CSE 309

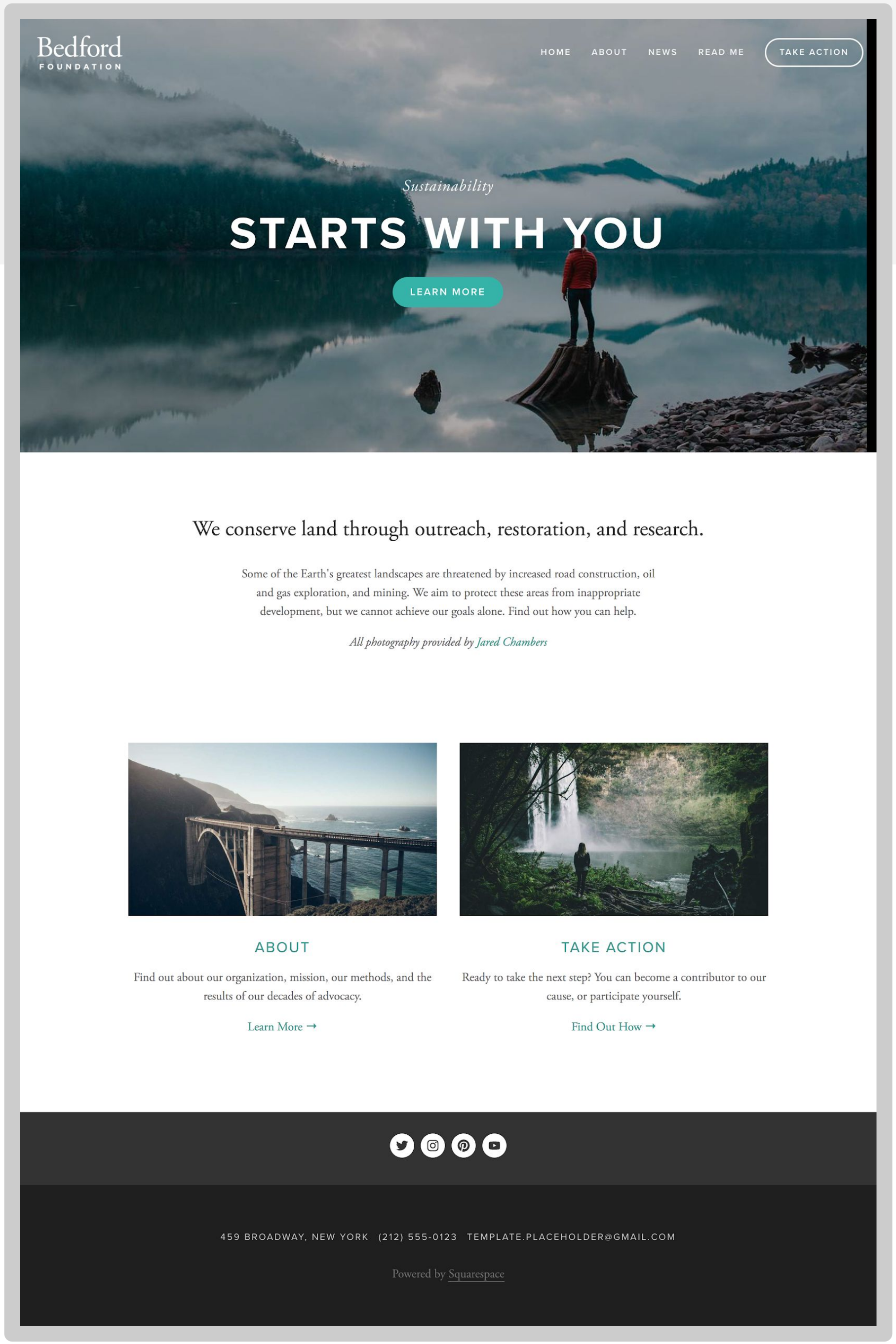
Web Applications and Internet

Sanzar Adnan Alam
sanzar@iub.edu.bd

Layout exercise

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 our cause, or participate yourself.

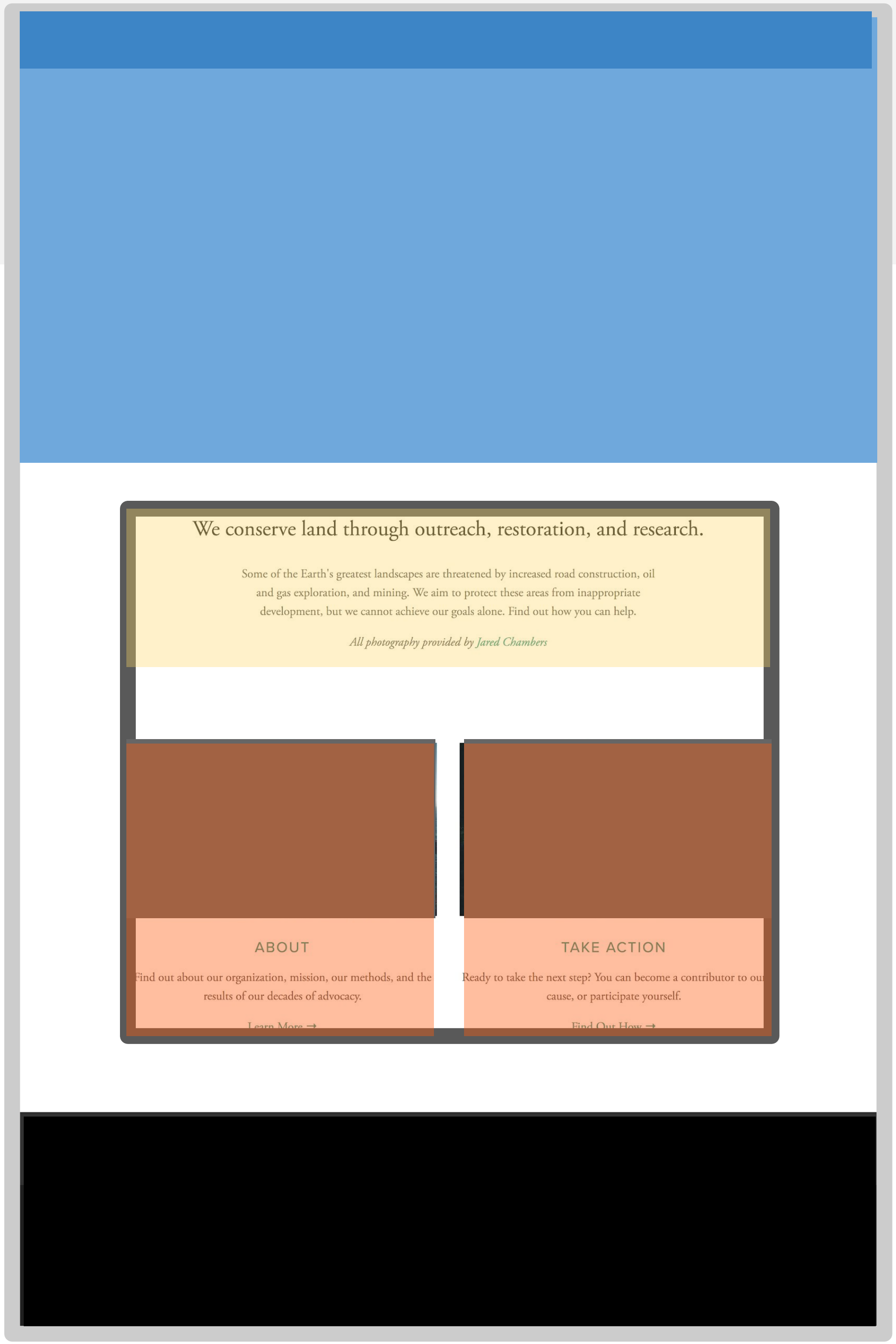
Find Out How →

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.

TAKE ACTION

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

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>`	Introdu
`<footer>`	Footer a
`<nav>`	Navigati

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

These elements do not "do" anyt

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

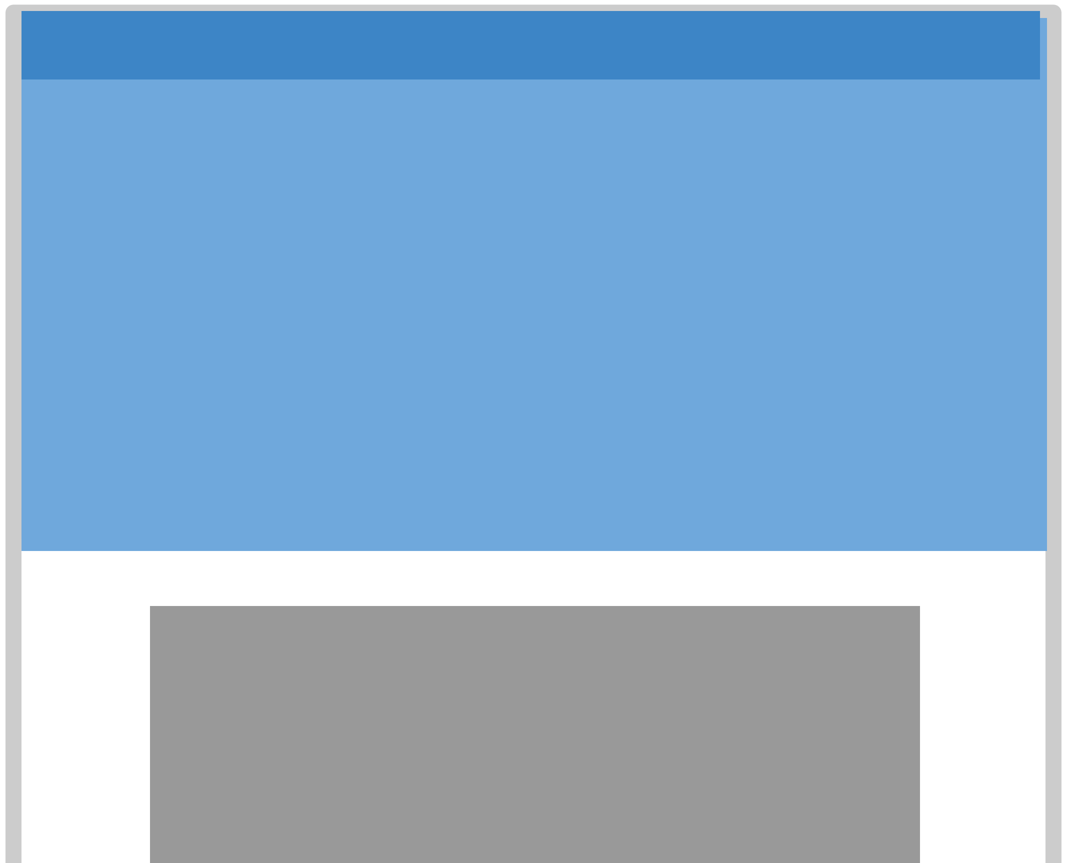
Header

Navbar:

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

Header:

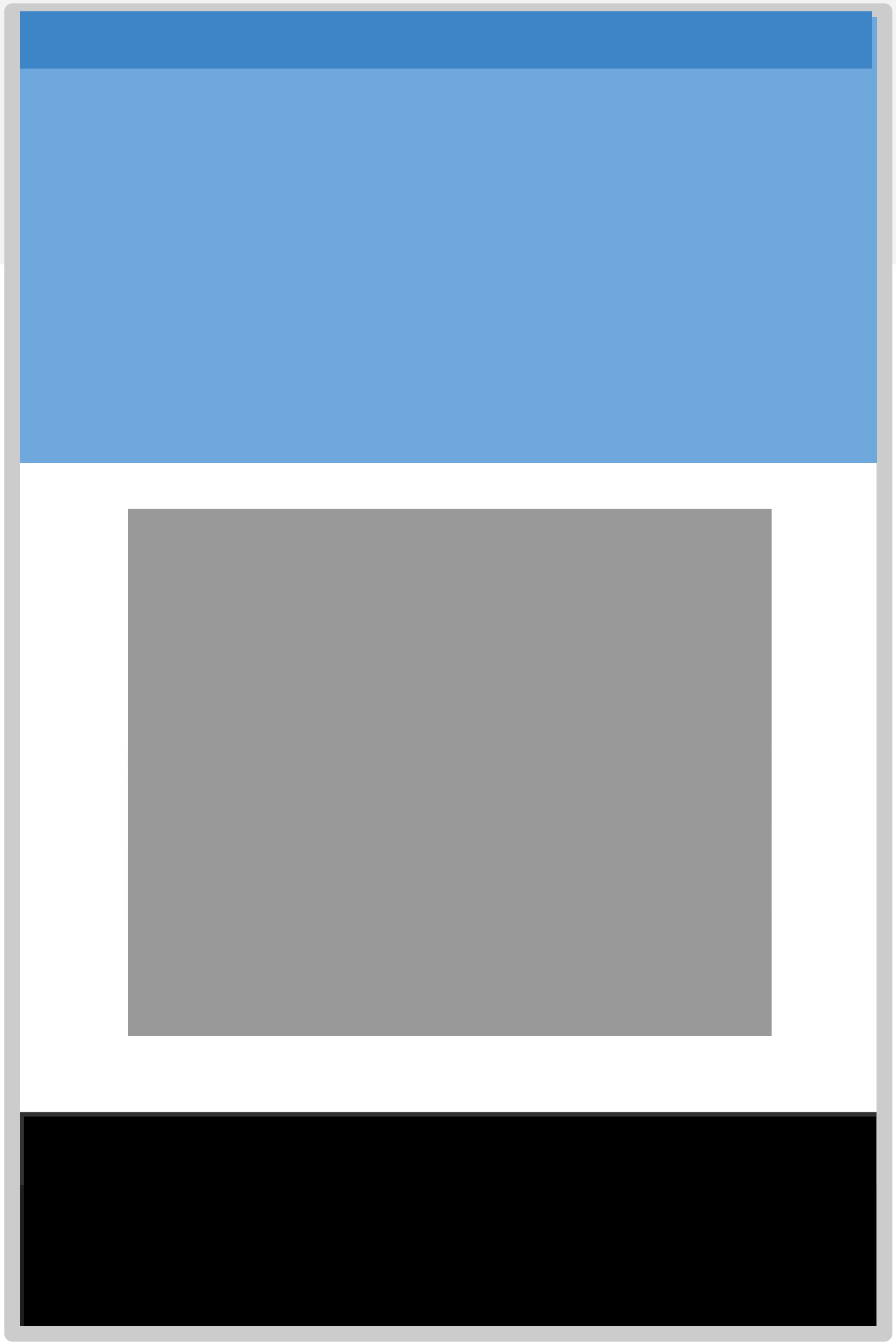
- Height: 400px;
- Background: lightskyblue
- `<header>`



Main section

Gray box:

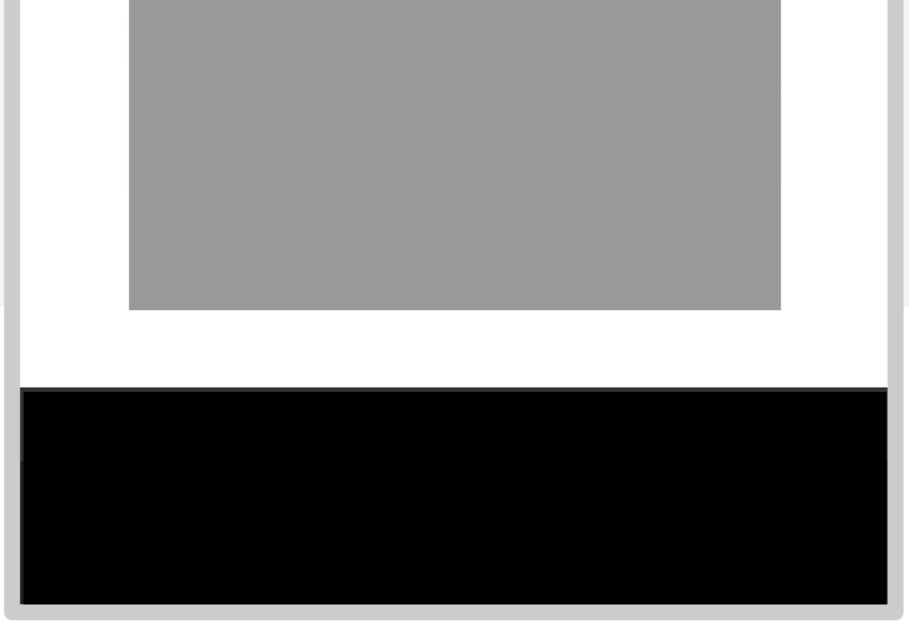
- Surrounding space: 75px above and below; 100px on each side
- Height: 500px
- Background: gray
- `<section>`



Footer

Footer:

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



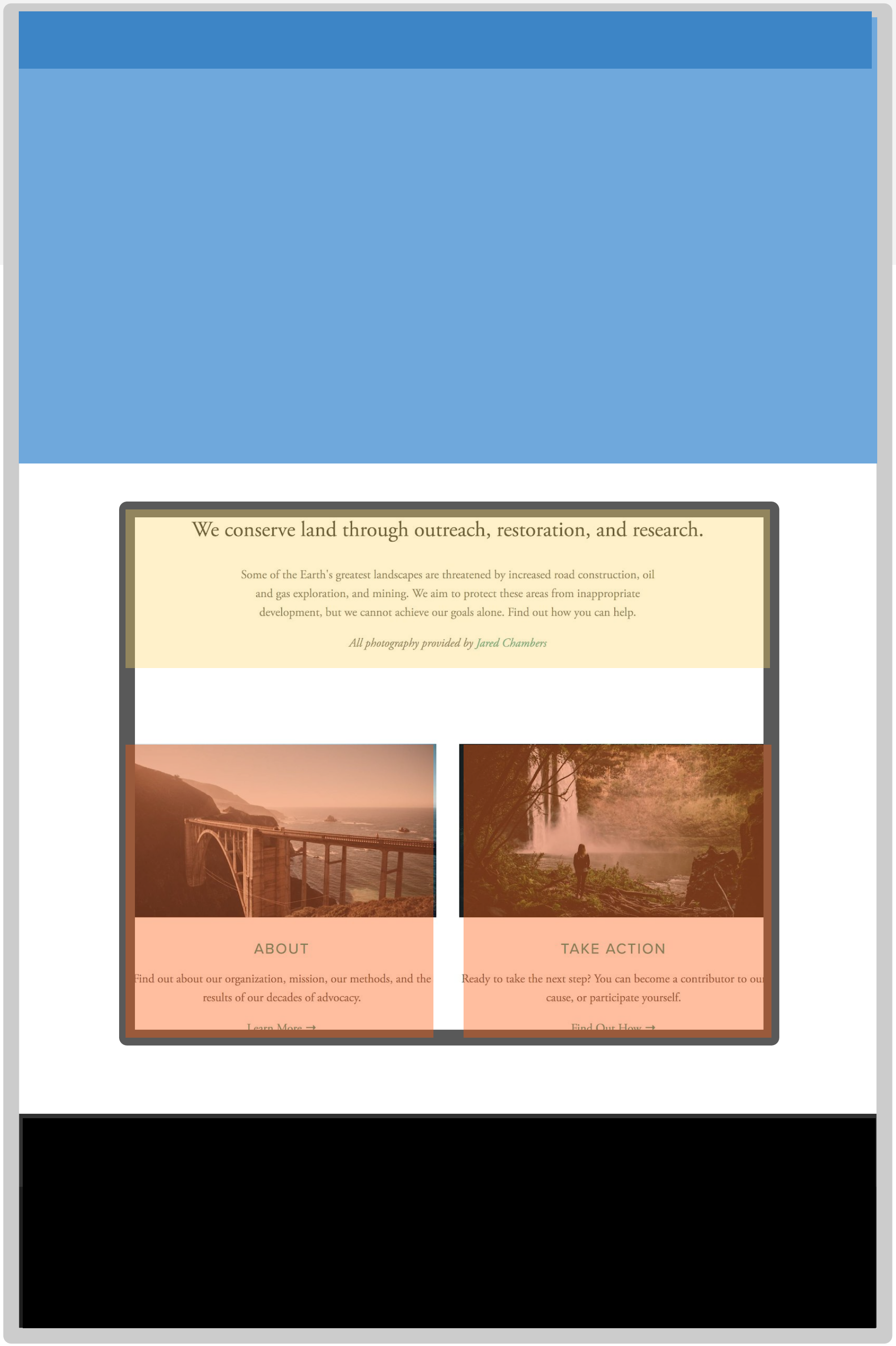
Main contents

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>`



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 →

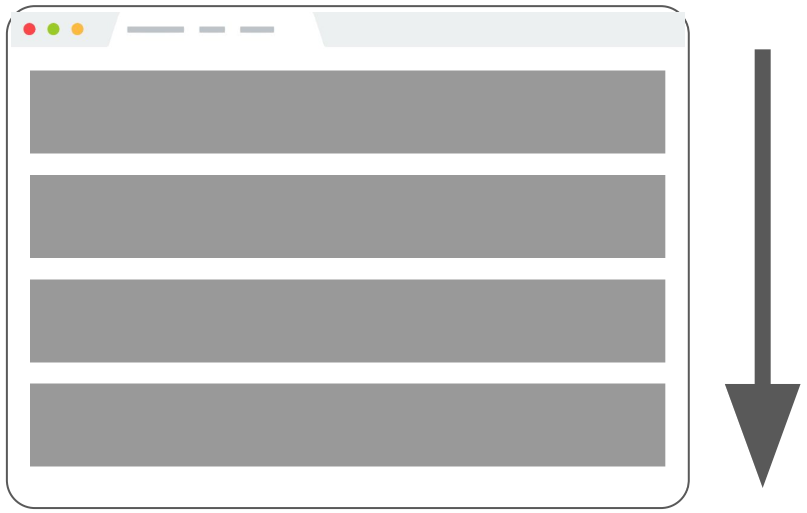
TAKE ACTION

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

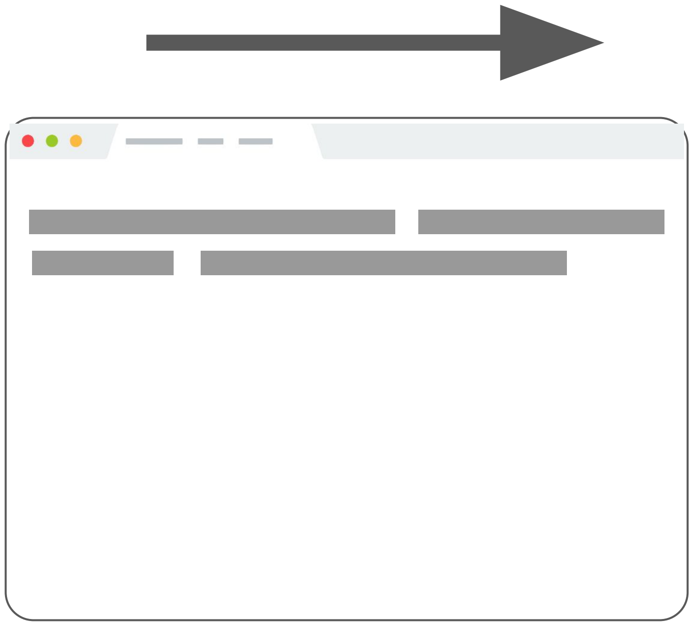
Find Out How →

Flexbox

CSS layout so far



Block layout:
Laying out large sections of a page

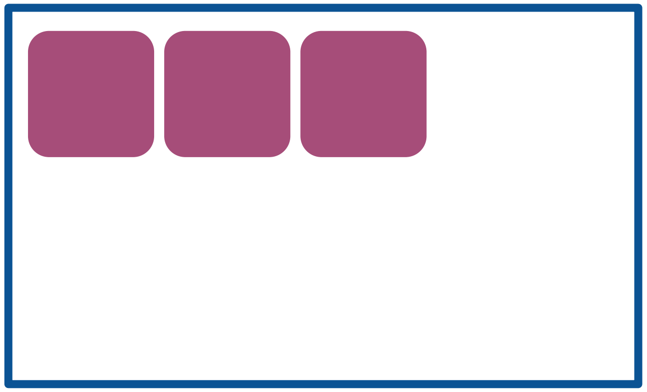


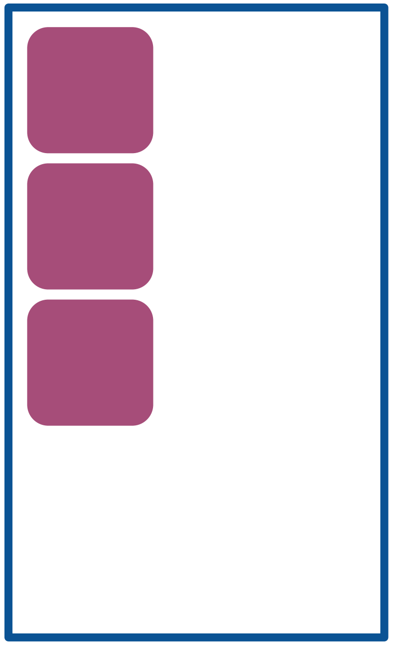
Inline layout:
Laying out text and other inline content within a section

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.

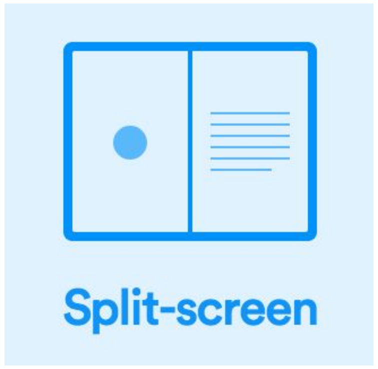


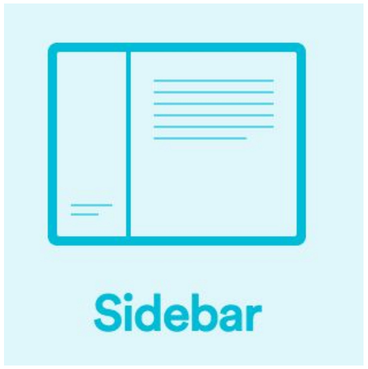


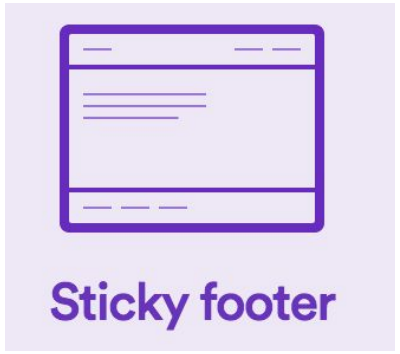
Flex layout

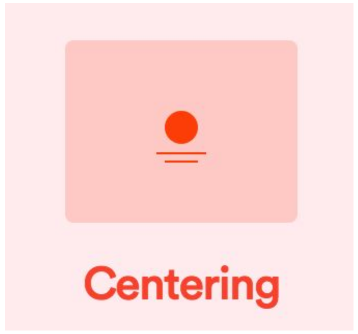
Flex layout solves all sorts of problems.

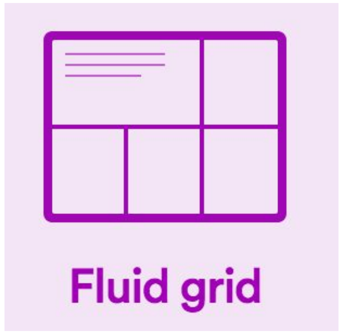
- Here are some examples of layouts that are easy to create with flex layout (and really difficult otherwise):

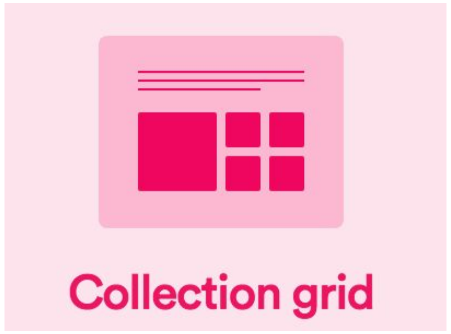


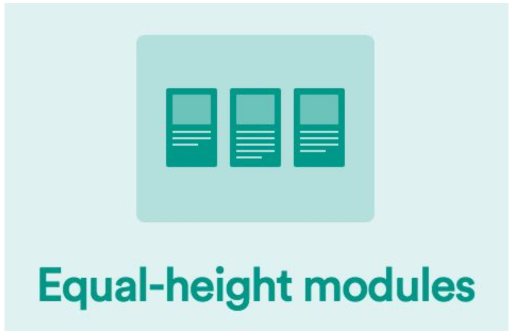










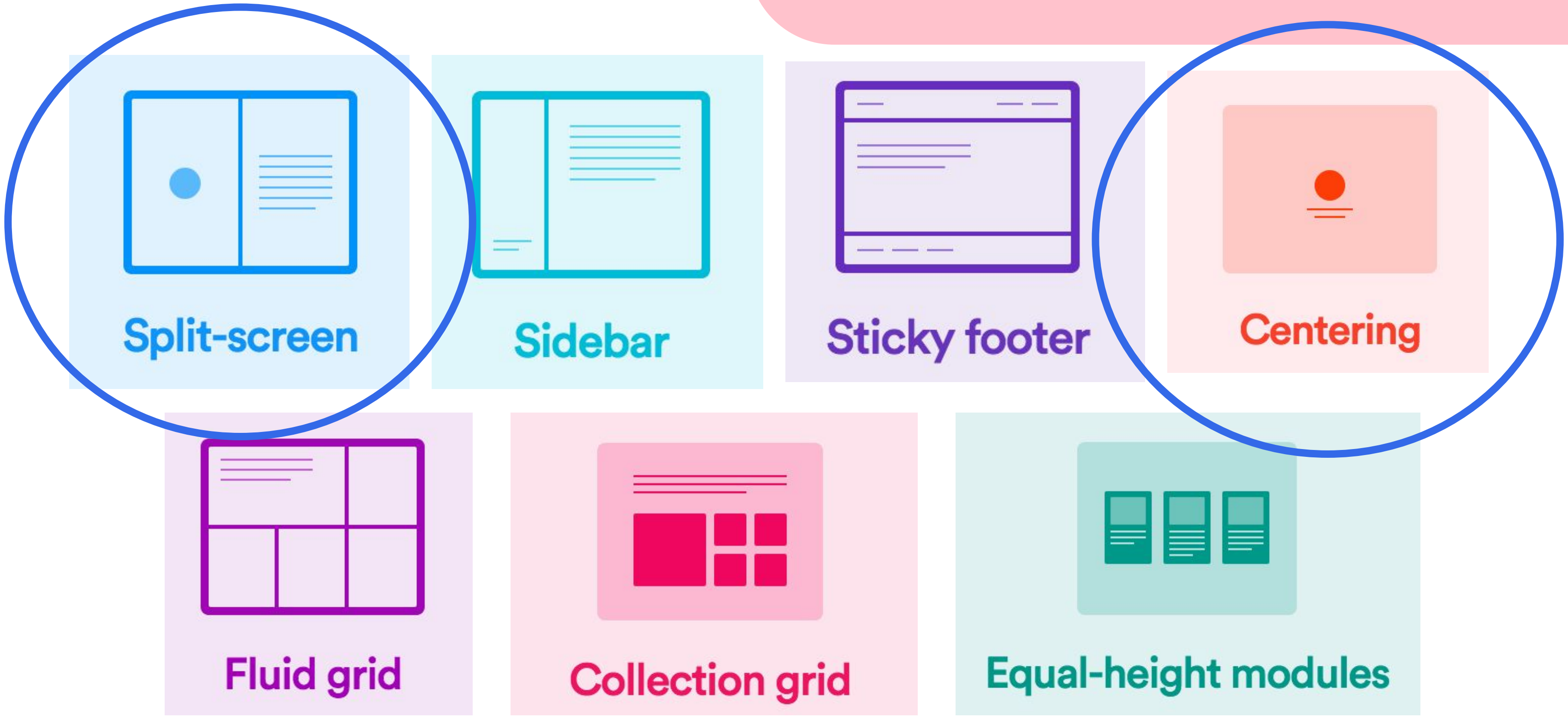


Flex layout

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!

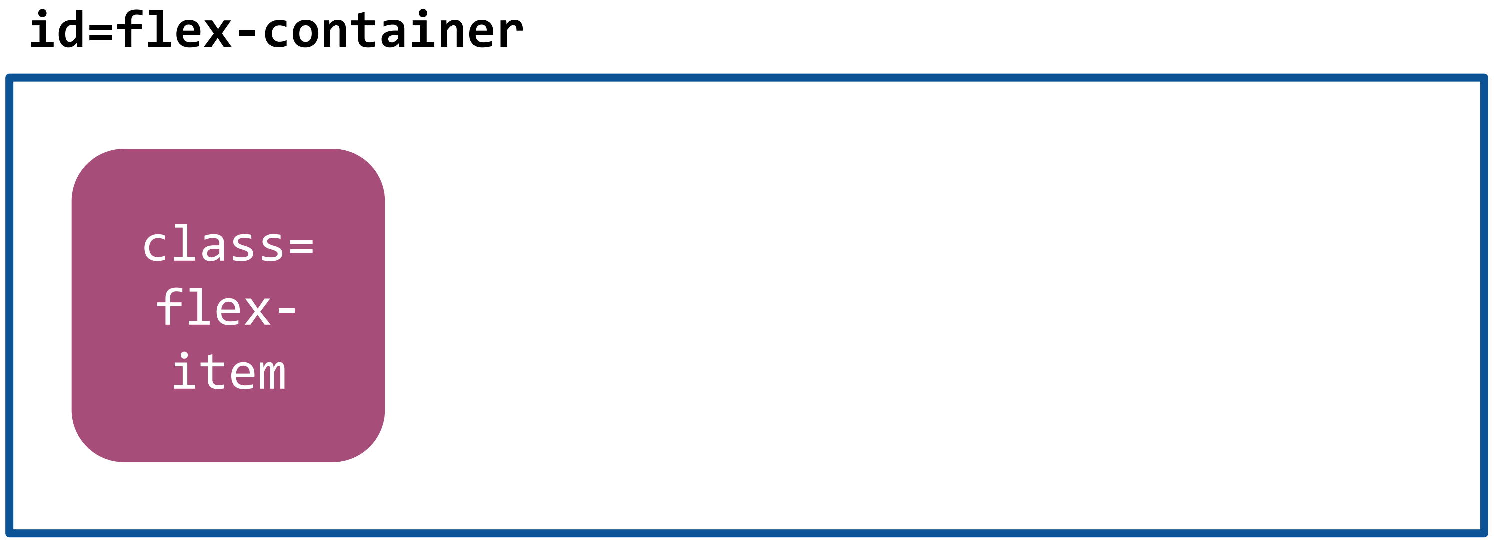


Flex basics

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.

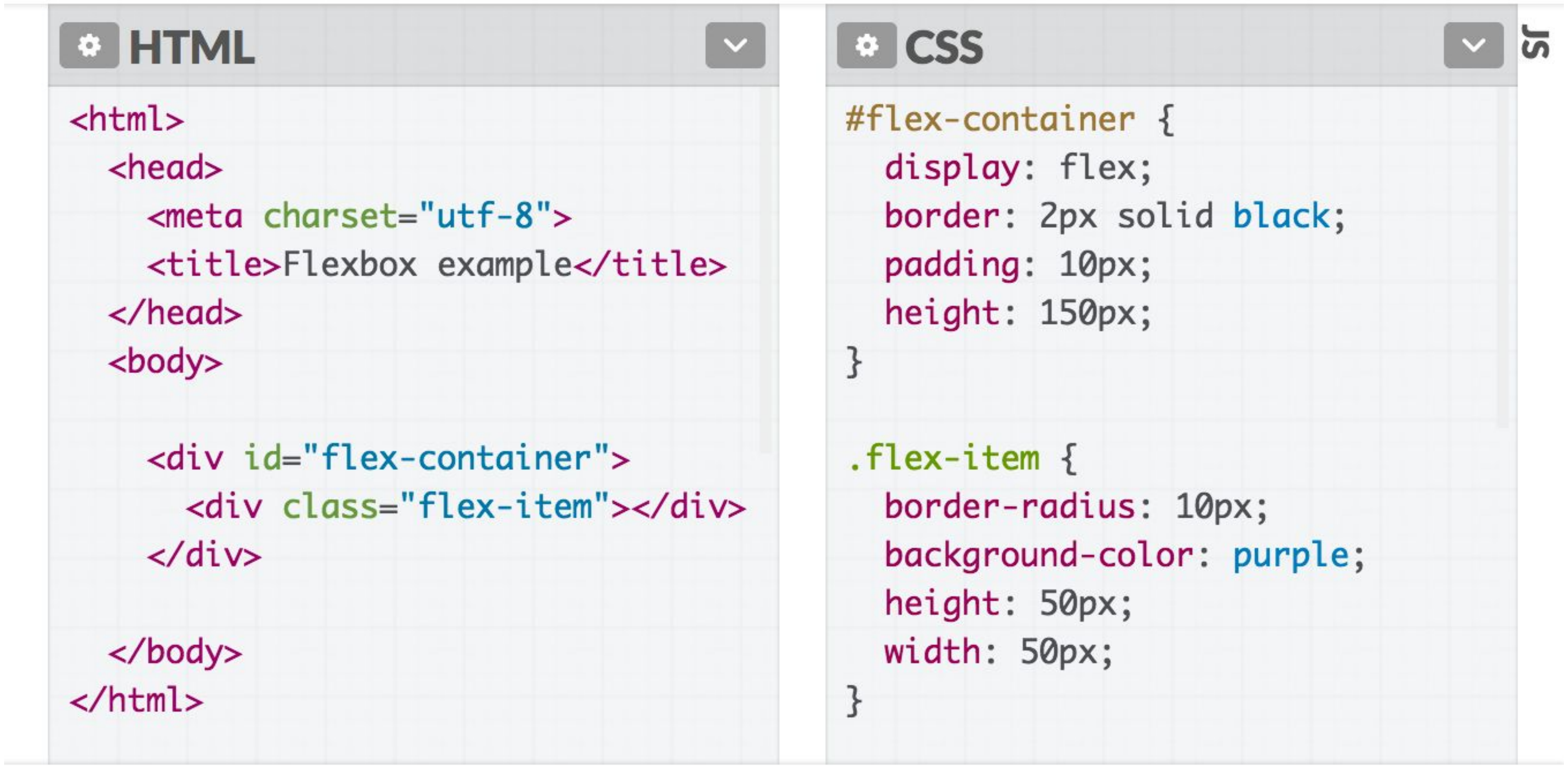


Flex basics

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

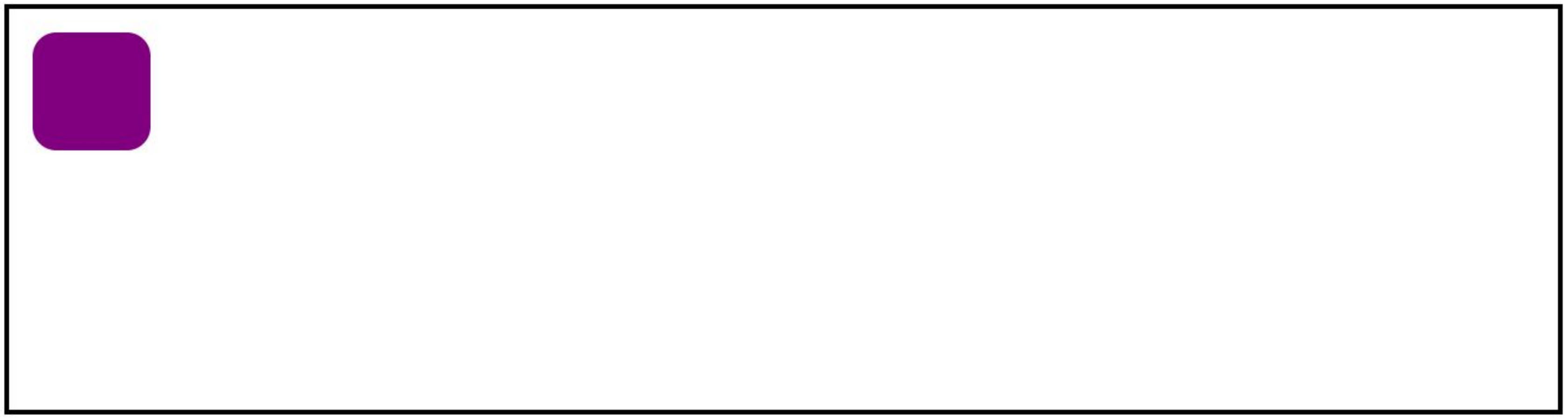
- Block container: `display: flex;` or
- Inline container: `display: inline-flex;`

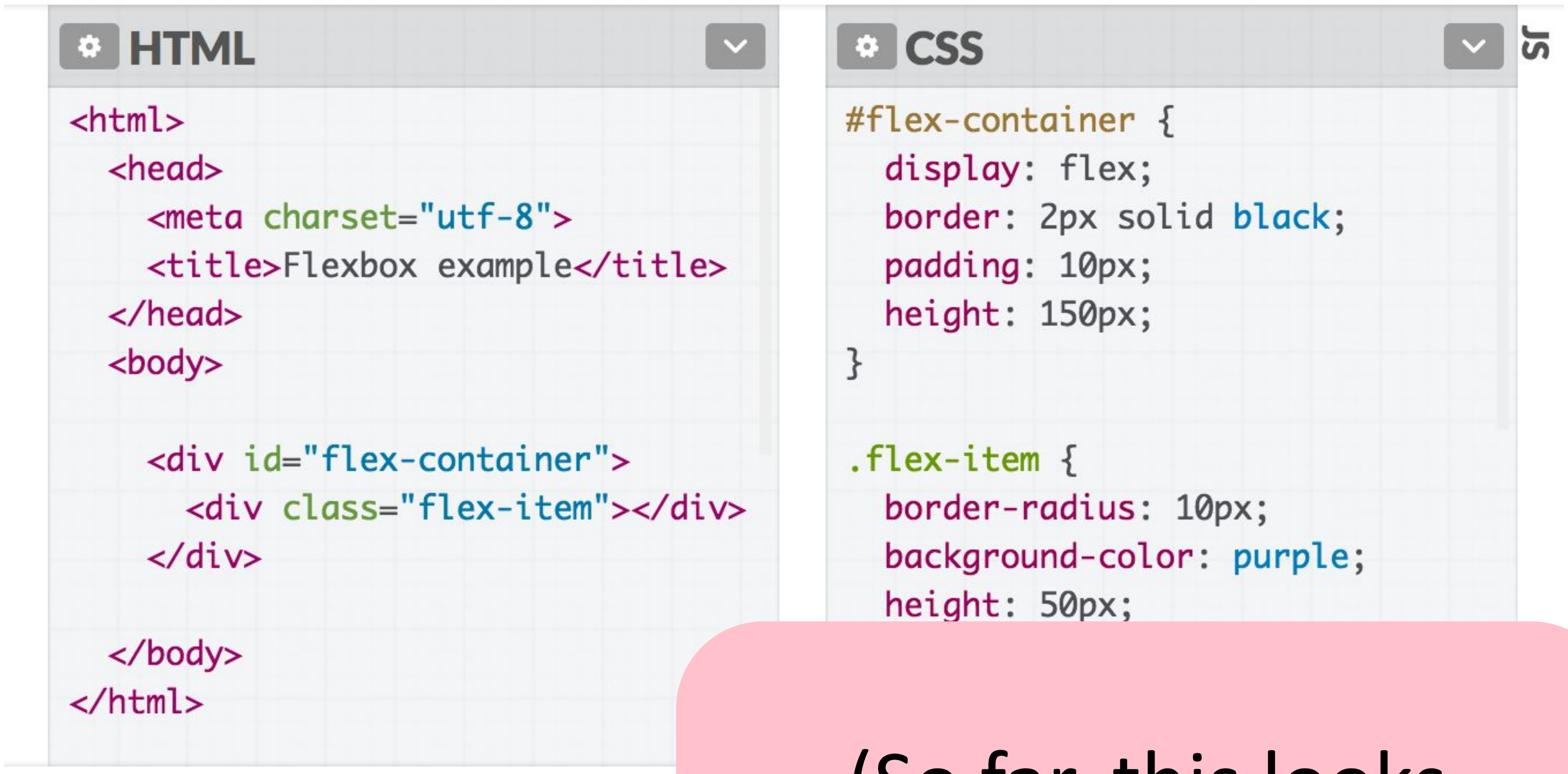
Follow along in Codepen



```html
<html>
  <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;
}
```



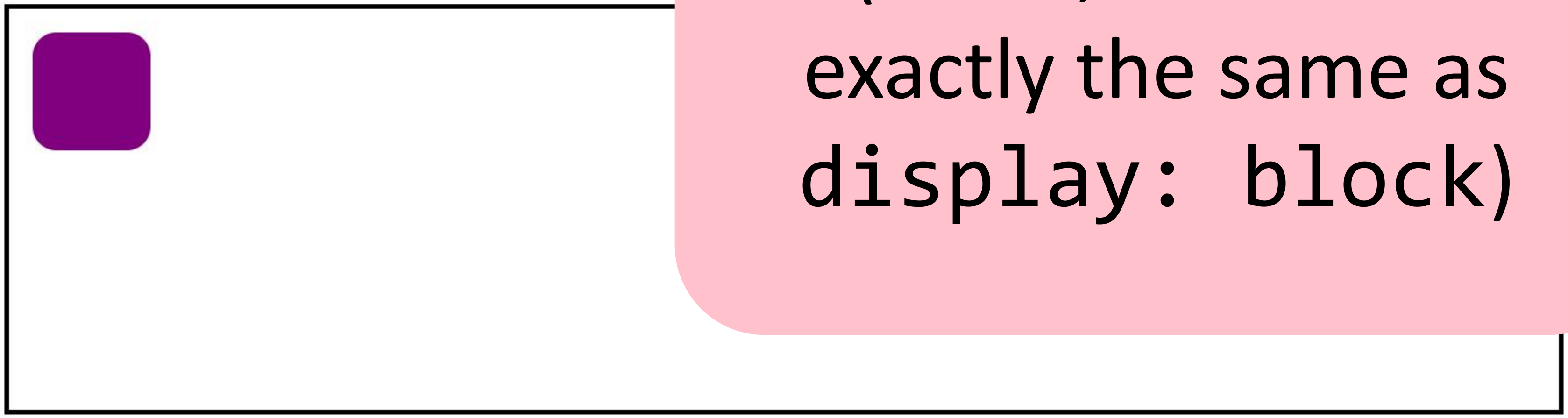


HTML

```html
<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;
```



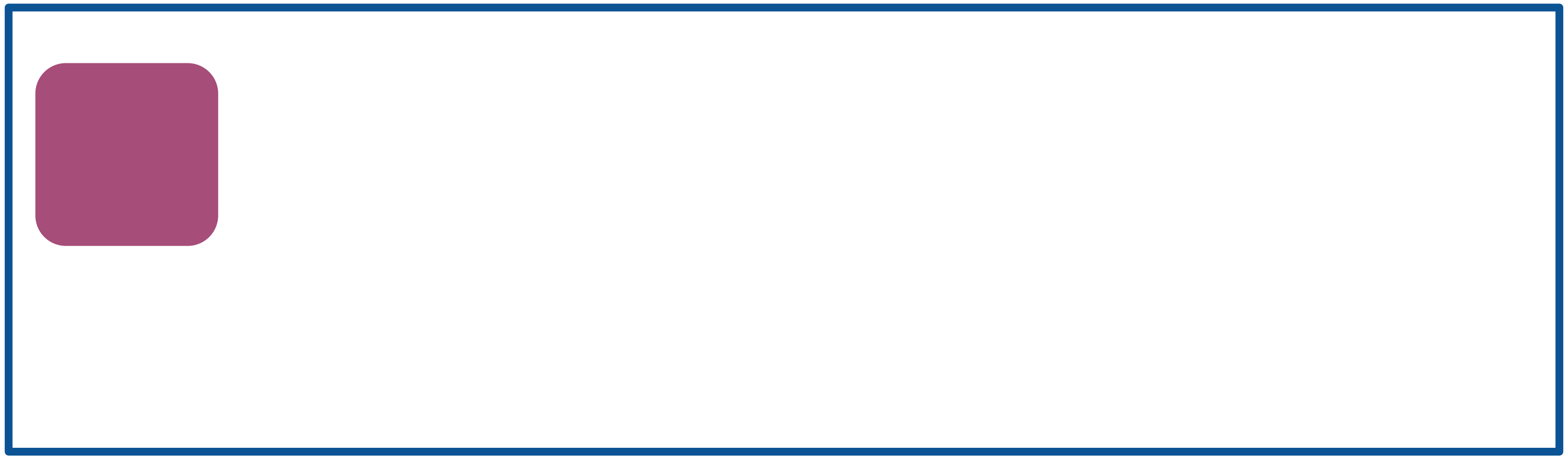
(So far, this looks exactly the same as `display: block`)

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.

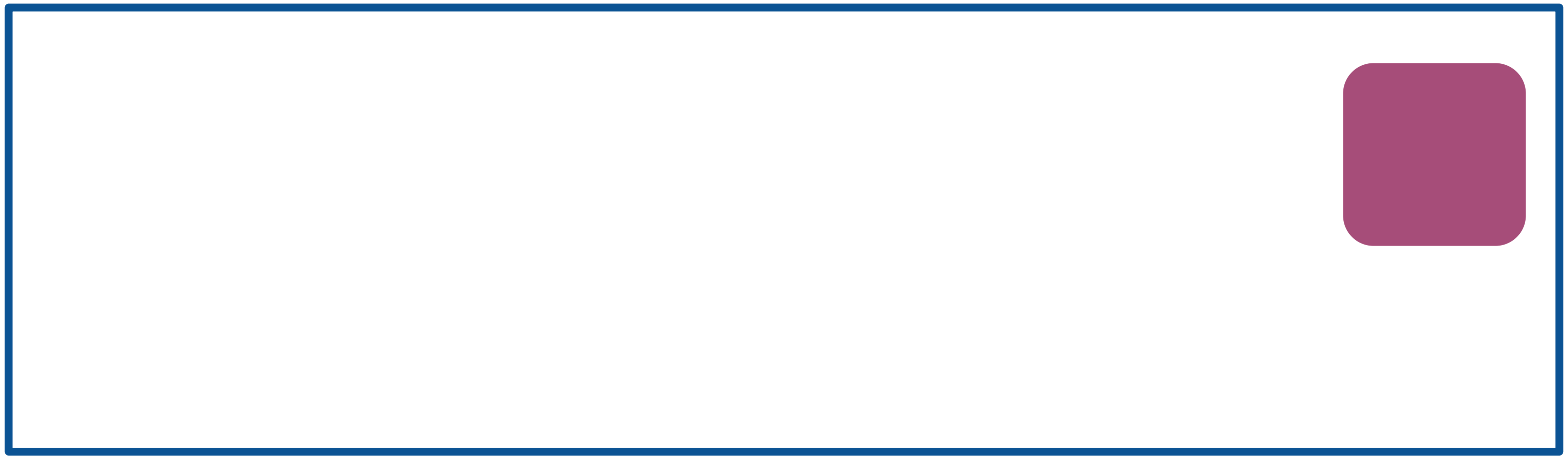


Flex basics: `justify-content`

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.

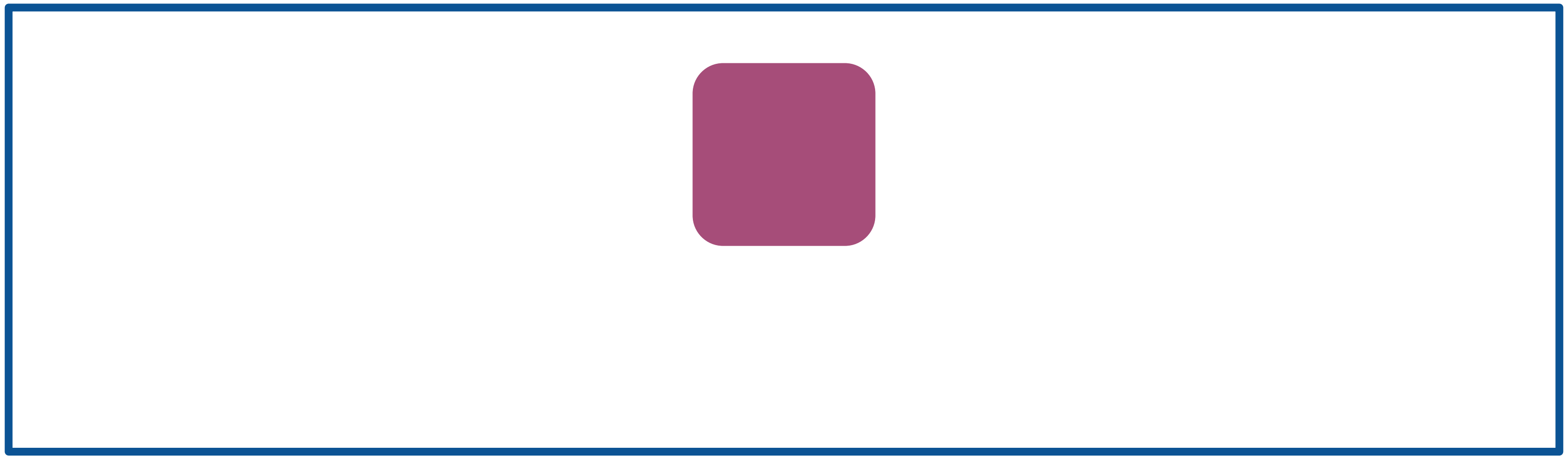


Flex basics: `justify-content`

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.

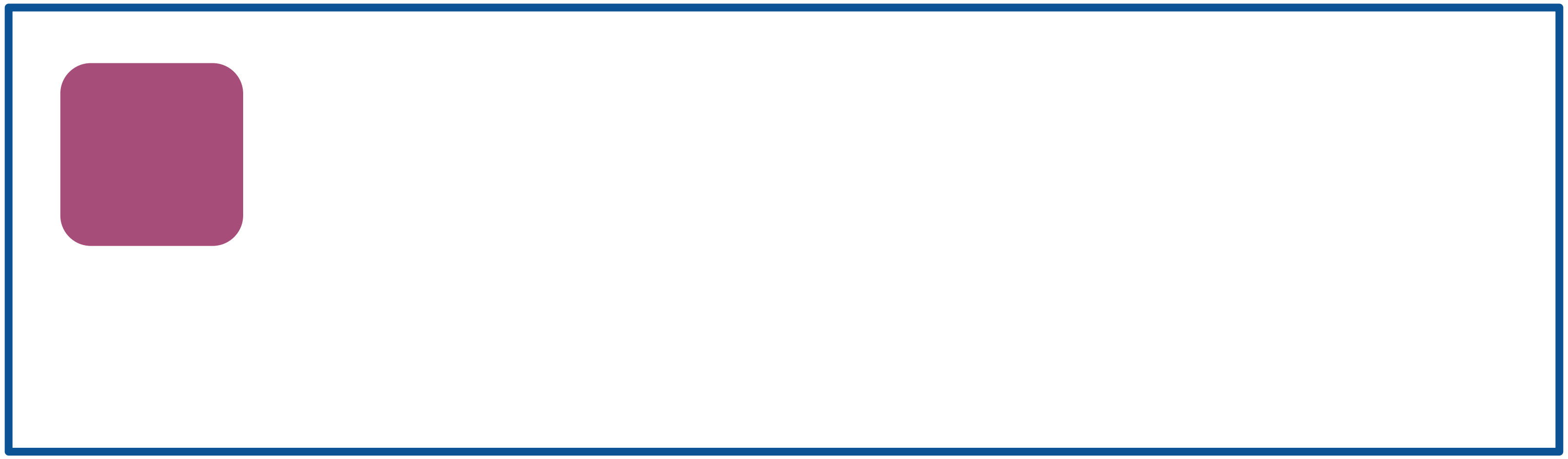


Flex basics: `align-items`

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.

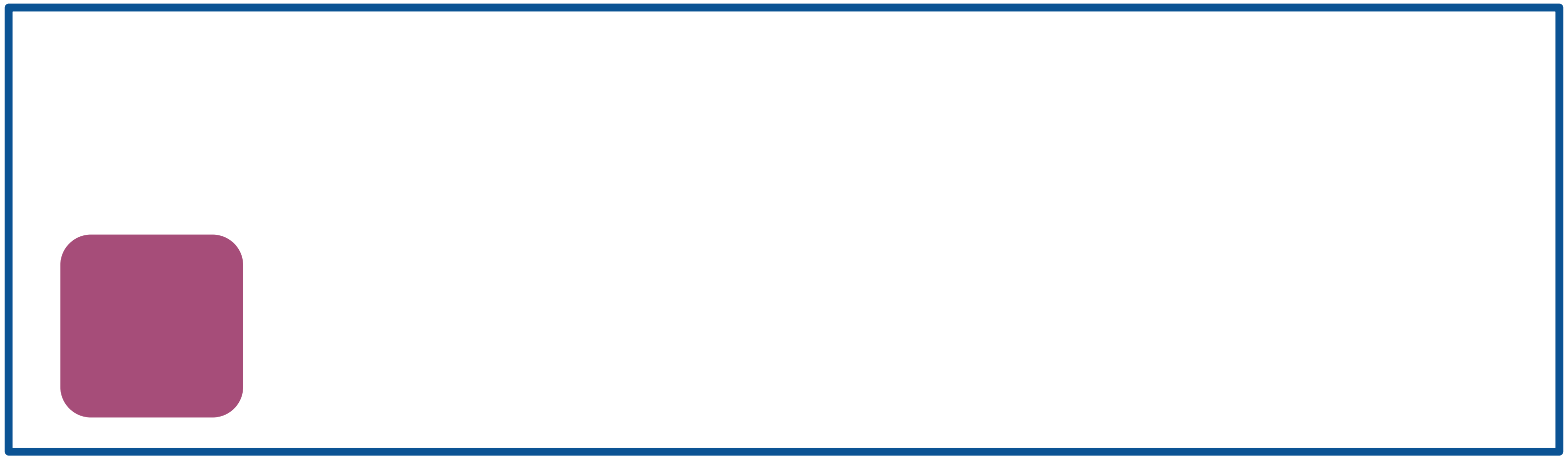


Flex basics: `align-items`

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.

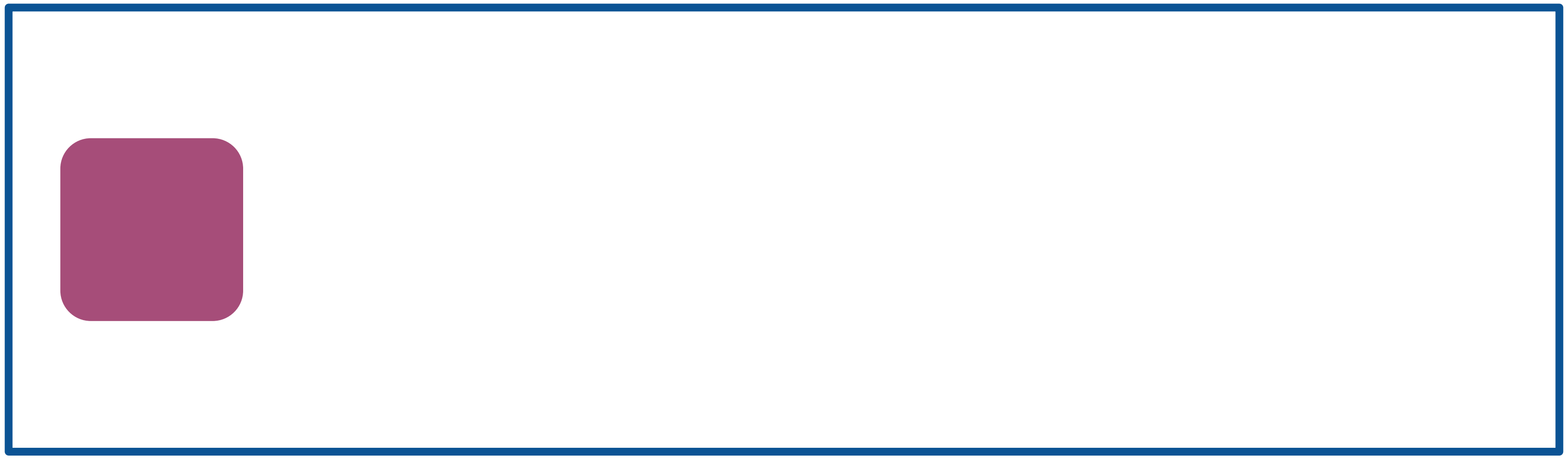


Flex basics: `align-items`

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;
}
```

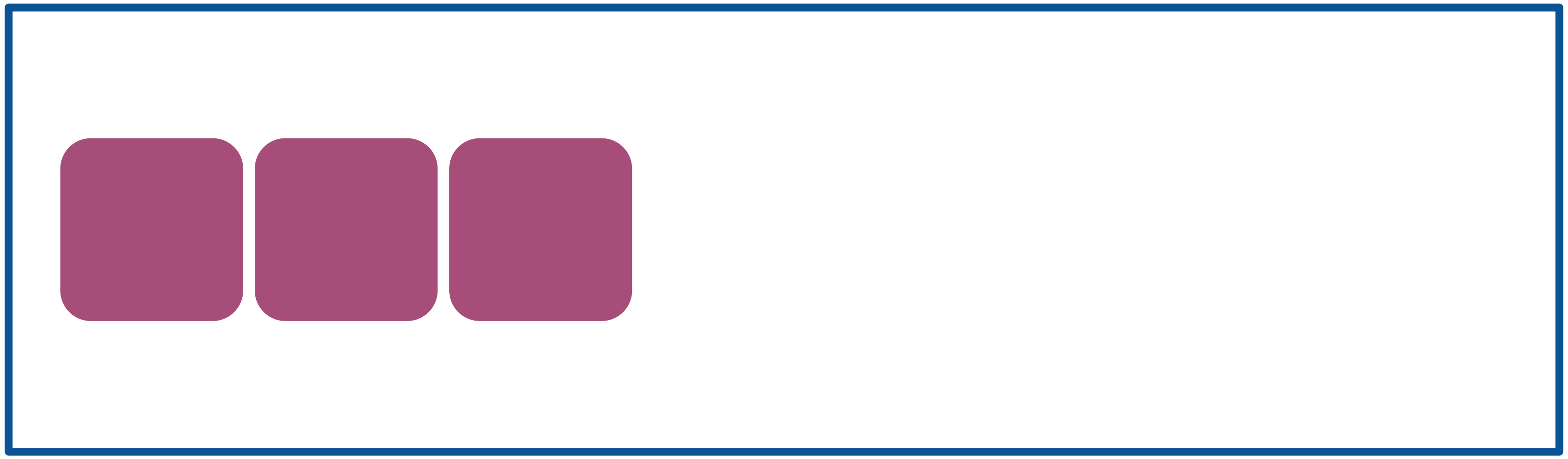
*when flex direction is `row`. We'll get to what "flex direction" means soon.



Multiple items

Same rules apply with multiple flex items:

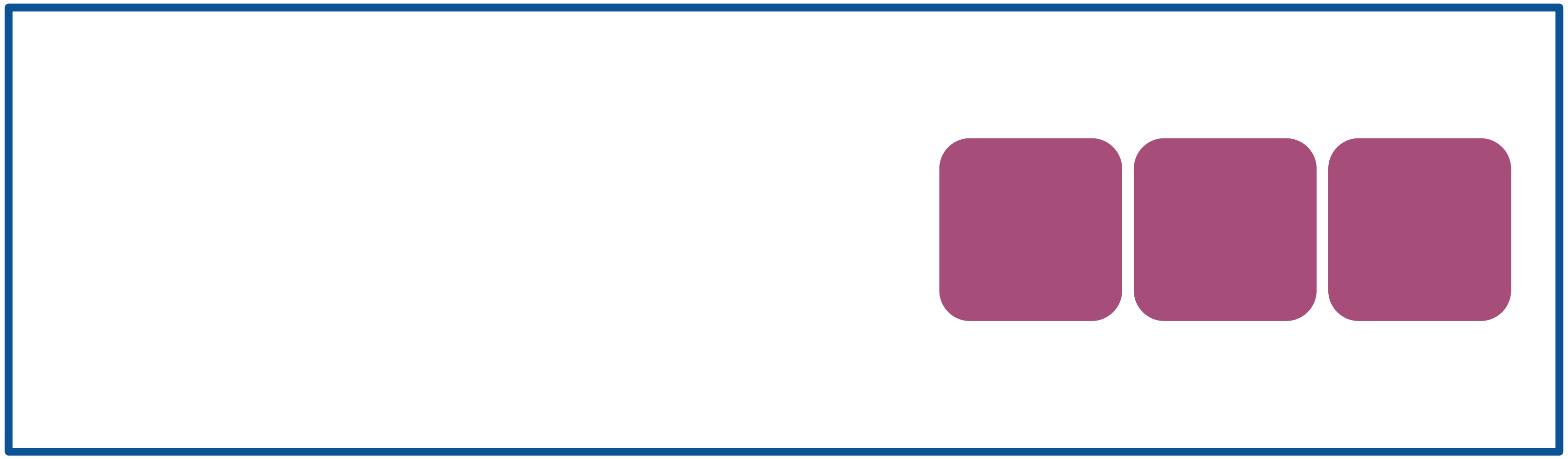
```
#flex-container {
  display: flex;
  justify-content: flex-start;
  align-items: center;
}
```



Multiple items

Same rules apply with multiple flex items:

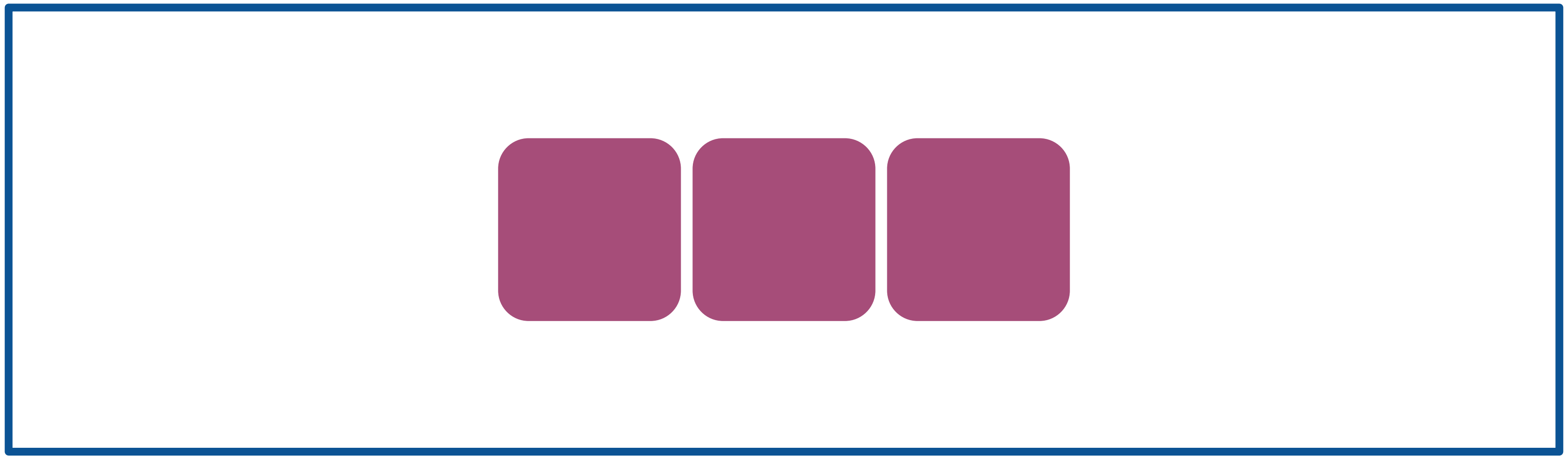
```
#flex-container {
  display: flex;
  justify-content: flex-end;
  align-items: center;
}
```



Multiple items

Same rules apply with multiple flex items:

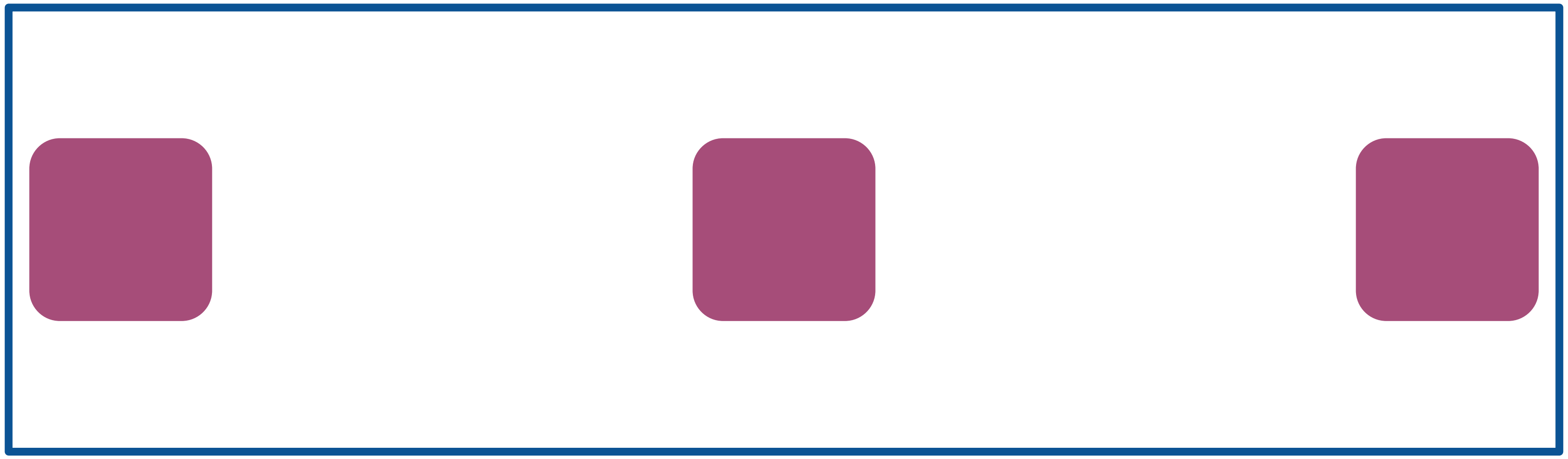
```
#flex-container {
  display: flex;
  Justify-content: center;
  align-items: center;
}
```



Multiple items

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

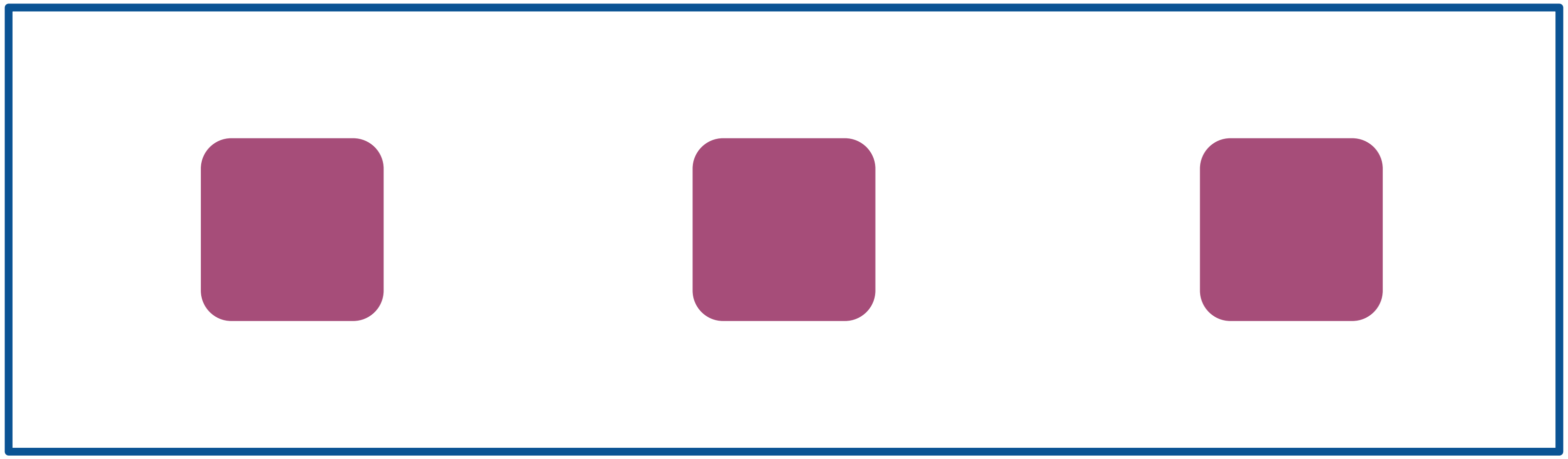
```
#flex-container {
  display: flex;
  Justify-content: space-between;
  align-items: center;
}
```



Multiple items

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

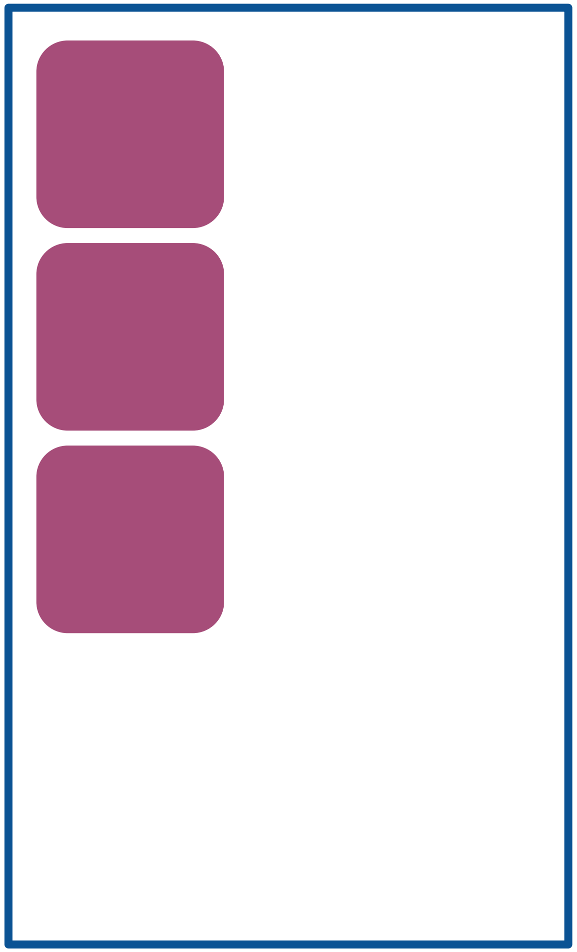
```
#flex-container {
  display: flex;
  Justify-content: space-around;
  align-items: center;
}
```



flex-direction

And you can also lay out columns instead of rows:

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

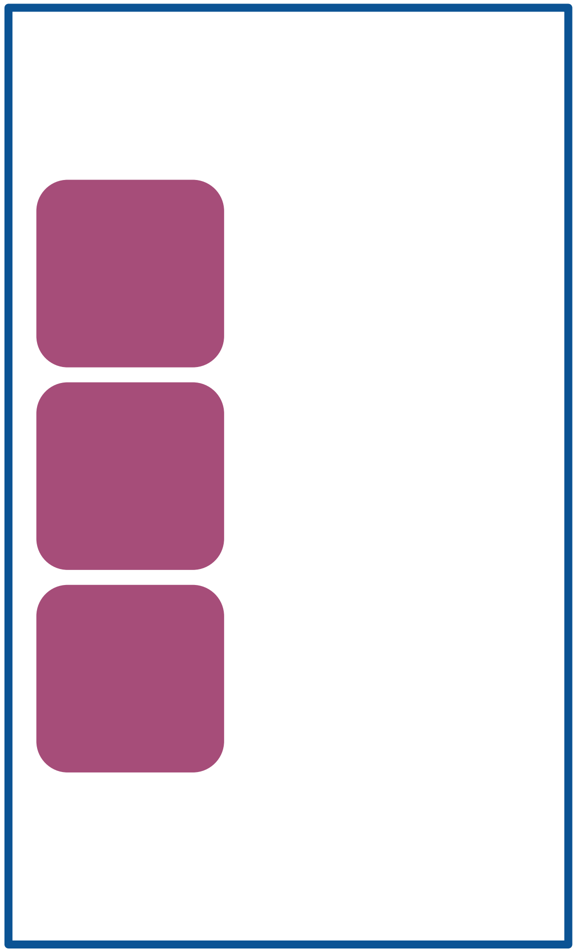


flex-direction

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

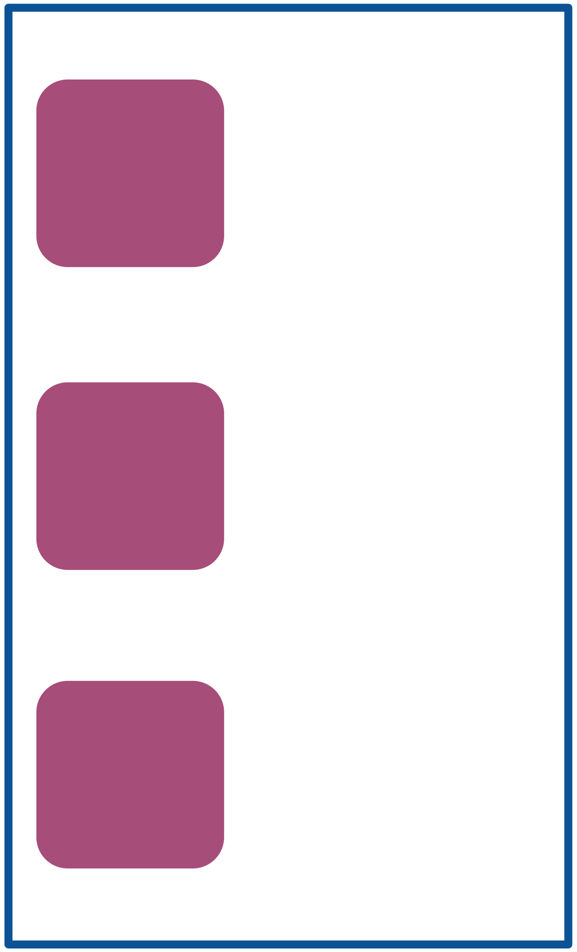


`flex-direction`

And you can also lay out columns instead of rows:

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

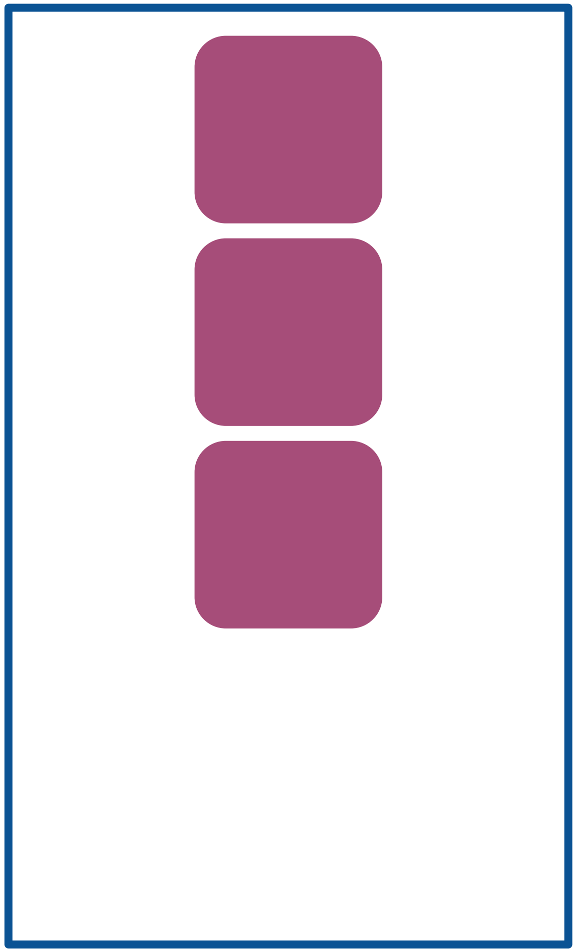
Now **`justify-content`** controls where the column is vertically in the box



flex-direction

And you can also lay out columns instead of rows:

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

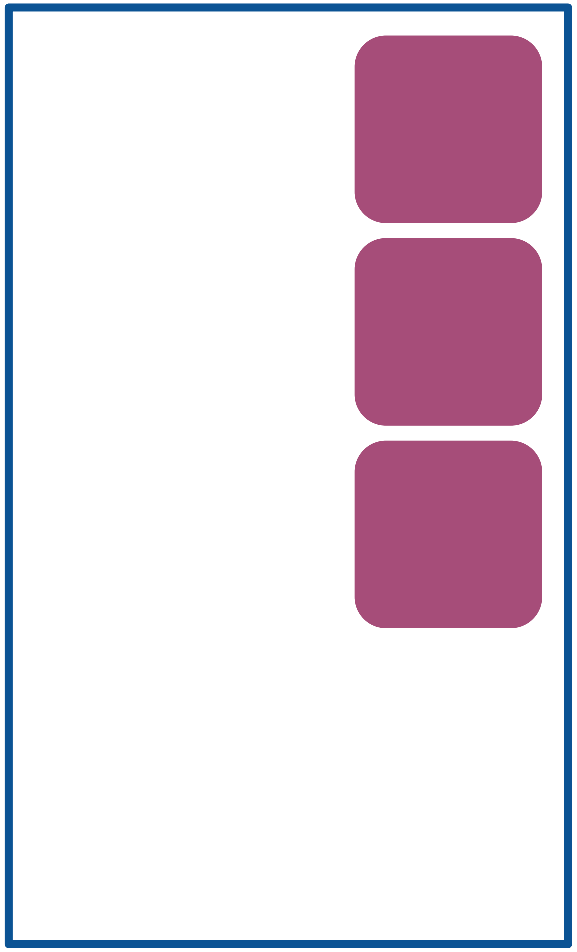


Now **align-items** controls where the column is horizontally in the box

flex-direction

And you can also lay out columns instead of rows:

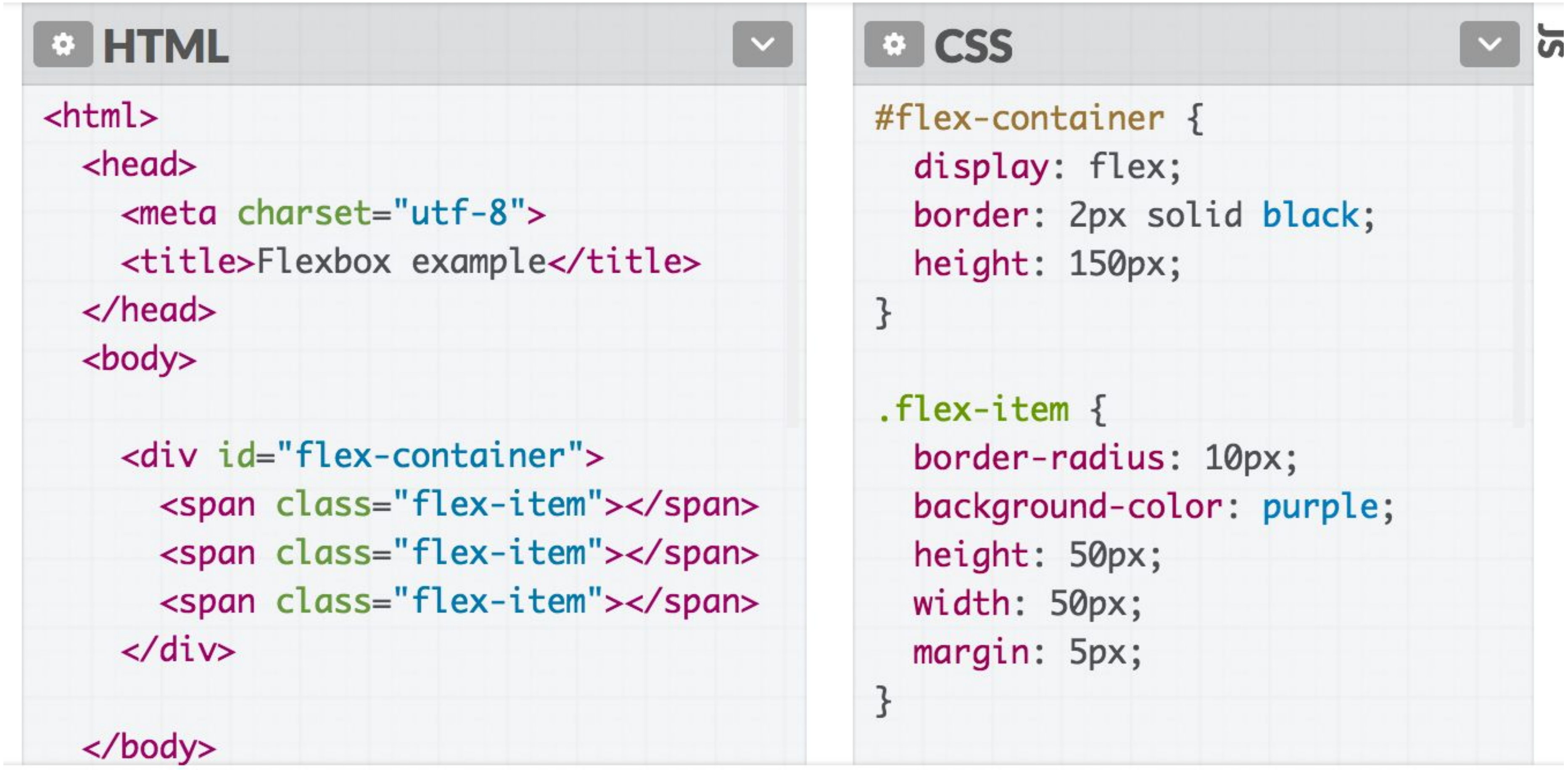
```
#flex-container {
  display: flex;
  flex-direction: column;
  align-items: flex-end;
}
```



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

Before we move on…

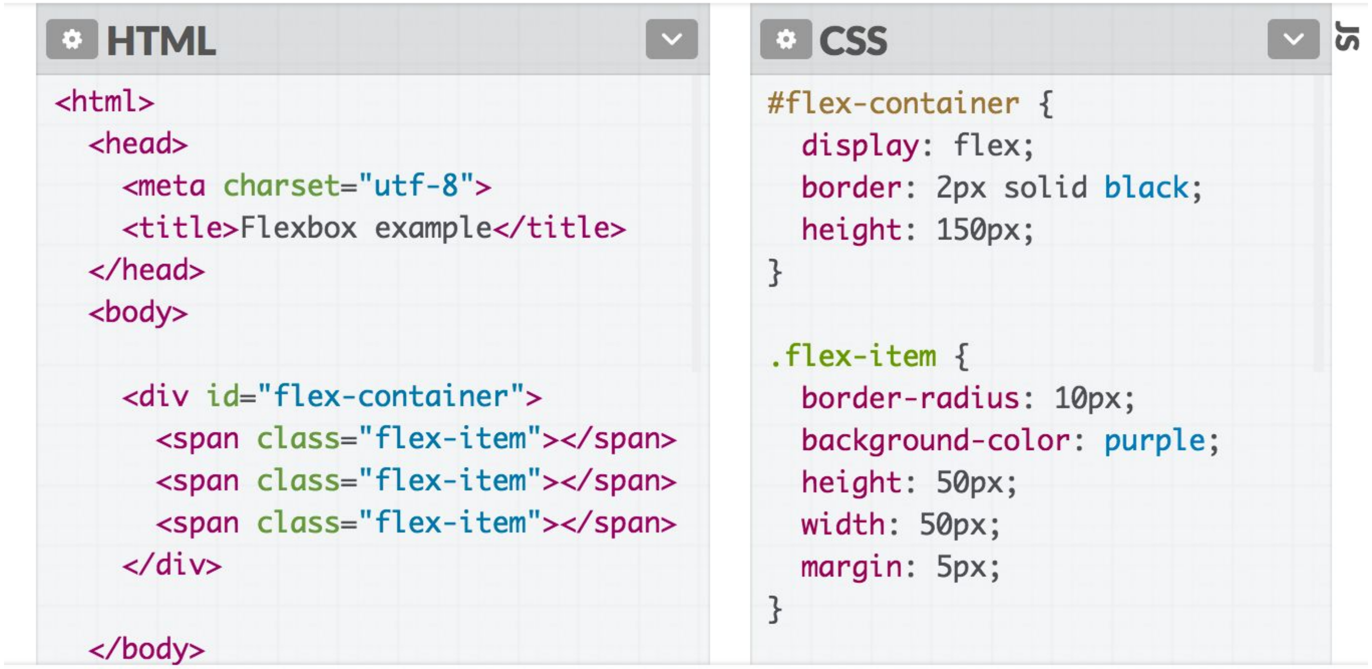
What happens if the flex item is an inline element?



```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;
}
```

???

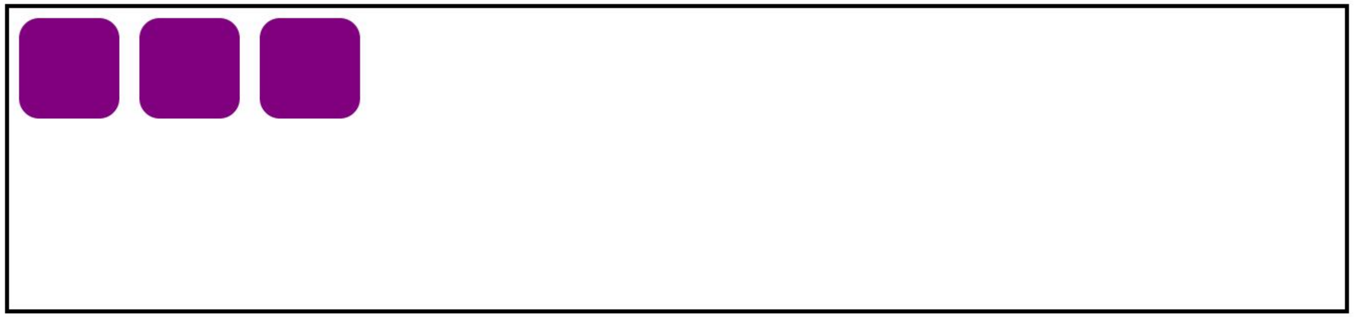


HTML

```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>
```

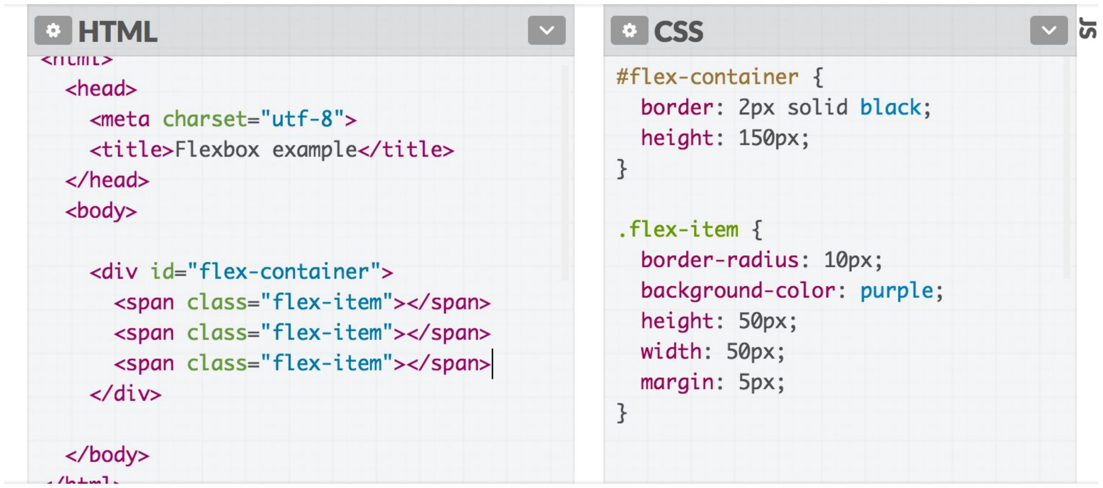
CSS

```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;
}
```



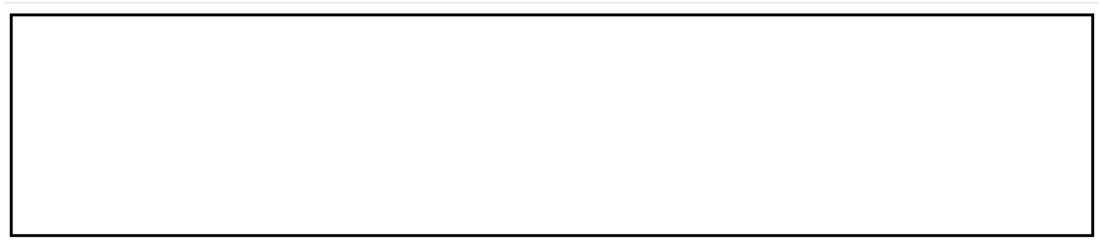
Recall: block layouts

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



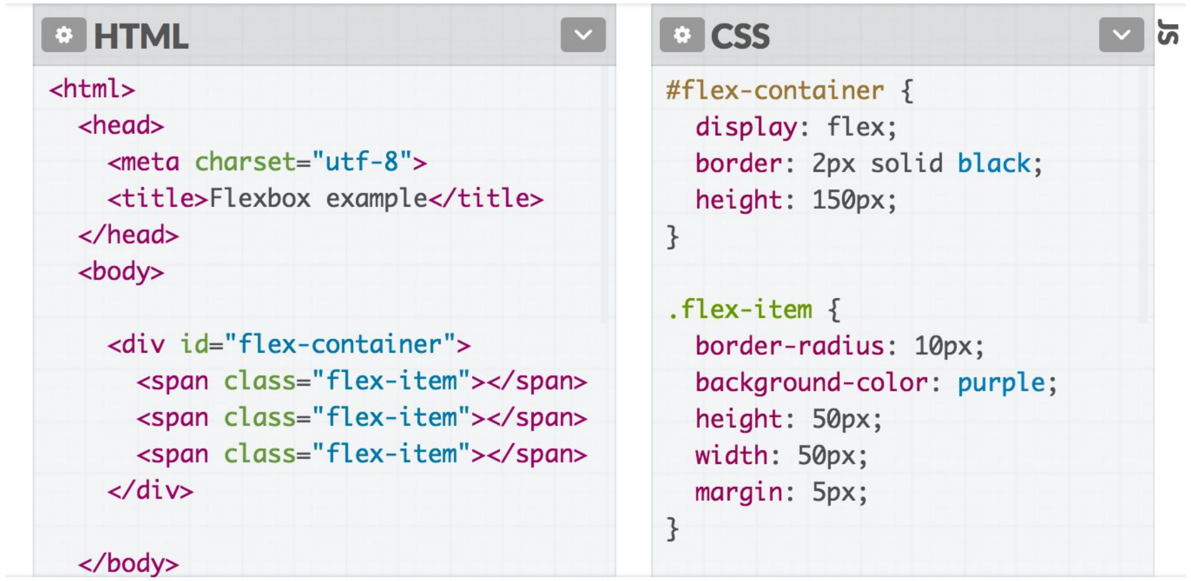
```html
<html>
  <head>
    <meta charset="utf-8">
    <title>Flexbox example</title>
  </head>
  <body>

    <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;
}
```



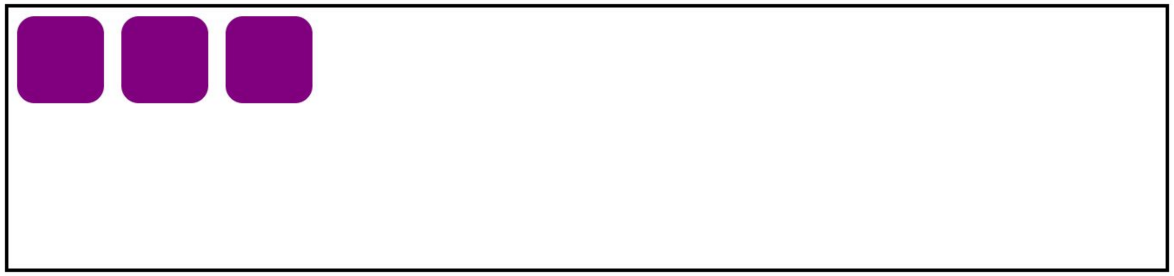
Then the `span` `flex-items` would not show up because `span` elements are inline, which don't have a height and width

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?

Flex: A different rendering mode

- When you set a container to `display: flex`, the direct children in that container are **flex items** and follow a new set of rules.
- **Flex items are not block or inline;** they have different rules for their height, width, and layout.
  - The *contents* of a flex item follow the usual block/inline rules, relative to the flex item's boundary.
- The **height** and **width** of flex items are… complicated.

Flex item sizing

Flex basis

Flex items have an initial width*, which, by default is either:

- The content width, or
- The explicitly set **`width`** property of the element, or
- The explicitly set **`flex-basis`** property of the element

This initial width* of the flex item is called the **flex basis**.

*width in the case of rows; height in the case of columns

Flex basis

Flex items have an initial width*, which, by default is either:

- The content width, or
- The explicitly set **`width`** property of the element, or
- The explicitly set **`flex-basis`** property of the element

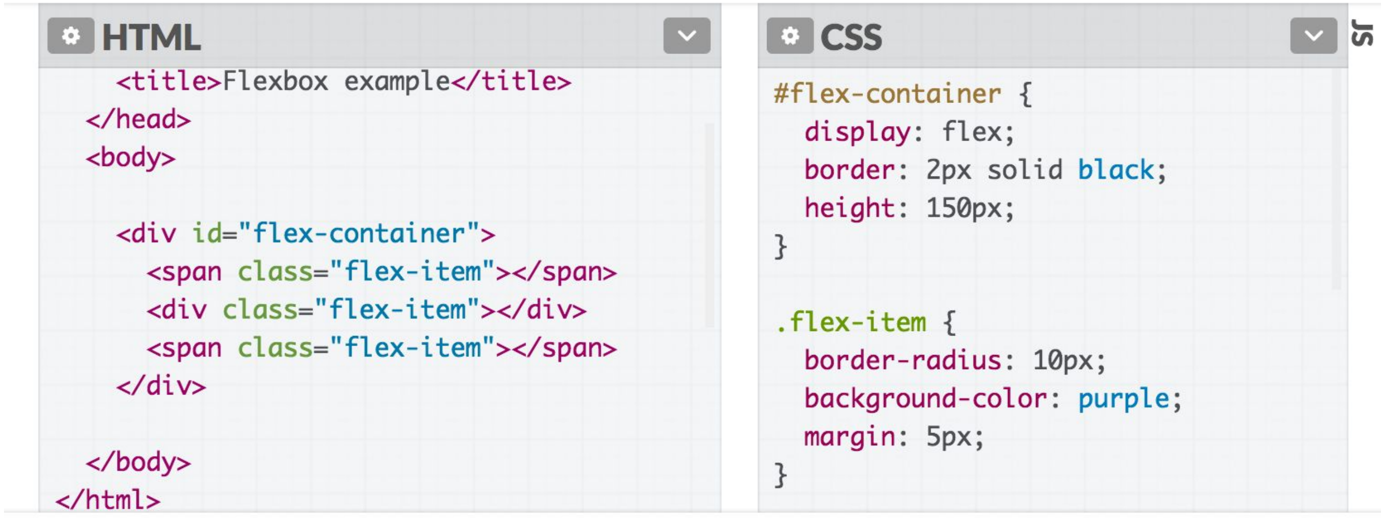
This initial width* of the flex item is called the **flex basis**.

The explicit width* of a flex item is respected *for all flex items*, regardless of whether the flex item is inline, block, or inline-block.

*width in the case of rows; height in the case of columns

Flex basis

If we unset the `height` and `width`, our flex items disappears, because the **flex basis** is now the content size, which is empty:



```html
    <title>Flexbox example</title>
  </head>
  <body>

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

  </body>
</html>
```

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

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

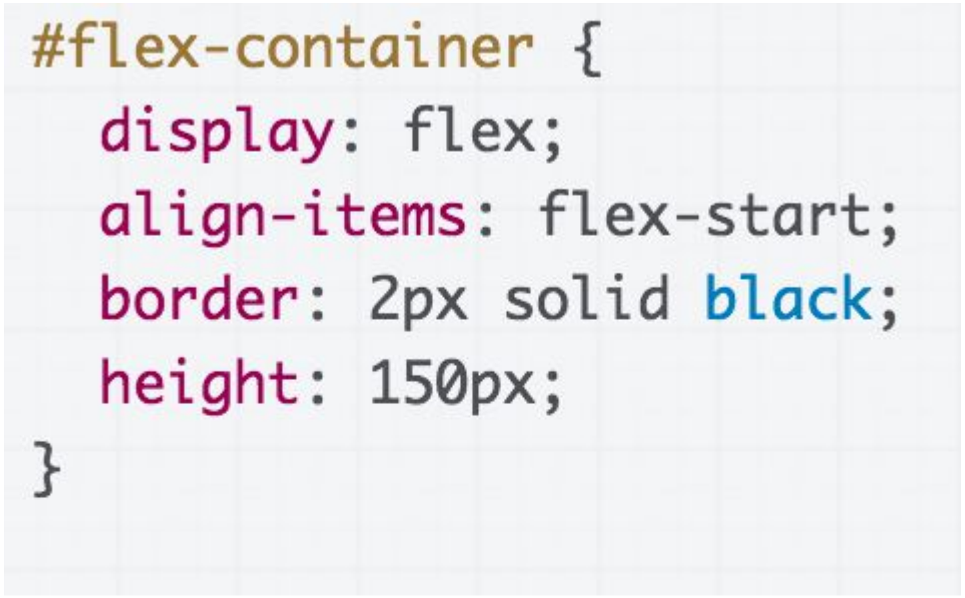
`flex-shrink`

The width* of the flex item can automatically shrink **smaller than the flex basis** via the `flex-shrink` property:

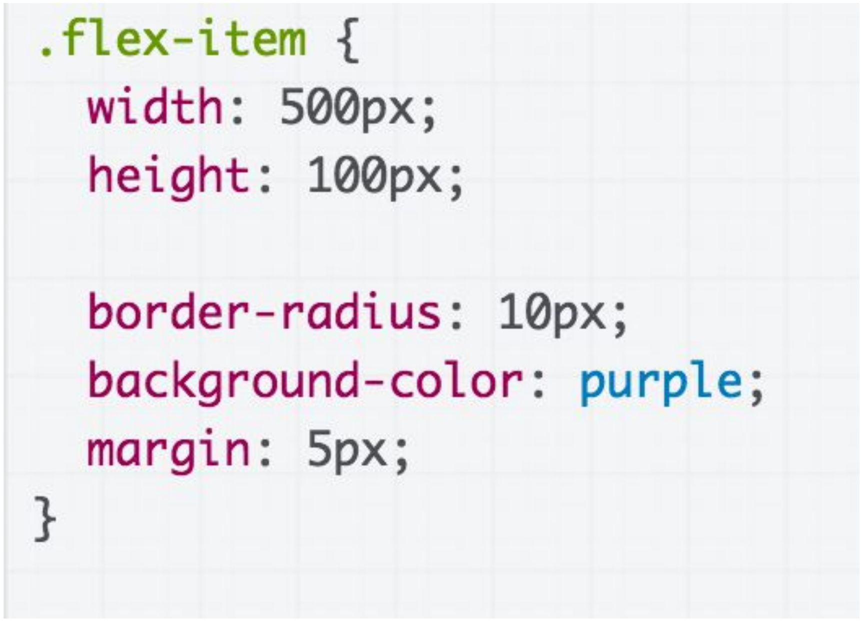
`flex-shrink`:
- If set to 1, the flex item shrinks itself as small as it can in the space available.
- If set to 0, the flex item does not shrink.

Flex items have `flex-shrink:` 1 by default.

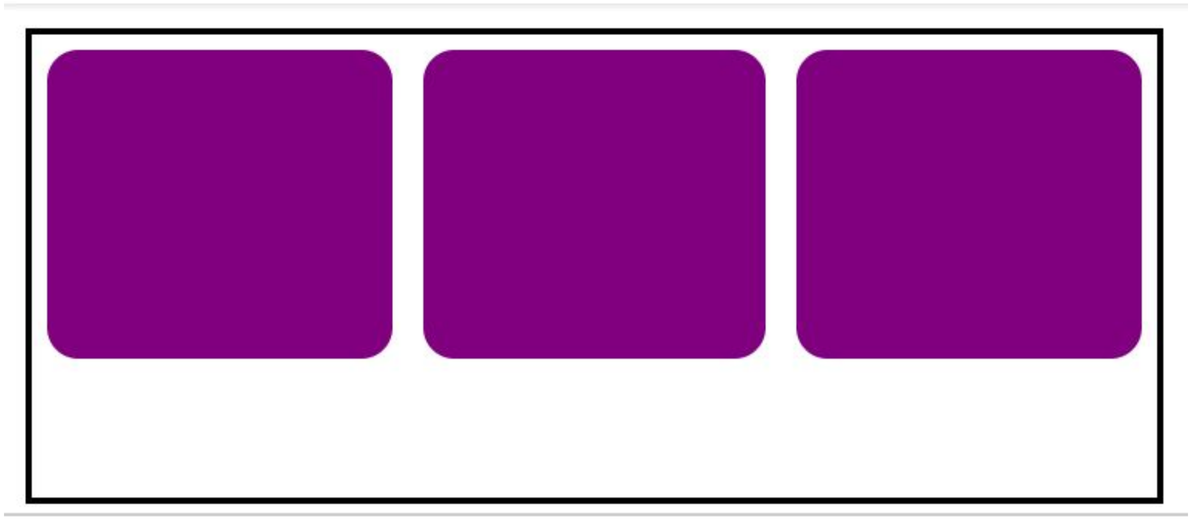
*width in the case of rows; height in the case of columns



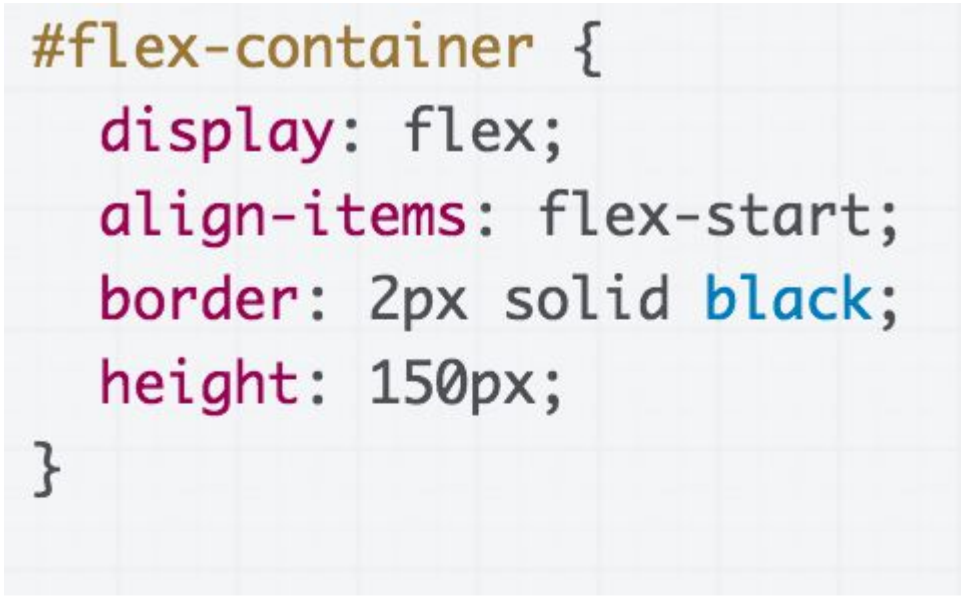
```css
#flex-container {
  display: flex;
  align-items: flex-start;
  border: 2px solid black;
  height: 150px;
}
```



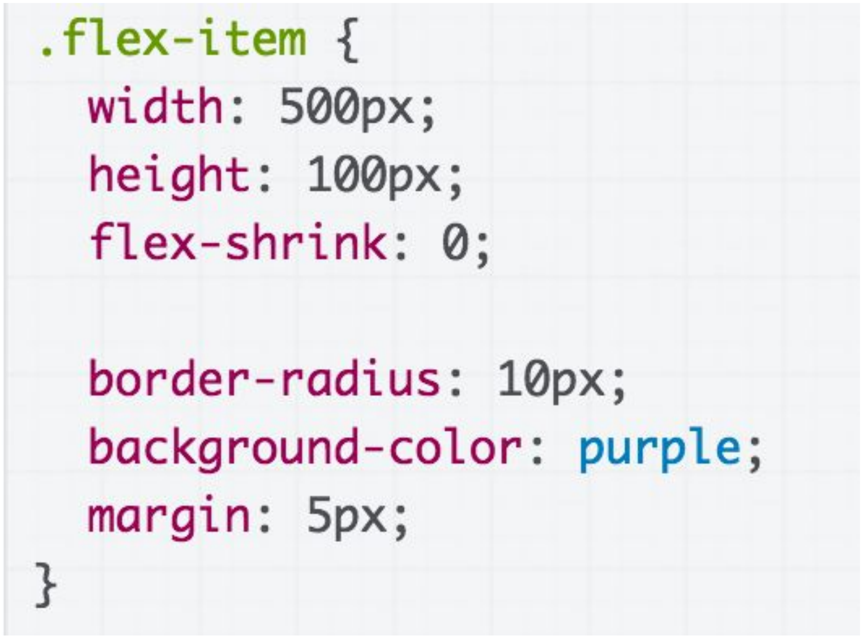
```css
.flex-item {
  width: 500px;
  height: 100px;

  border-radius: 10px;
  background-color: purple;
  margin: 5px;
}
```



The flex items' widths all shrink to fit within the container.

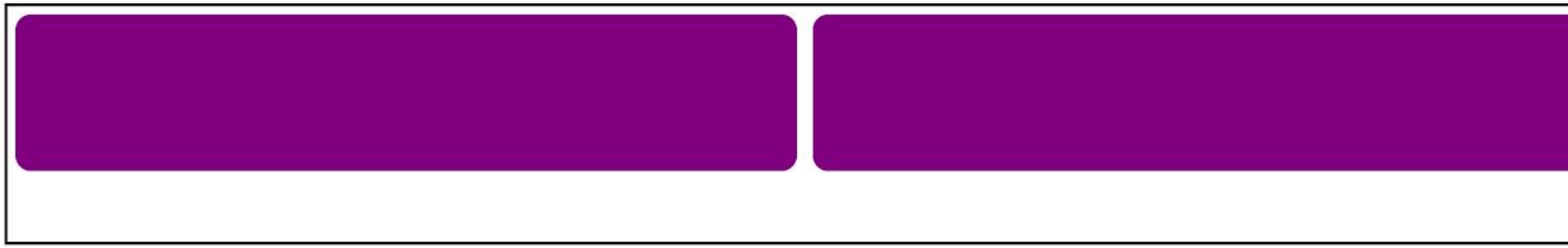


```
#flex-container {
  display: flex;
  align-items: flex-start;
  border: 2px solid black;
  height: 150px;
}
```



```
.flex-item {
  width: 500px;
  height: 100px;
  flex-shrink: 0;

  border-radius: 10px;
  background-color: purple;
  margin: 5px;
}
```

Setting `flex-shrink: 0;` undoes the shrinking behavior, and the flex items do not shrink in any circumstance:



`flex-grow`

The width* of the flex item can automatically **grow larger than the flex basis** via the `flex-grow` property:

`flex-grow`:

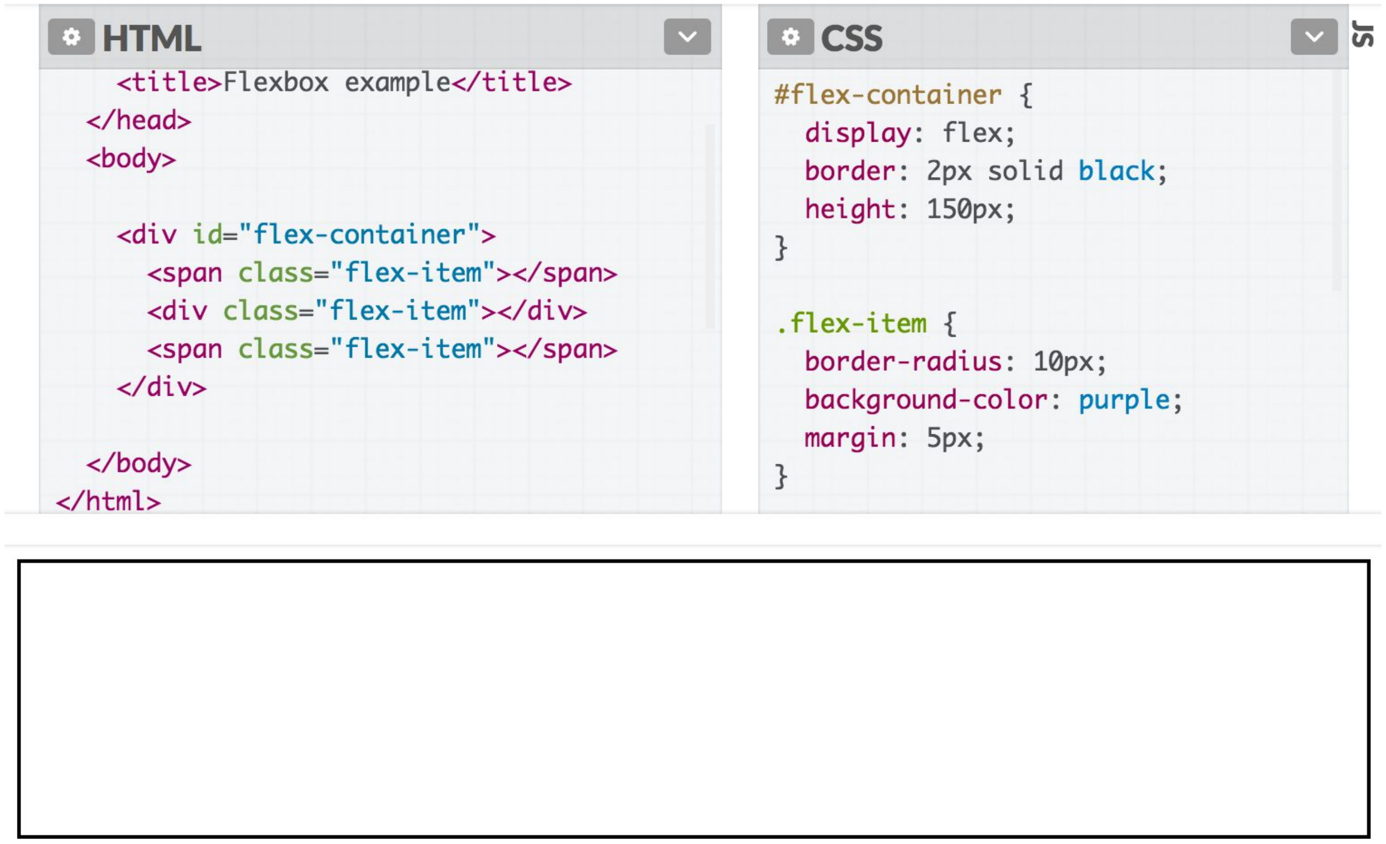
- If set to 1, the flex item grows itself as large as it can in the space remaining.
- If set to 0, the flex-item does not grow.

Flex items have `flex-grow: 0` by default.

*width in the case of rows; height in the case of columns

`flex-grow` example

Let's unset the height and width of our flex items again:

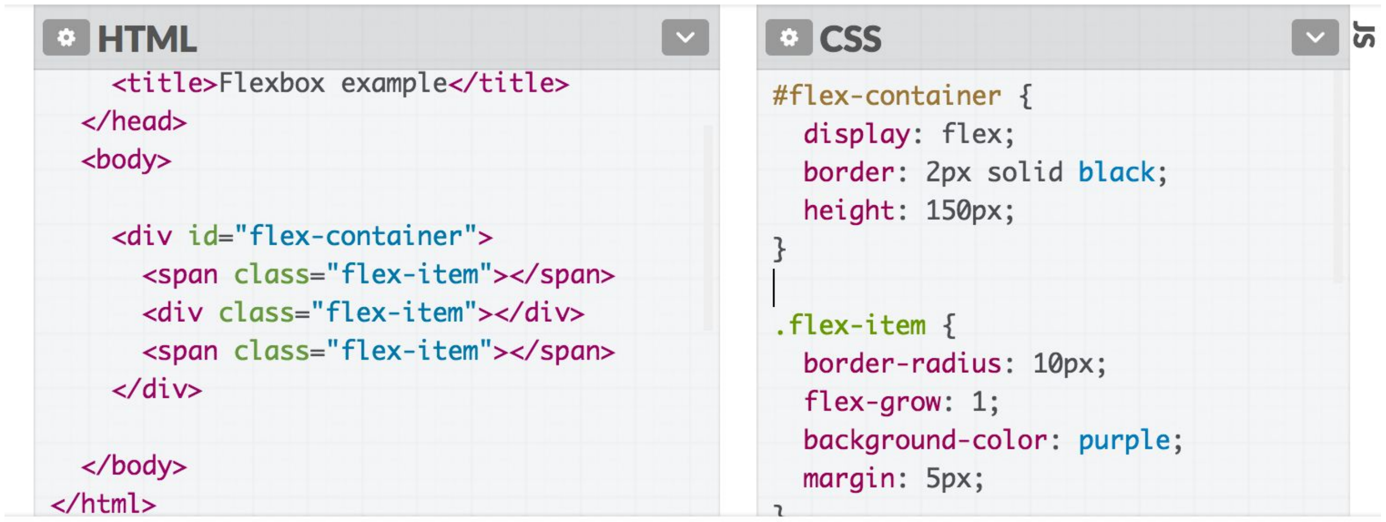


```html
    <title>Flexbox example</title>
  </head>
  <body>

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

  </body>
</html>
```

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

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

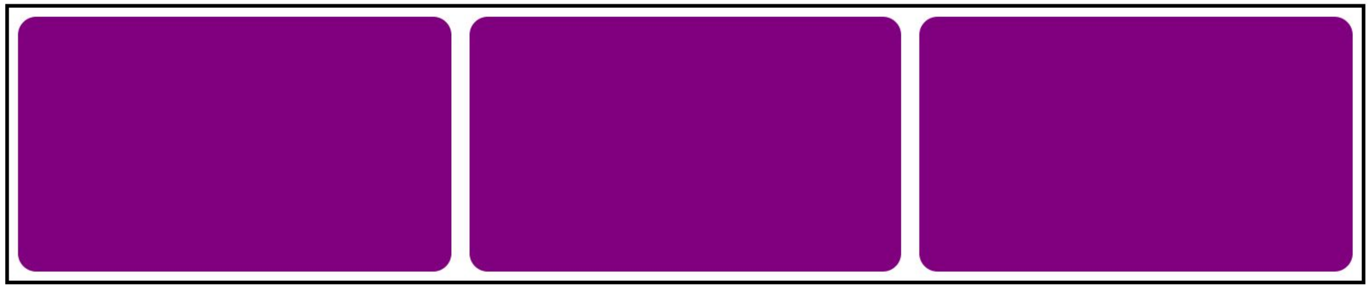
`flex-grow` example

If we set `flex-grow: 1`, the flex items fill the empty space:



```html
    <title>Flexbox example</title>
  </head>
  <body>

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

  </body>
</html>
```

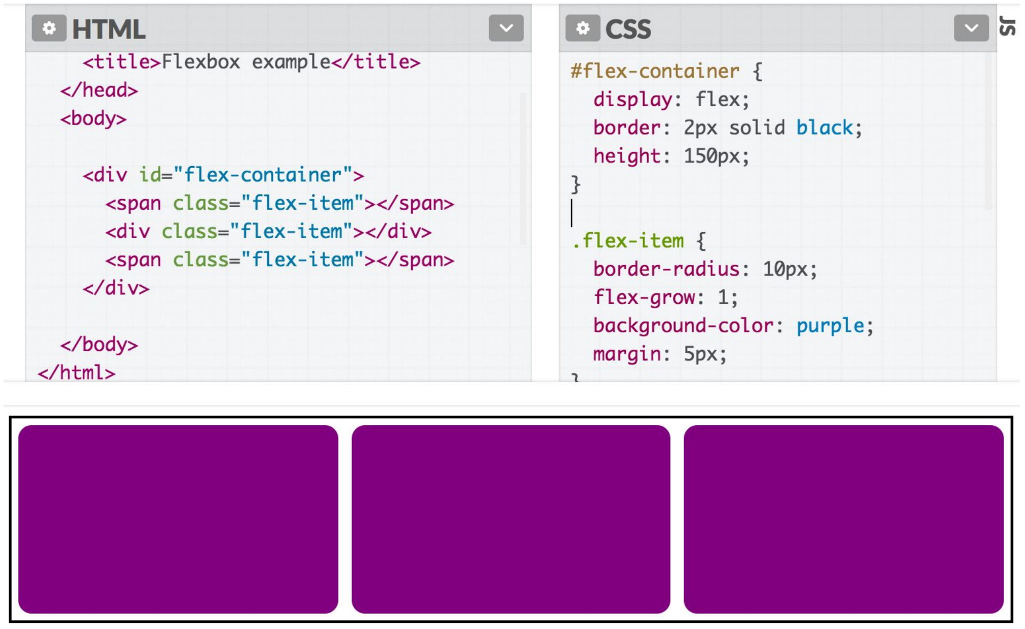
```css
#flex-container {
  display: flex;
  border: 2px solid black;
  height: 150px;
}

.flex-item {
  border-radius: 10px;
  flex-grow: 1;
  background-color: purple;
  margin: 5px;
```



Flex item height**?!

Note that `flex-grow` only controls width*.

So why does the height** of the flex items seem to "grow" as well?



```html
    <title>Flexbox example</title>
  </head>
  <body>

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

  </body>
</html>
```

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

.flex-item {
  border-radius: 10px;
  flex-grow: 1;
  background-color: purple;
  margin: 5px;
```

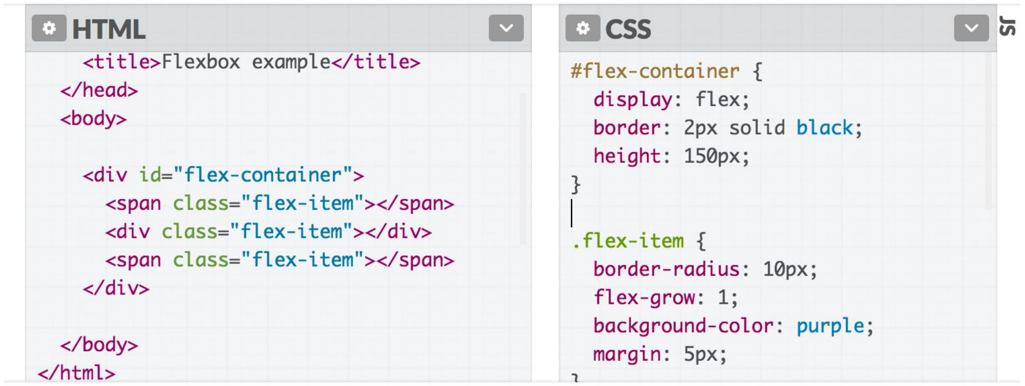
*width in the case of rows; height in the case of columns

**height in the case of rows; width in the case of columns

`align-items: stretch;`

The default value of `align-items` is `stretch`, which means every flex item grows vertically* to fill the container by default.

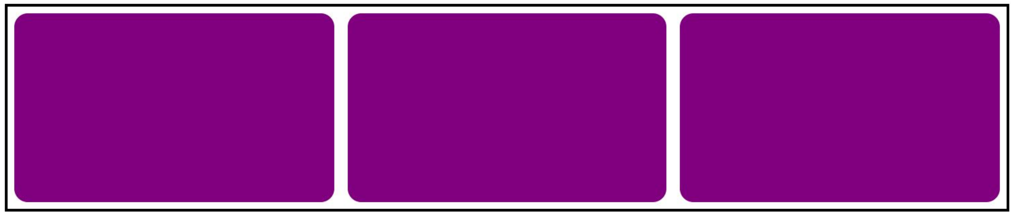
(This will not happen if the `height` on the flex item is set)


HTML

```html
    <title>Flexbox example</title>
  </head>
  <body>

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

  </body>
</html>
```

CSS

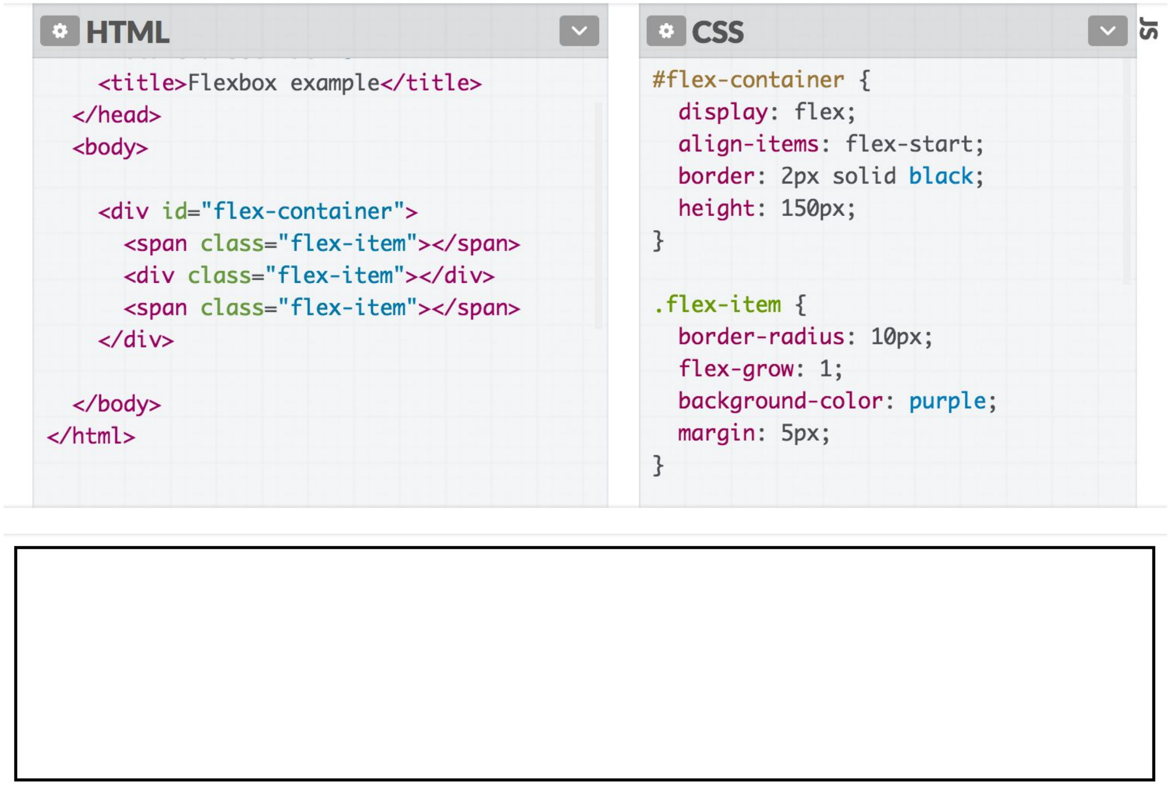
```css
#flex-container {
  display: flex;
  border: 2px solid black;
  height: 150px;
}

.flex-item {
  border-radius: 10px;
  flex-grow: 1;
  background-color: purple;
  margin: 5px;
```



*vertically in the case of rows; horizontally in the case of columns

`align-items: stretch;`

If we set another value for `align-items`, the flex items disappear again because the height is now content height, which is 0:



HTML

```html
    <title>Flexbox example</title>
  </head>
  <body>

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

  </body>
</html>
```

CSS

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

.flex-item {
  border-radius: 10px;
  flex-grow: 1;
  background-color: purple;
  margin: 5px;
}
```

FLEXBOX FROGGY

Most entertaining way to learn and practice Flexbox:

https://flexboxfroggy.com/

More next time!