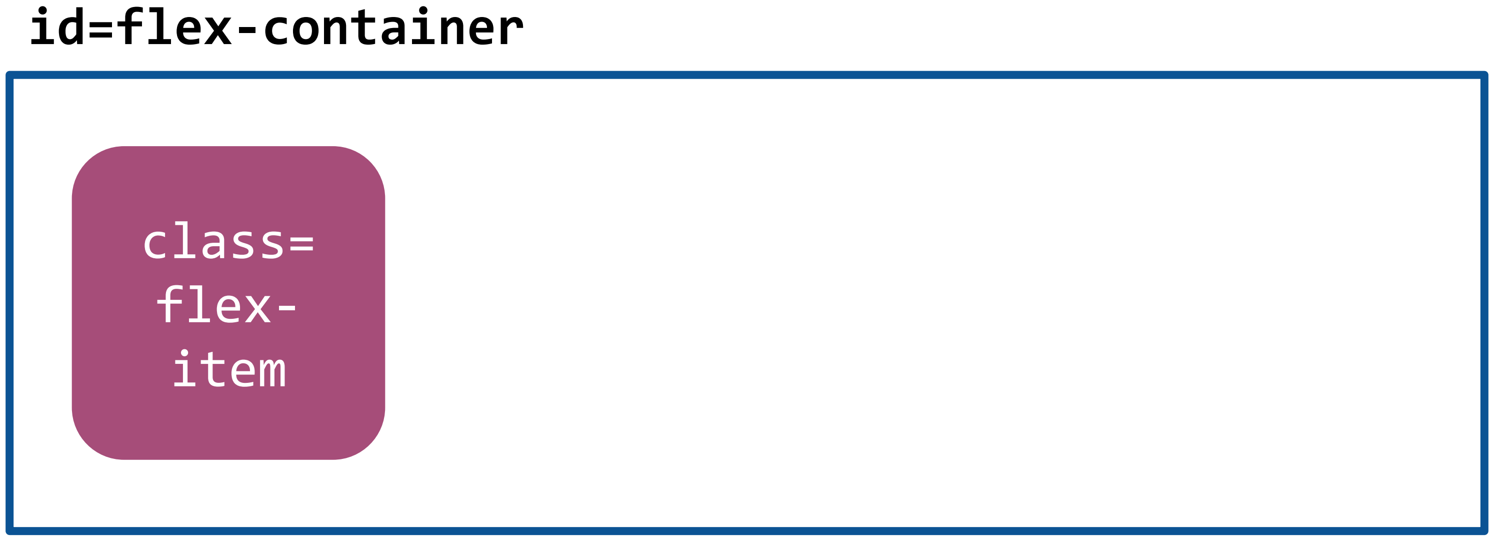


Flex basics

Flex layouts are composed of:

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

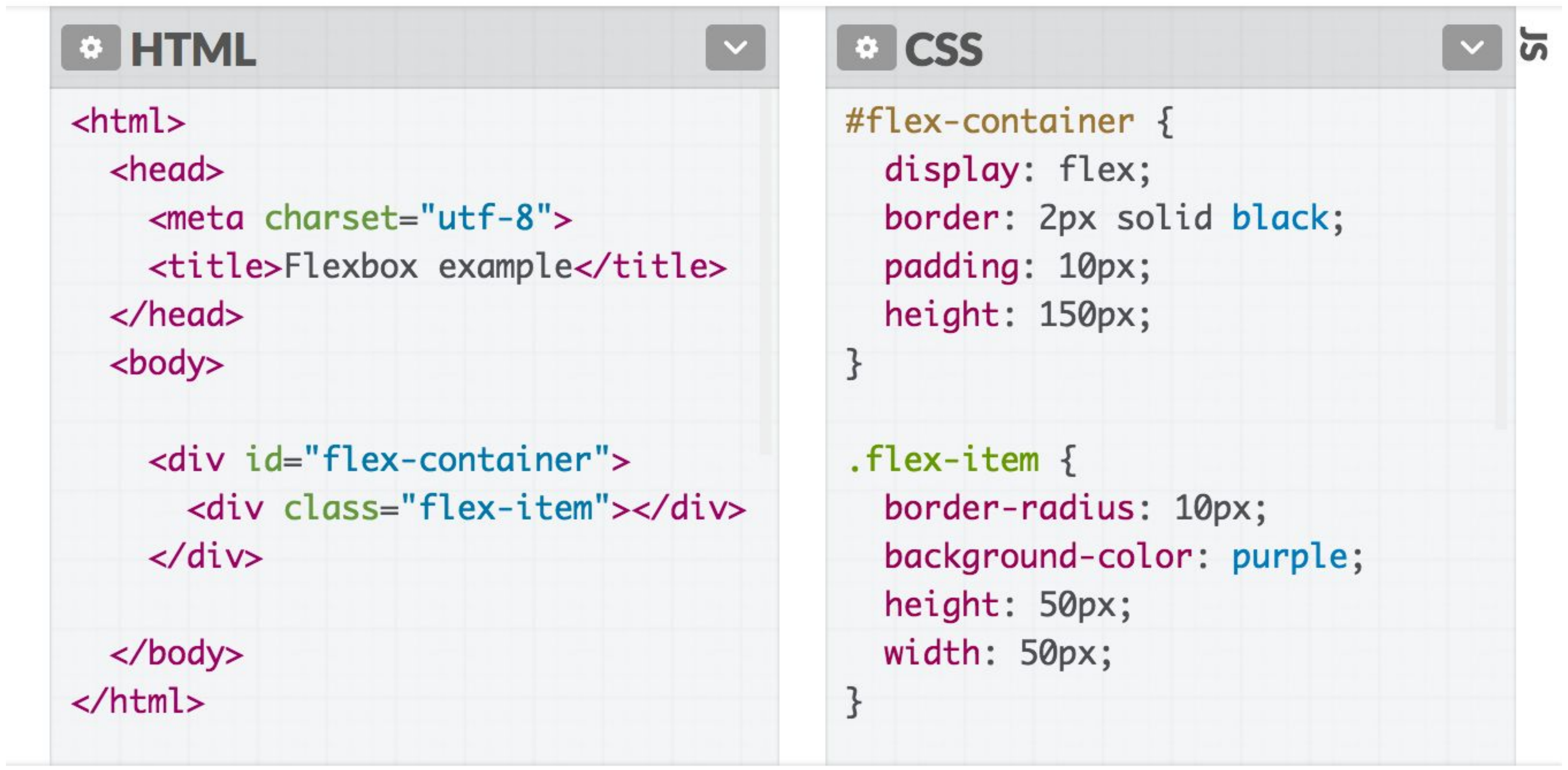
You can then apply CSS properties on the **flex container** to dictate how the flex items are displayed.



Flex basics

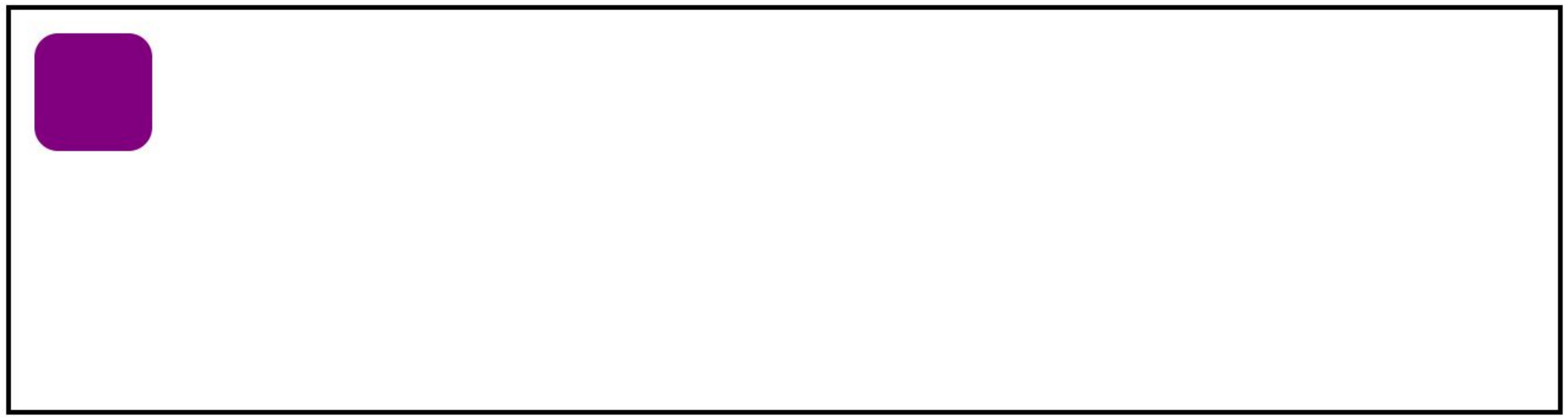
To make an element a flex container, change `display`:

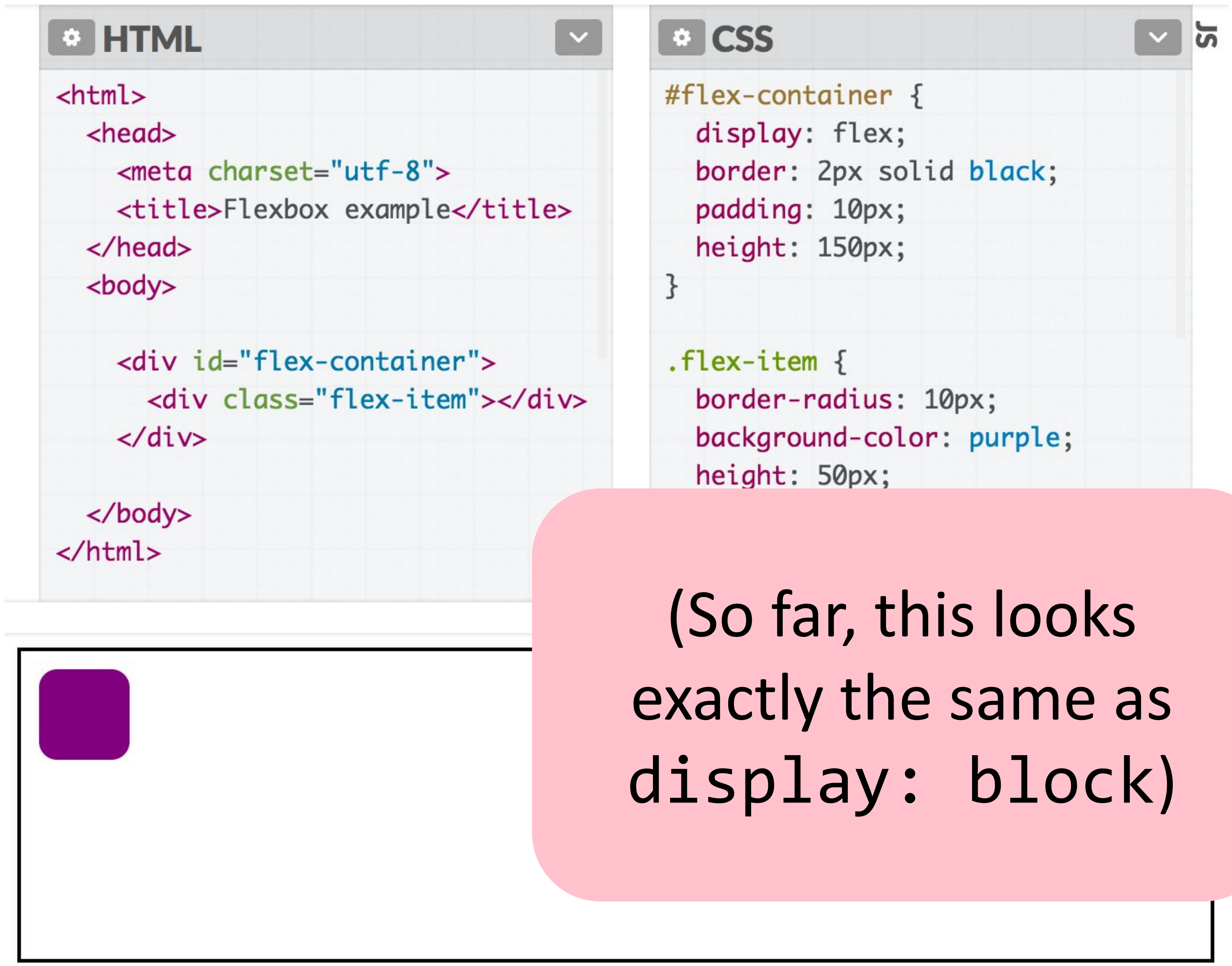
- Block container: `display: flex;` or
- Inline container: `display: inline-flex;`



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

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

  </body>
</html>
```

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

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

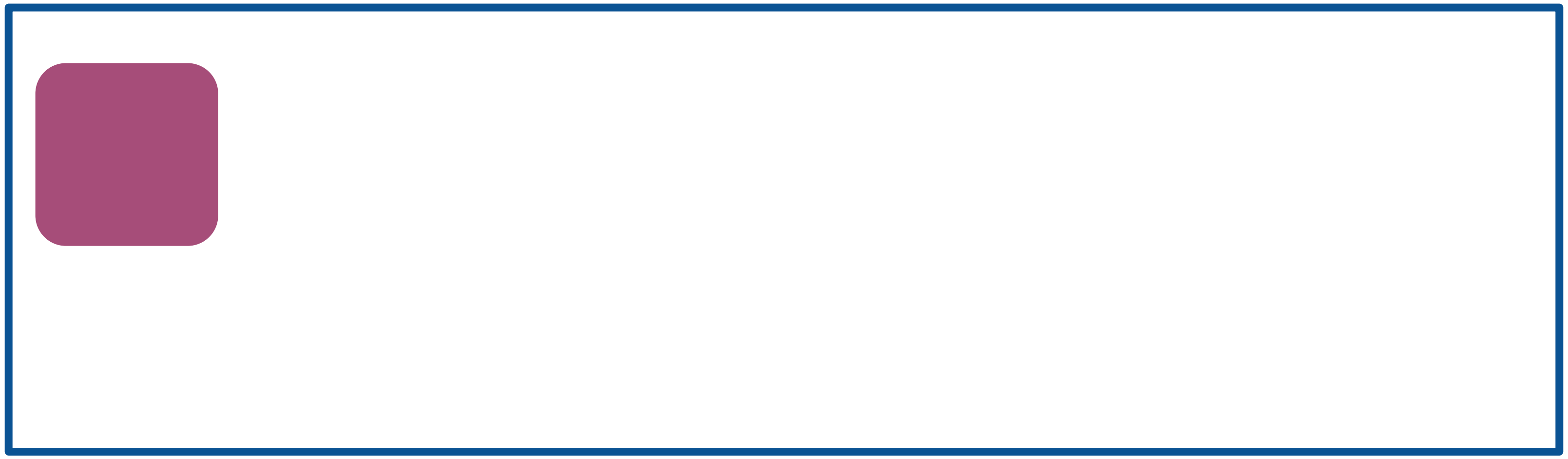
(So far, this looks exactly the same as `display: block`)

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.

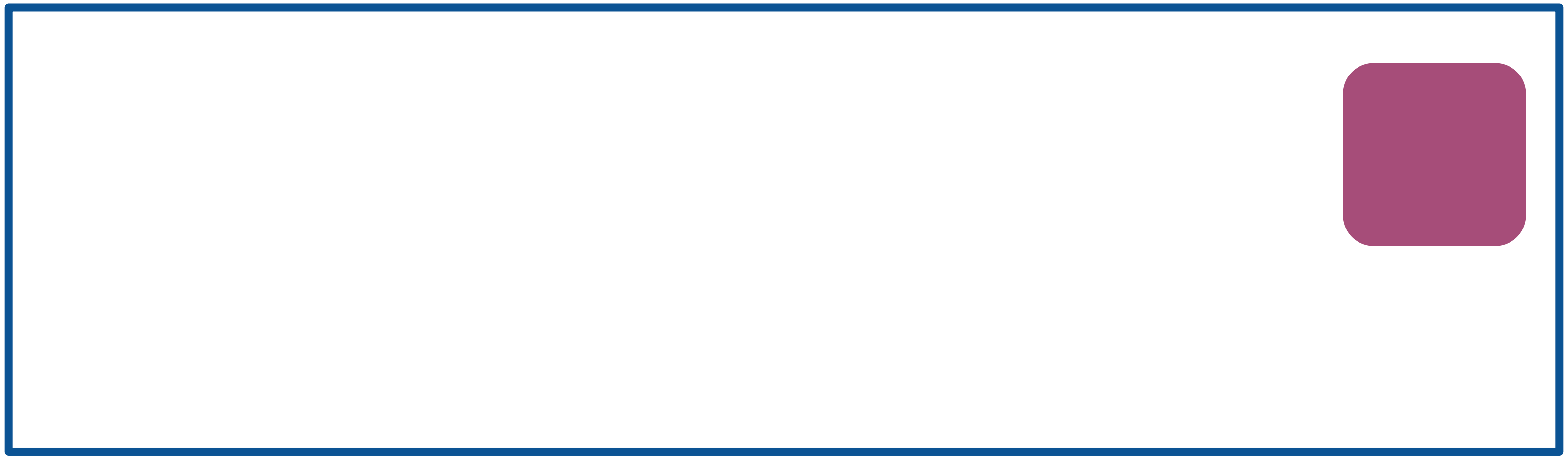


Flex basics: `justify-content`

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.

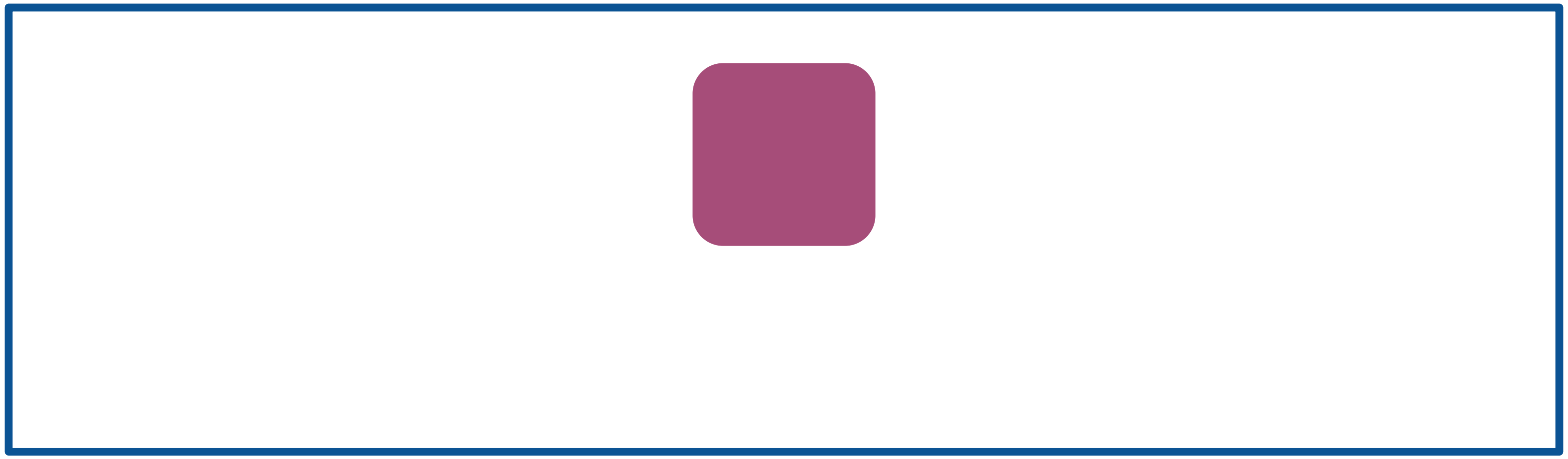


Flex basics: `justify-content`

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.

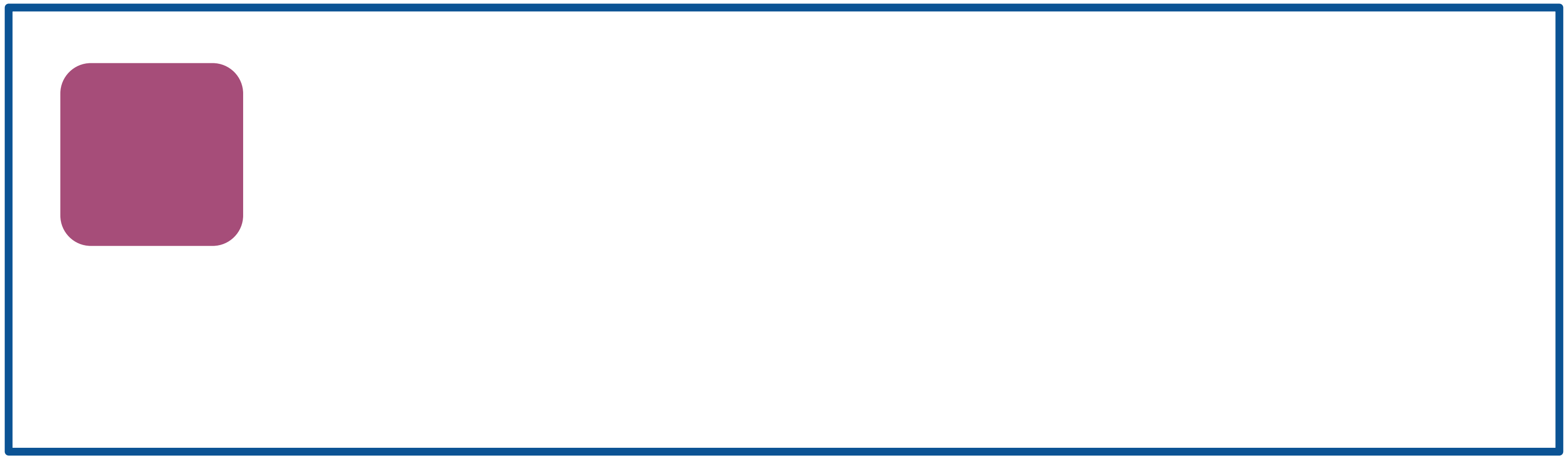


Flex basics: `align-items`

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.

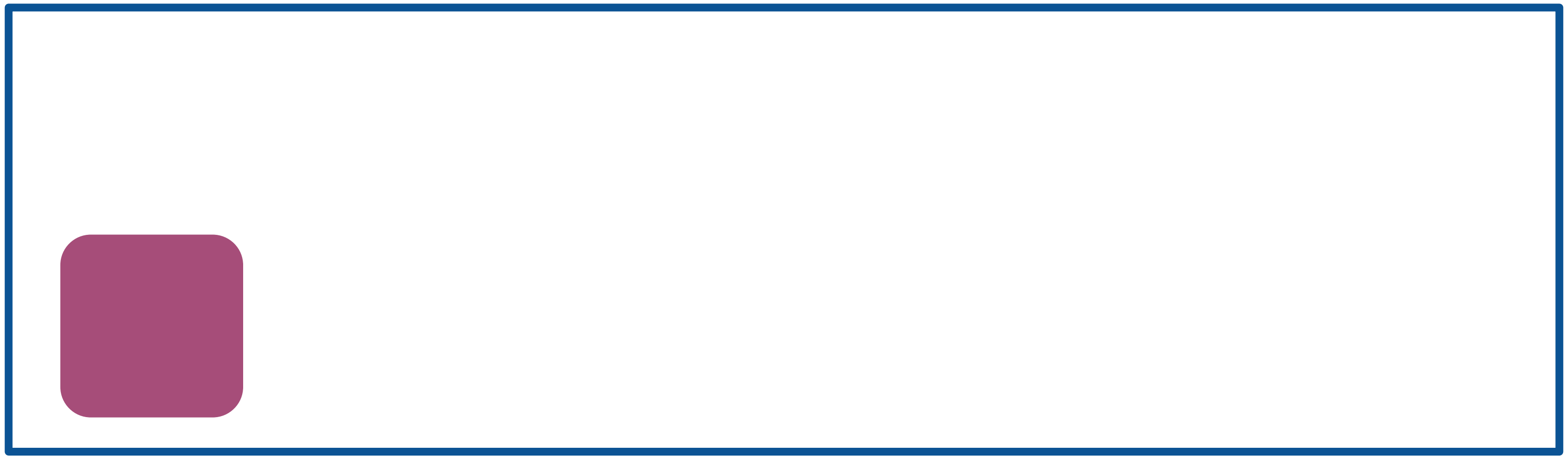


Flex basics: `align-items`

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.

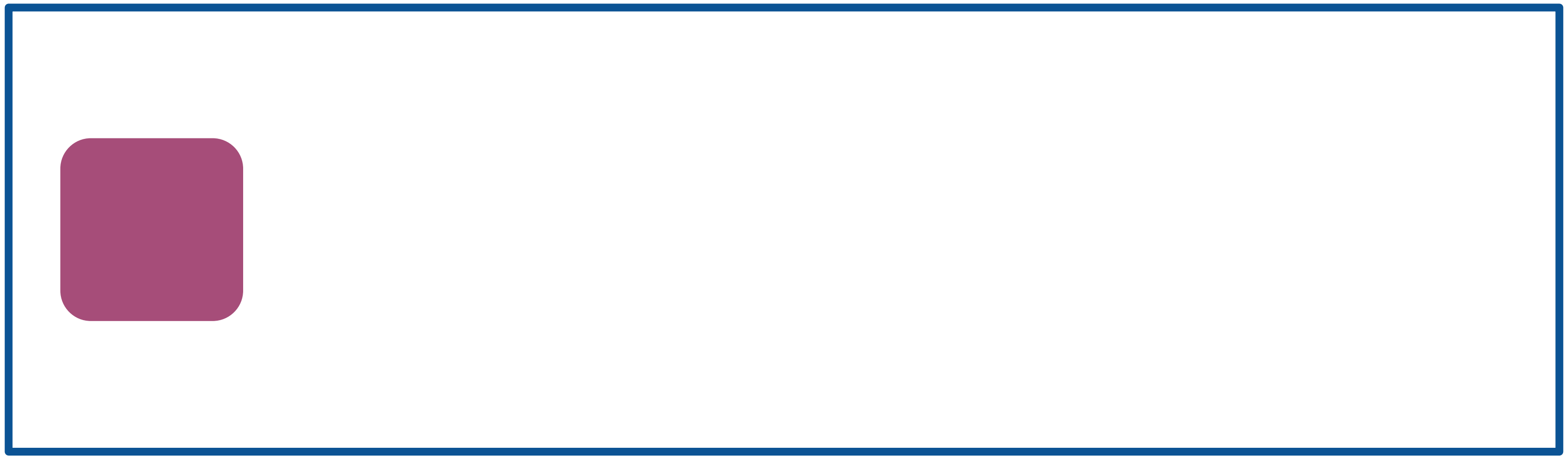


Flex basics: `align-items`

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;
}
```

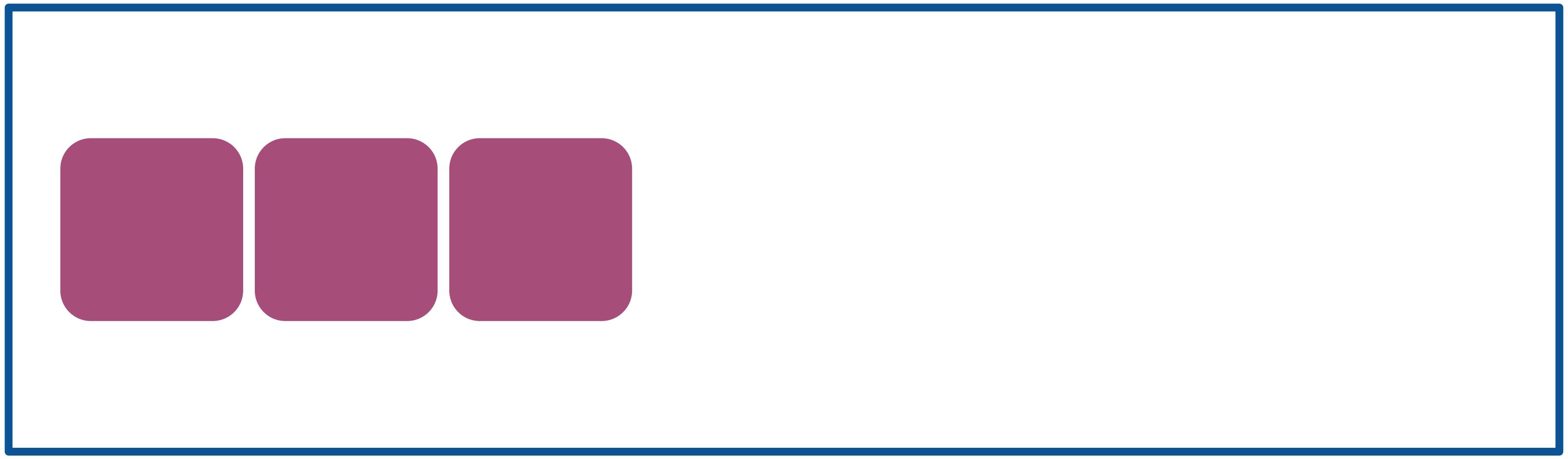
*when flex direction is `row`. We'll get to what "flex direction" means soon.



Multiple items

Same rules apply with multiple flex items:

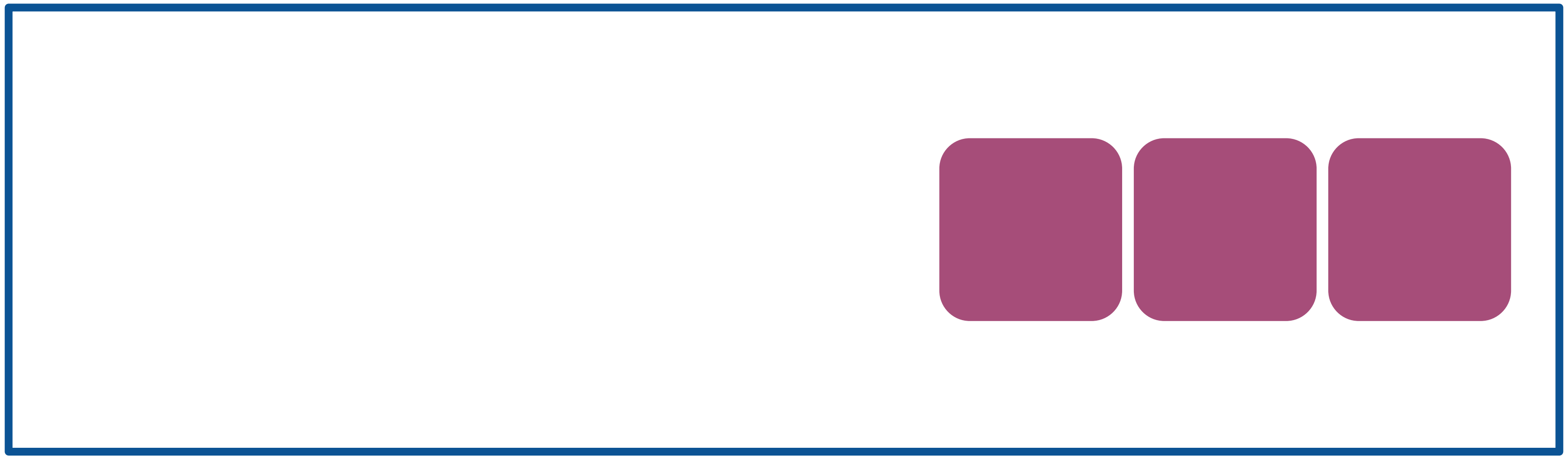
```
#flex-container {
  display: flex;
  justify-content: flex-start;
  align-items: center;
}
```



Multiple items

Same rules apply with multiple flex items:

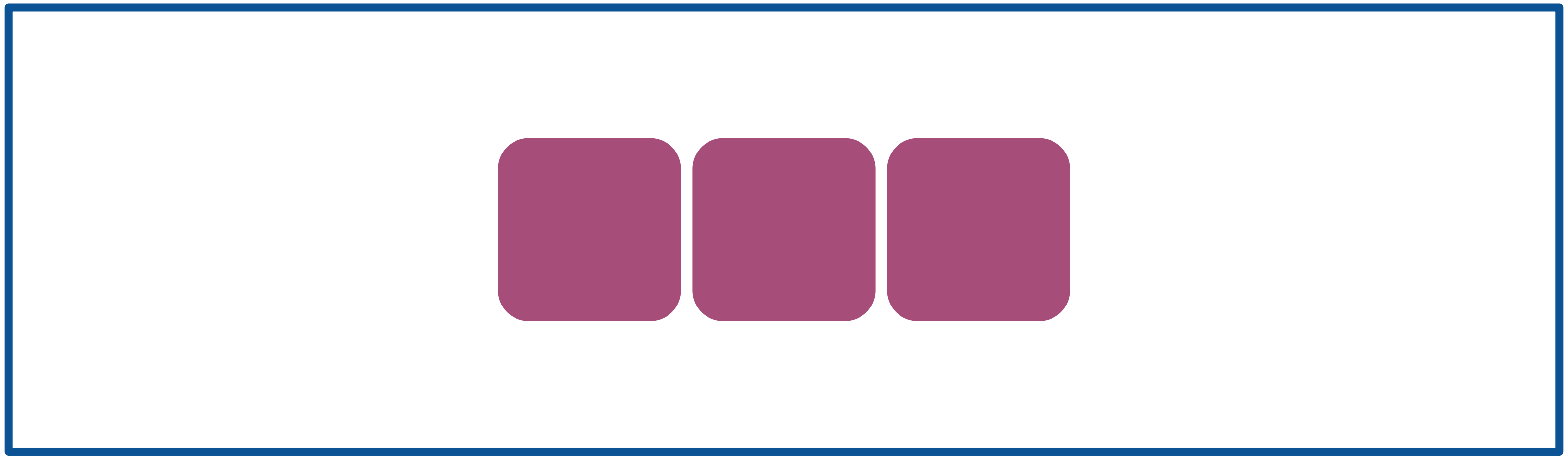
```
#flex-container {
  display: flex;
  justify-content: flex-end;
  align-items: center;
}
```



Multiple items

Same rules apply with multiple flex items:

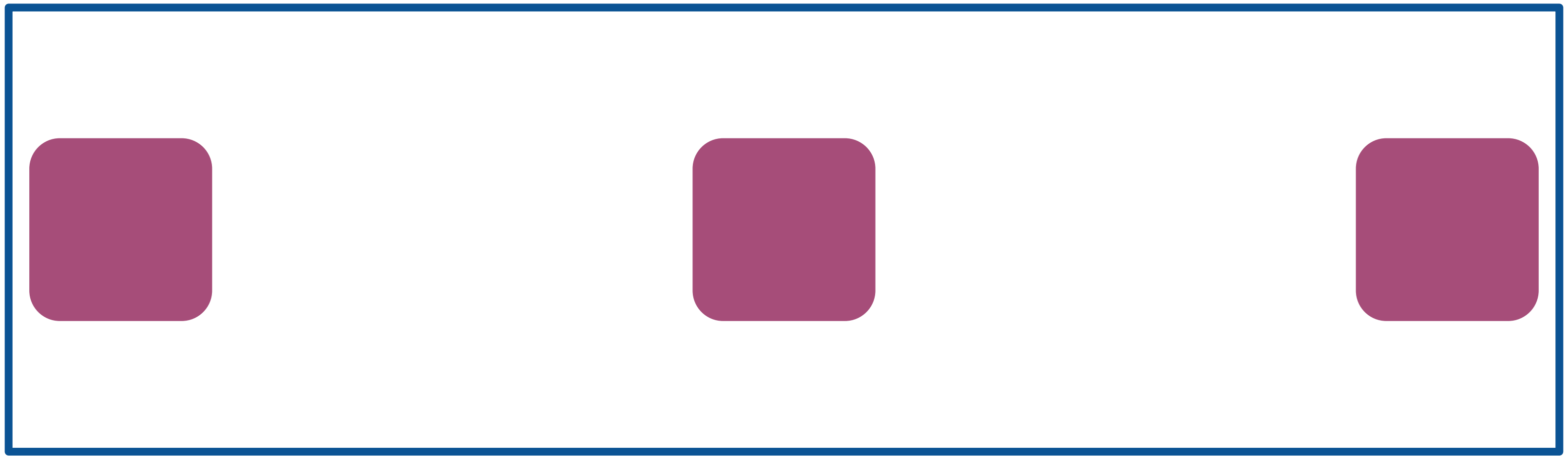
```
#flex-container {
  display: flex;
  Justify-content: center;
  align-items: center;
}
```



Multiple items

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

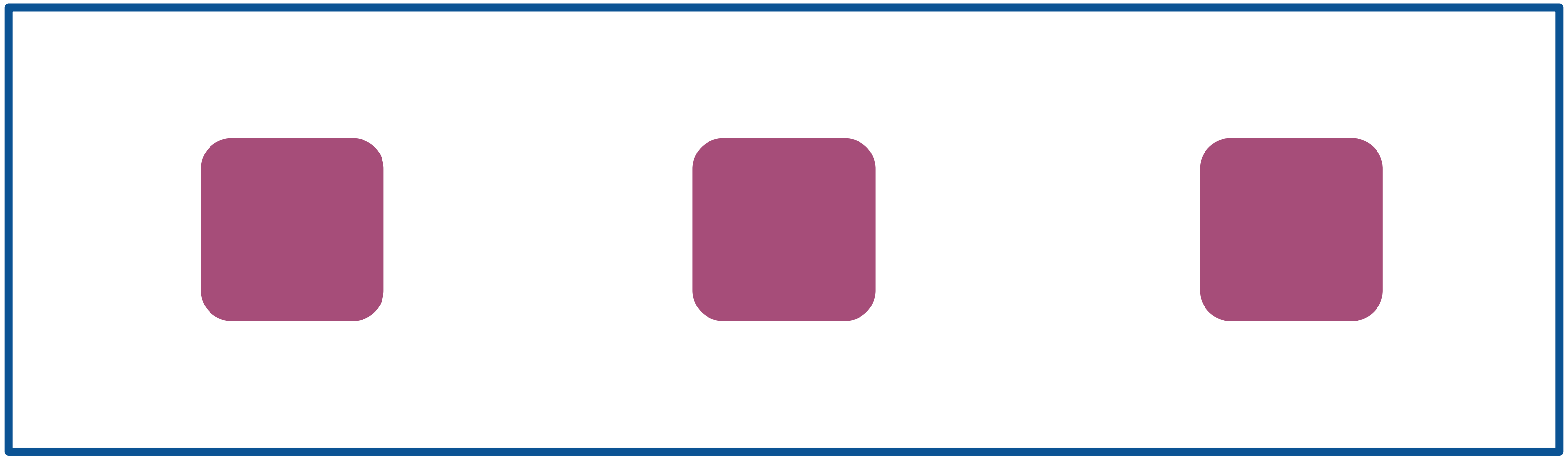
```
#flex-container {
  display: flex;
  Justify-content: space-between;
  align-items: center;
}
```



Multiple items

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

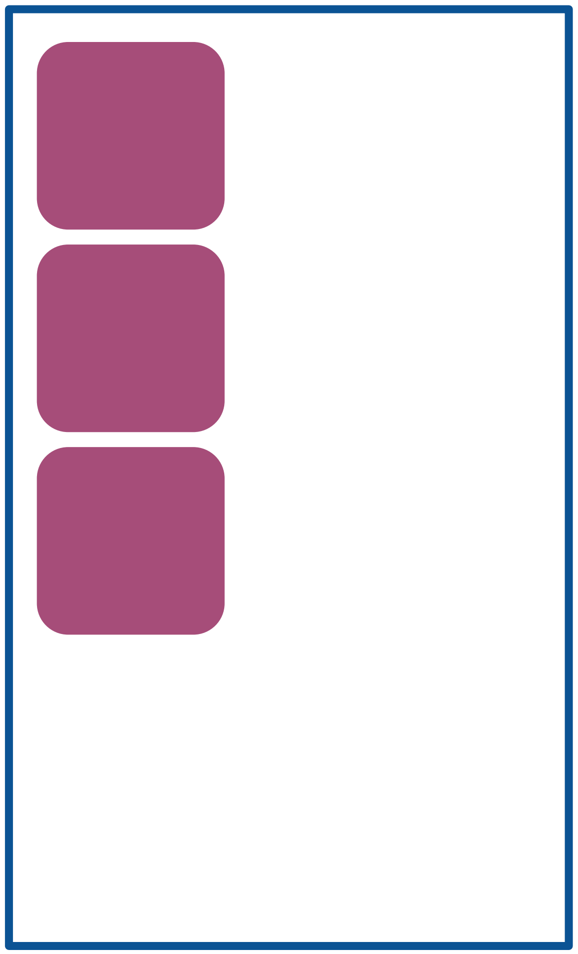
```
#flex-container {
  display: flex;
  Justify-content: space-around;
  align-items: center;
}
```



flex-direction

And you can also lay out columns instead of rows:

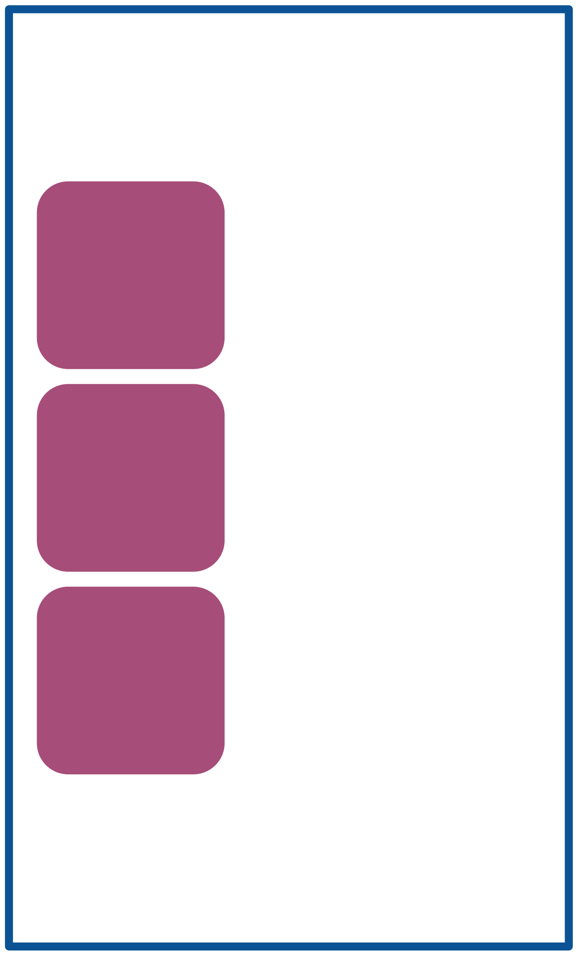
```
#flex-container {
  display: flex;
  flex-direction: column;
}
```



`flex-direction`

And you can also lay out columns instead of rows:

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

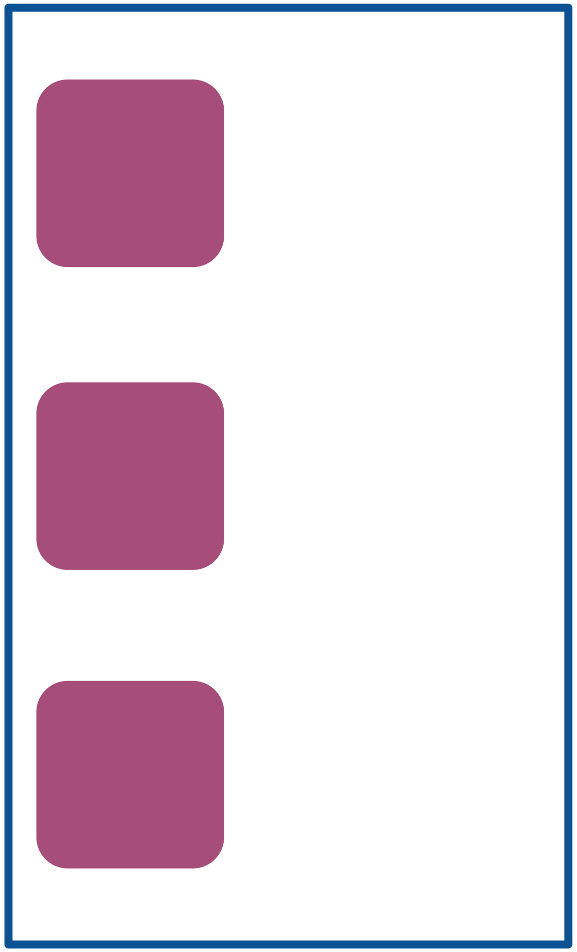


Now **`justify-content`** controls where the column is vertically in the box

flex-direction

And you can also lay out columns instead of rows:

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

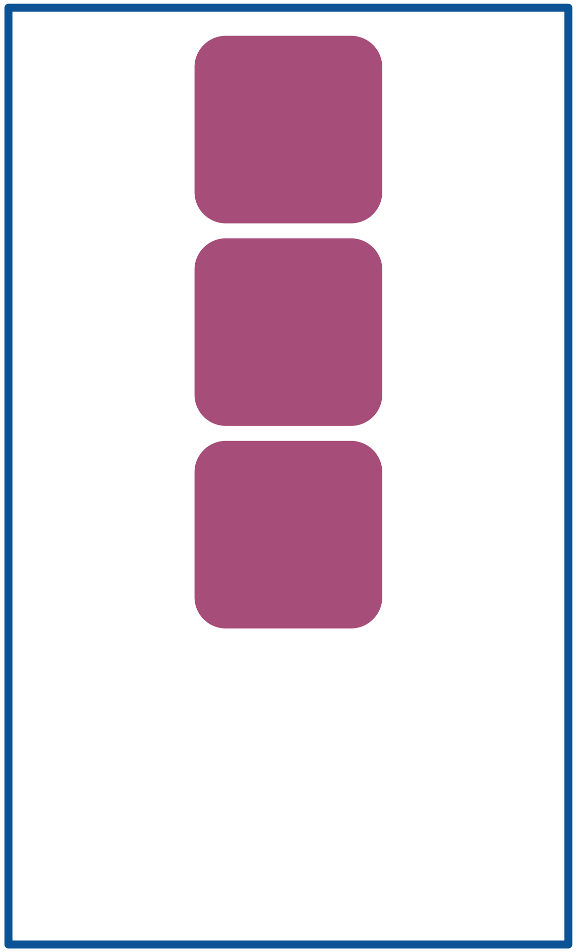


Now **justify-content** controls where the column is vertically in the box

`flex-direction`

And you can also lay out columns instead of rows:

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

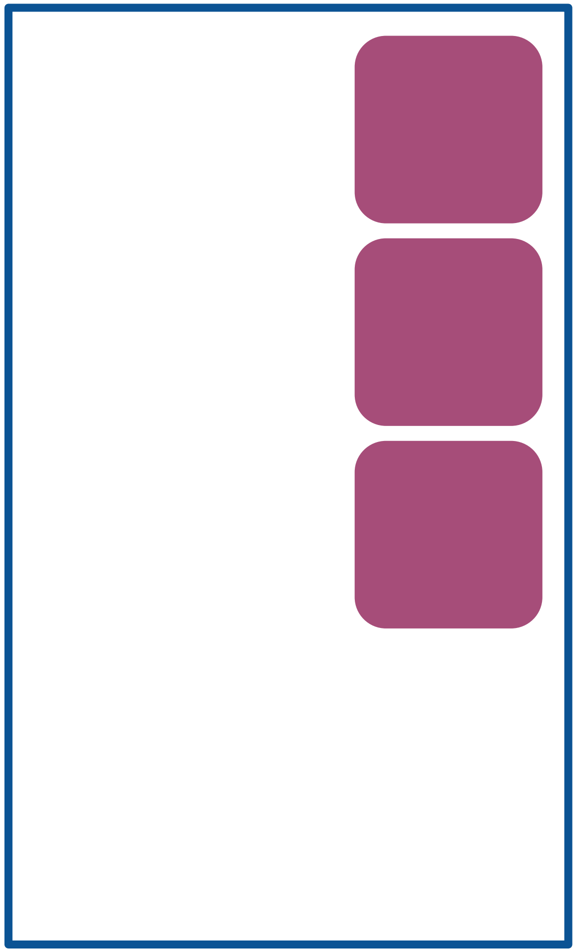


Now **`align-items`** controls where the column is horizontally in the box

flex-direction

And you can also lay out columns instead of rows:

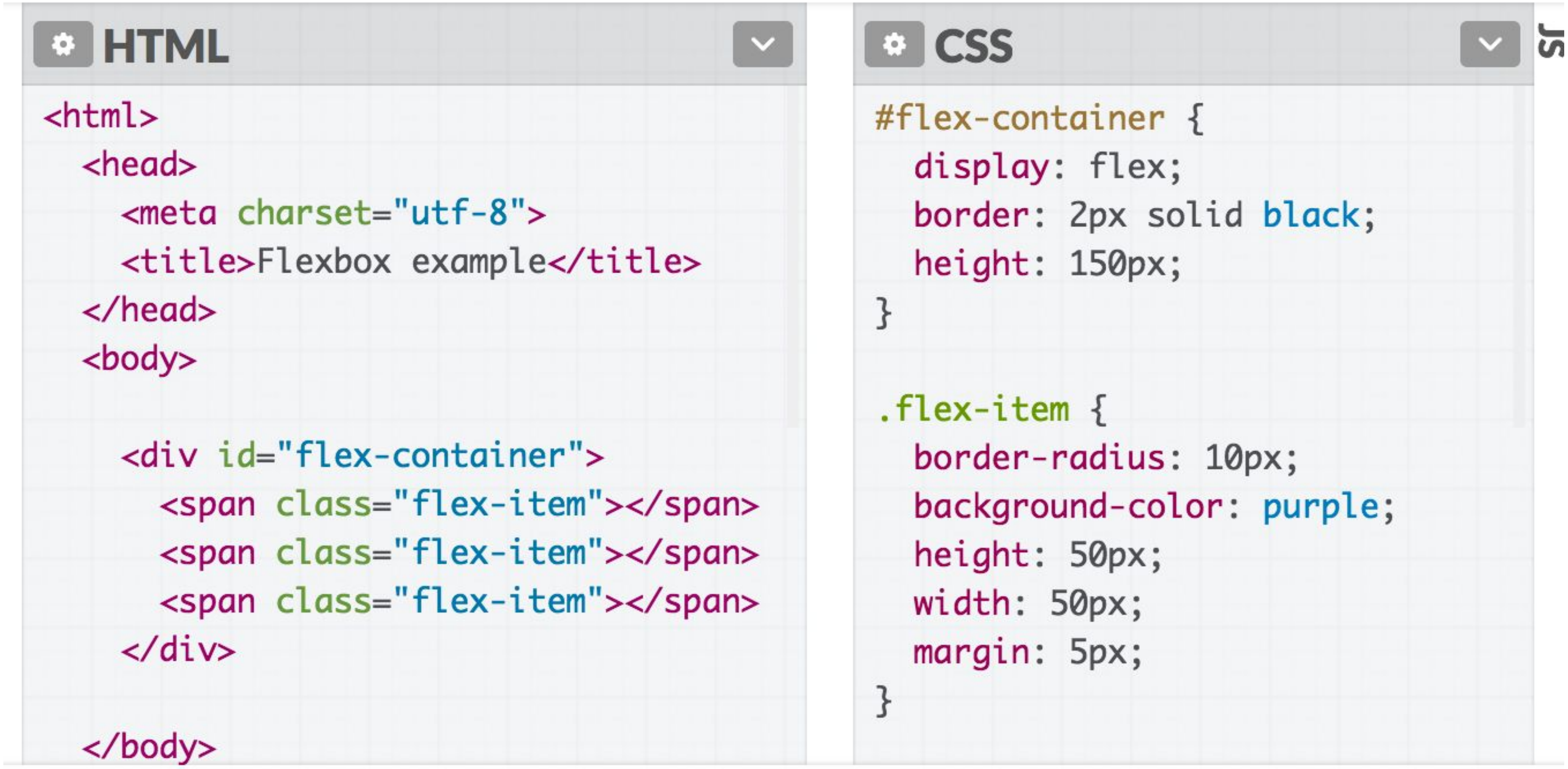
```
#flex-container {
  display: flex;
  flex-direction: column;
  align-items: flex-end;
}
```



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

Before we move on…

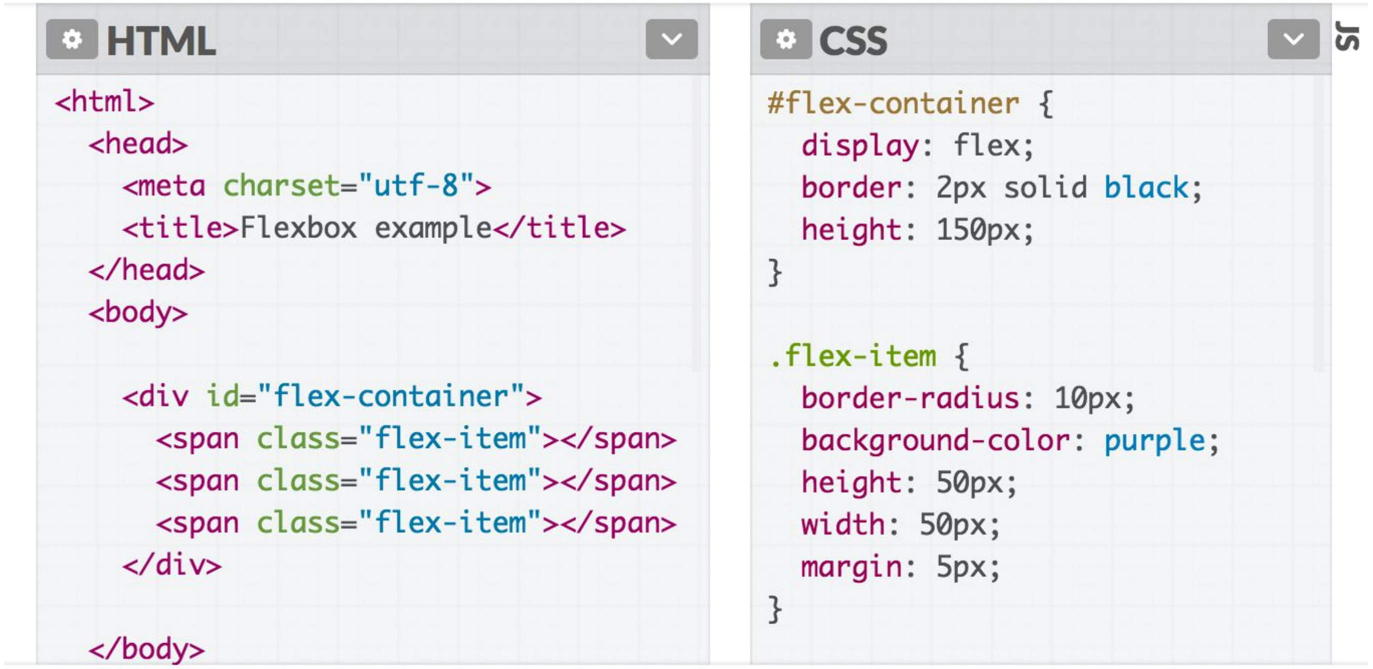
What happens if the flex item is an inline element?



```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>
```

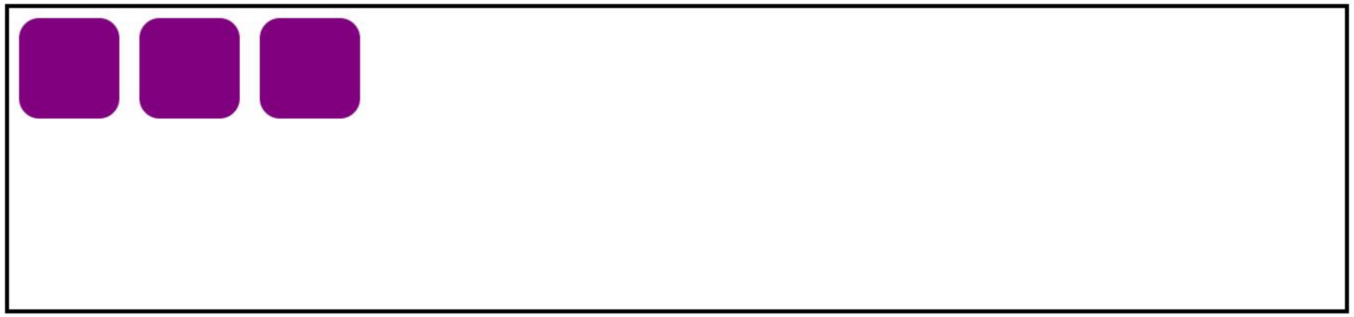
```css
#flex-container {
  display: flex;
  border: 2px solid black;
  height: 150px;
}

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

???



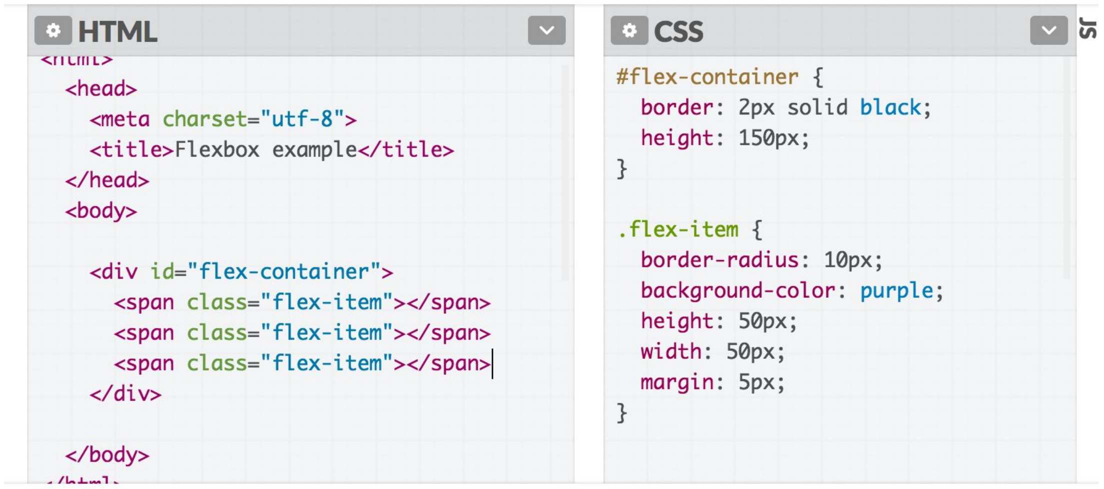
```
<html>
  <head>
    <meta charset="utf-8">
    <title>Flexbox example</title>
  </head>
  <body>

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

  </body>
```

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

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



Recall: block layouts

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

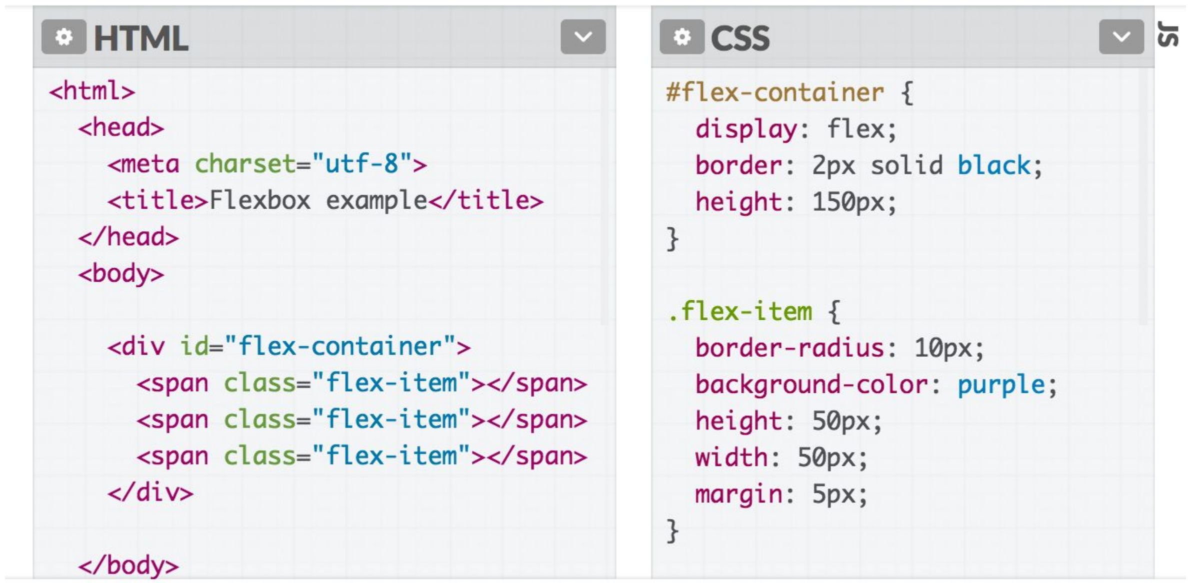


```html
<html>
  <head>
    <meta charset="utf-8">
    <title>Flexbox example</title>
  </head>
  <body>

    <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;
}
```

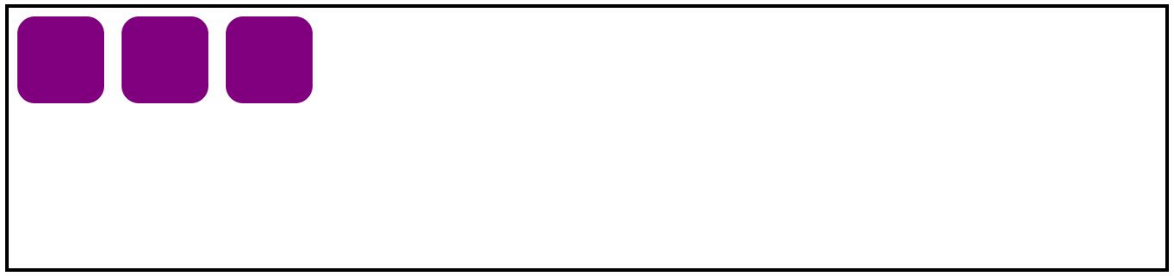
Then the span `flex-items` would not show up because span elements are inline, which don't have a height and width

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?

More next time!