CS193X: Web Programming Fundamentals

Spring 2017

Victoria Kirst
(vrk@stanford.edu)

Today's schedule

Schedule:

- HTML: Background and history
- Complex selectors
- Box model
- Debugging with Chrome Inspector
- **Case study**: Squarespace Layout (will continue into Monday)

(Forgot to mention: Paths)

`img src`, `a href`, and `link href` can all take either **relative** or **absolute** paths to the resource:

- `<a href="about.html">About</a>`
- `<img src="http://i.imgur.com/WJToVGv.jpg" />`
- `<link rel="stylesheet" href="css/style.css"/>`

If you are unfamiliar with paths, check out the following:

- Absolute vs relative paths
- Unix directories and file paths
- If anything's still unclear, come to office hours!

HTML: Background and History

What HTML elements can I use?

Q: Instead of `<span class="highlight"></span>`, can I create a `<highlight>` element?

```
<p>
  The homework is
  <highlight>due Friday</highlight>.
</p>
```

```
  highlight {
    background-color: yellow;
  }
```

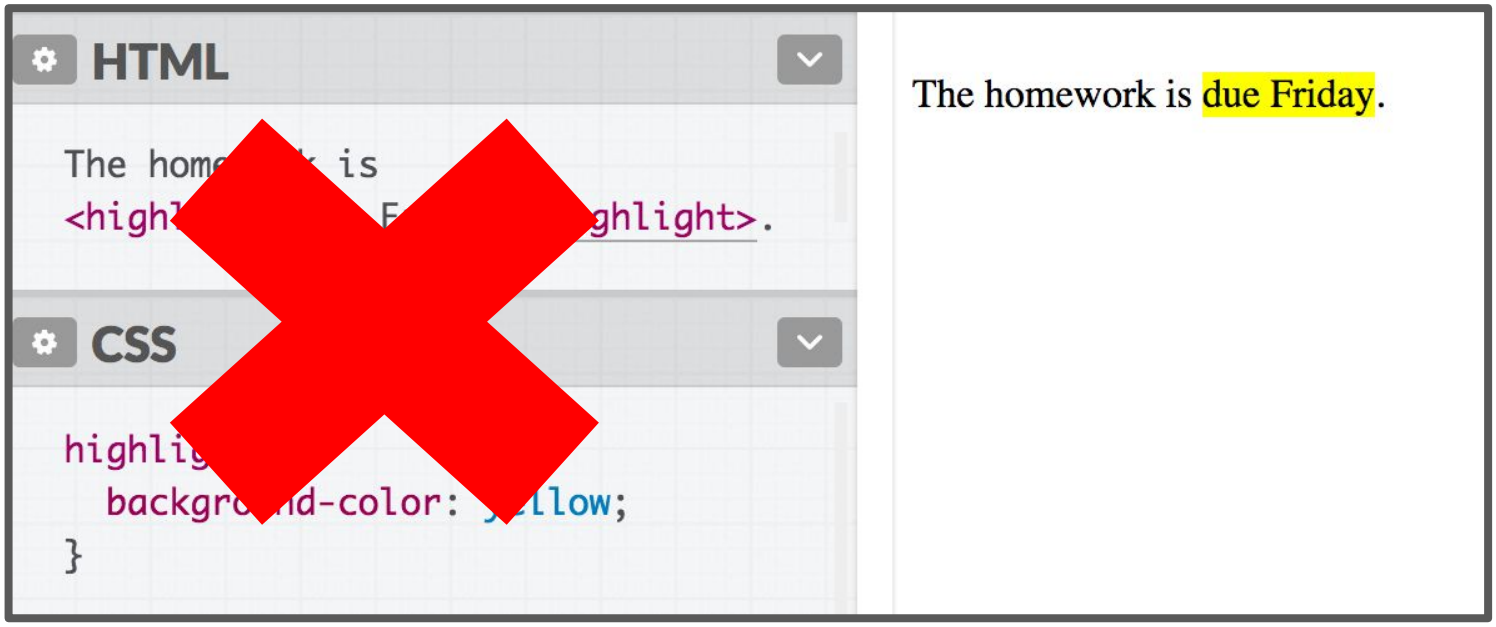
Q: Does this even work?

What HTML elements can I use?

This renders correctly:

What HTML elements can I use?

This renders correctly:

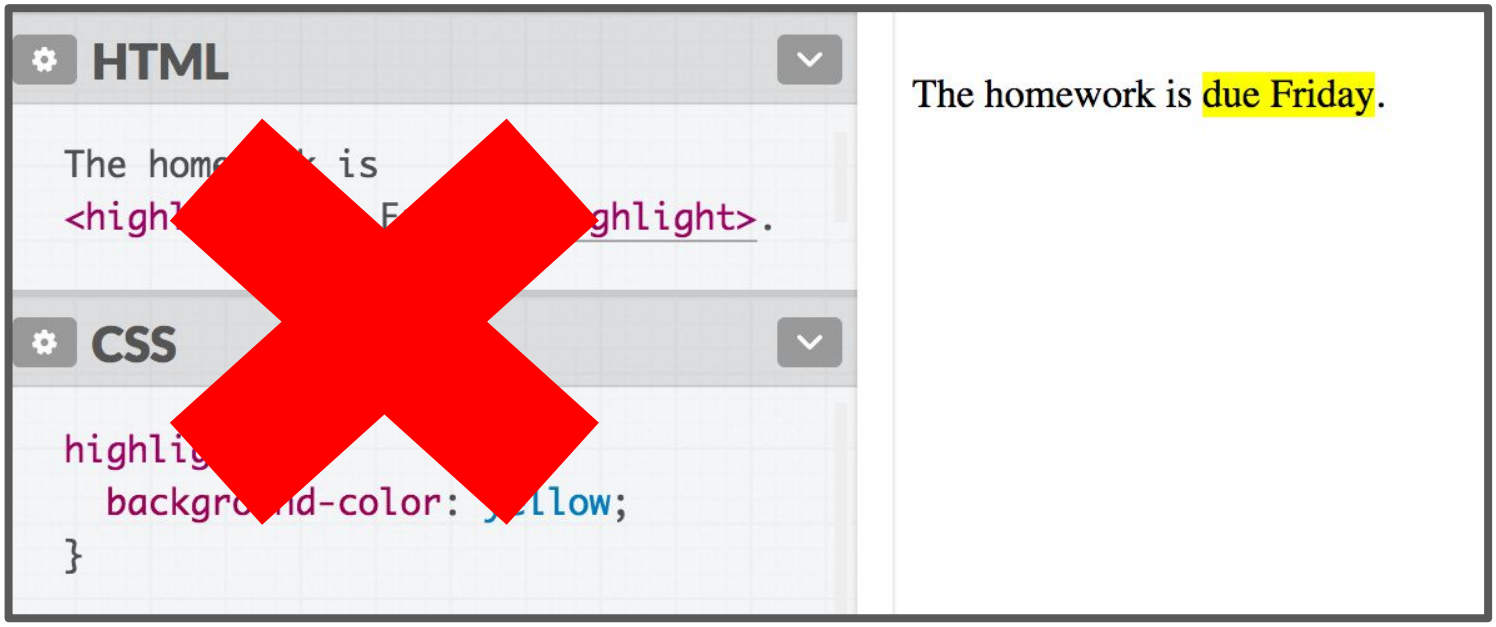


But you shouldn't do this!

It is non-standard behavior.

What HTML elements can I use?

This renders correctly:



What?!?!?

But you shouldn't do this!

It is non-standard behavior.

What?!

- What is "standard" HTML?
- Why does invalid HTML/CSS still work sometimes?
  - If my Java code is wrong, I get a compiler error… If my HTML or CSS is wrong, why don't I get an error?
- Why does it matter that I follow "standard" HTML?

A very brief history of HTML

History

Tim Berners-Lee

- 1989: World Wide Web created
  (WWW: web pages and the protocol in which they are served HTTP/HTTPS)

- 1994: World Wide Web Consortium created
  - "**W3C**": Goal to maintain and develop standards about how the web should work
  - Oversees several languages:
    - HTML, CSS, DOM, XML, etc

- 1997: "HTML4" published
  - The first major stable version of HTML

Degrading gracefully

The W3C HTML spec lists several design principles, and one is degrading gracefully:

"An escalator can never break: it can only become stairs"

This is why browsers do a **best-effort** to render non-standard ("invalid") HTML and CSS.

Best-effort rendering

It's also why `<highlight>` "works", even though it's Invalid HTML.

Why not enforce strict HTML?

It's super weird that:

- Browsers don't fail when given invalid HTML / CSS
- Browsers not only don't fail, but they render invalid HTML/CSS seemingly "correctly"

Q: Why doesn't the browser reject poorly written HTML/CSS?

Why not enforce strict HTML?

It's super weird that:

- Browsers don't fail when given invalid HTML / CSS
- Browsers not only don't fail, but they render invalid HTML/CSS seemingly "correctly"

Q: Why doesn't the browser reject poorly written HTML/CSS?

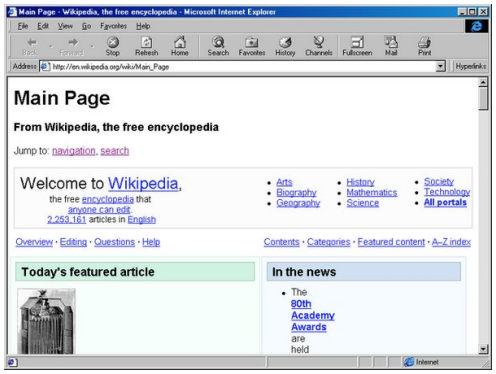
A: There was a (failed) attempt to enforce this, but it was too late: the Internet grew too big!

The nerdy, mostly* accurate backstory for HTML today

*I would be more accurate, but it's hard to get valid sources online… so I'm going off of what I can + the lore I've heard while working on a browser.

State of the world, 1997:

Standards say one thing,

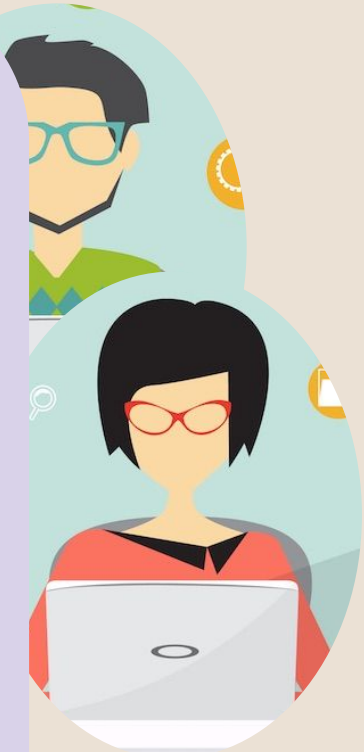


Browsers do another thing,

Developers write weird, non-standard code.

State of the world, 1997:

Standard
one thing,



lopers write
rd, non-standard
code.

In 1997, things are kind of a mess!

another thing,

2000ish:

W3C

Oops, that "degrading gracefully" thing was a mistake.

Browsers, stop rendering pages that have invalid HTML.

(This what the proposal of XHTML 1.1)

2000ish: (not totally accurate)

W3C

What?!?!

Sure whatever

That would break the internet!

We can't do that!

2004: WHATWG formed

Let's burn everything and start from scratch with XHTML 1.1

(break approx. 64 million websites)

WHATWG

Let's work on HTML5

(an imperfect but realistic standard)

Fast forward 2017?!

- W3C gave up XHTML 1.1 in 2007
- W3C and WHATWG are mostly friends (I think), though they are still separate entities
- Can still find some snarky quotes on WHATWG website

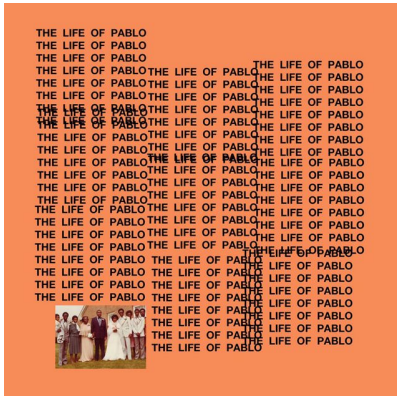
"HTML5" vs HTML

W3C maintains HTML5:

- More stable version of WHATWG's HTML
- Usually copies what WHATWG does after the dust settles

WHATWG maintains HTML: The Living Standard

- No number, no versions
- Updated frequently and being updated today!
- Most browsers implement WHATWG
- This is why I don't say "HTML5"



What you need to know

Q: What HTML elements can I choose from?

- Check MDN's list of HTML tags

Q: How do I know if an HTML tag (or CSS property, or JS feature) **is implemented on all browsers?**

- Check caniuse.com

Q: Why shouldn't I use non-standard HTML/CSS/JavaScript, even if it works in every browser?

What you need to know

Q: What HTML elements can I choose from?

- Check MDN's list of HTML tags

Q: How do I know if an HTML tag (or CSS property, or JS feature) **is implemented on all browsers?**

- Check caniuse.com

Q: Why shouldn't I use non-standard HTML/CSS/JavaScript, even if it works in every browser?

- Because it won't be guaranteed to work in the future
- Because it won't be guaranteed to work on all "user agents" (not just browsers)

What you need to know

Q: Wouldn't it be super useful to create custom elements?

- Yes! There is a spec for this currently under development.
  - (Note that custom elements are not really meant for our example; custom elements are meant for defining custom behavior and not just style. For defining style, CSS classes/ids are still most appropriate.)

Back to writing code!

CSS Selectors: Classes and Ids

Classes and ids

There are 3 basic types of CSS selectors:

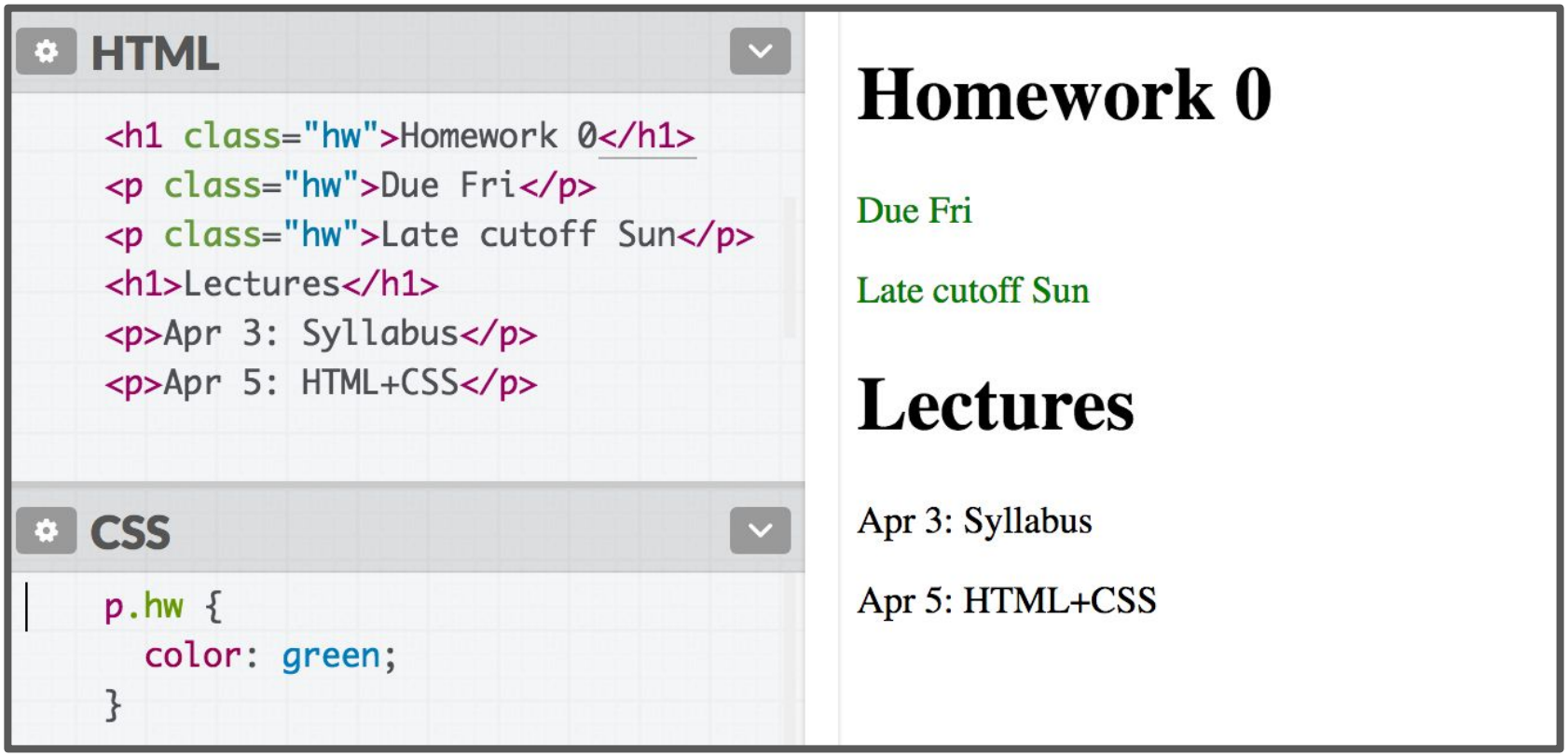
Element selector (this is the one we've been using)	**p**	All **<p>** elements
✧ **ID selector** ✧	**#abc**	element with **id="abc"**
✧ **Class selector** ✧	**.abc**	elements with **class="abc"**

```
<h1 id="title">Homework</h1>
<em class="hw">HW0</em> is due Friday.<br/>
<em class="hw">HW1</em> goes out Monday.<br/>
<em>All homework due at 11:59pm.</em>
```

Other selectors

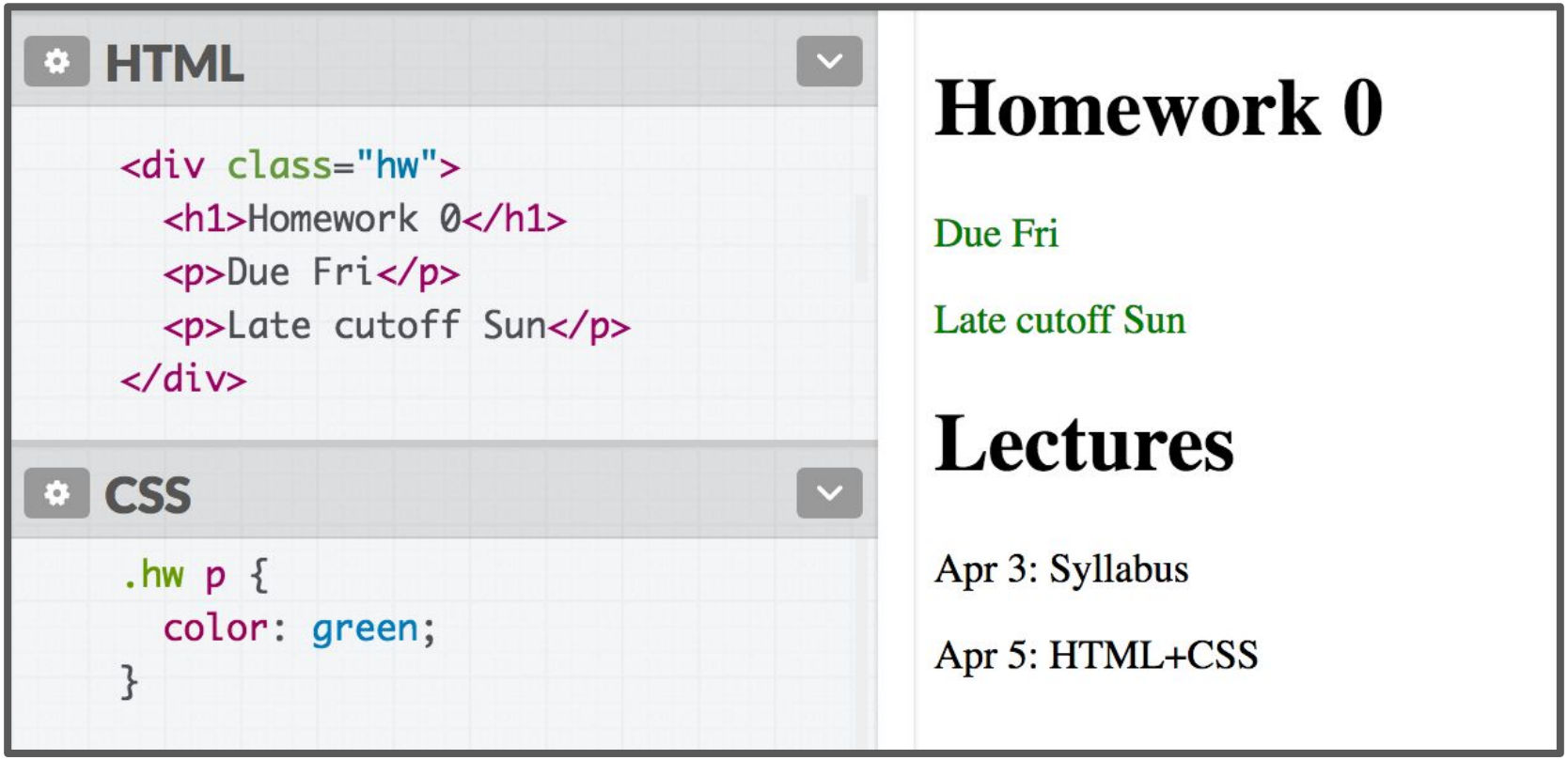
element.className

Syntax	Example	Example described
element.className	**p.abc**	**<p>** elements with abc class



Descendent selector

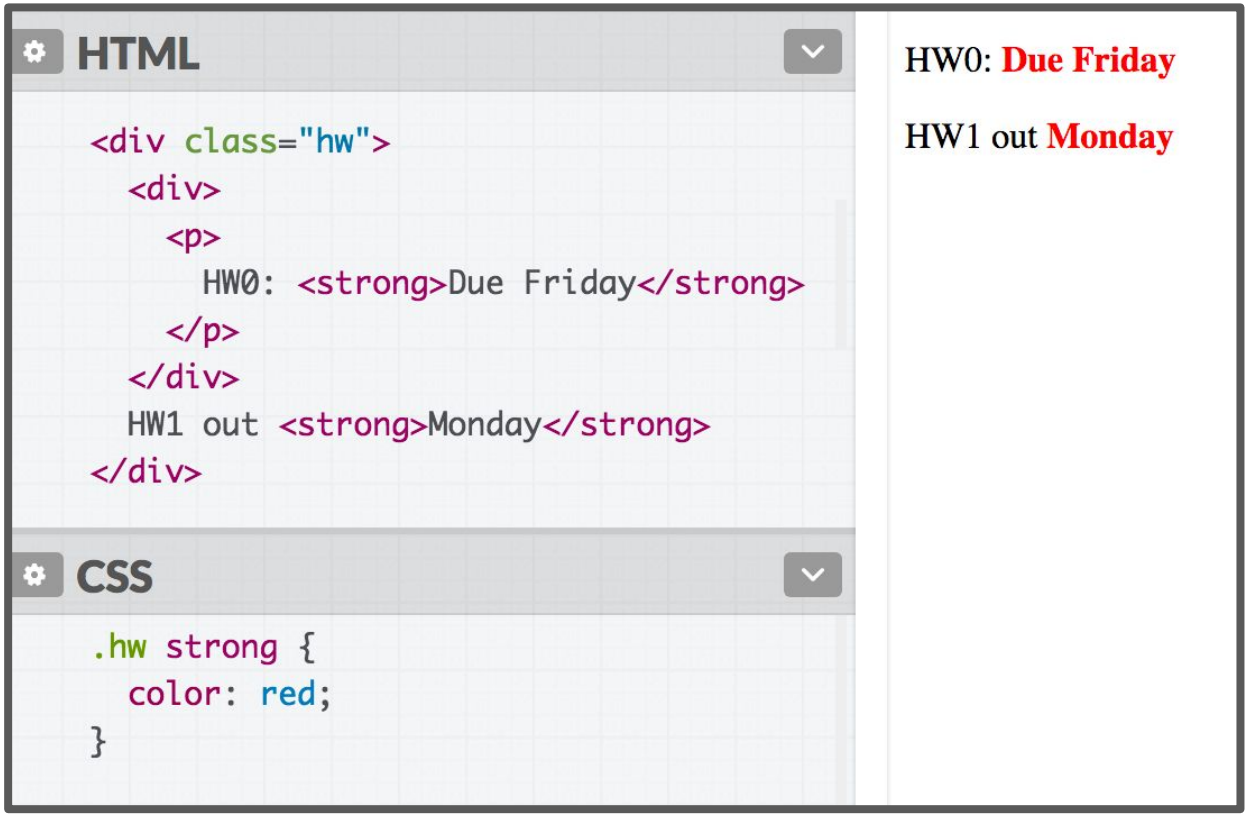
Syntax	Example	Example described
selector selector	`div strong`	`<strong>` elements that are descendants of a `<div>`



Descendent selector

Syntax	Example	Example described
selector selector	`div strong`	**** elements that are descendants of a ** **

Note: The element does not have to be a direct child. The descendent may be nested many layers in.



Descendent selector

Syntax	Example	Example described
selector selector	`div strong`	**** elements that are descendants of a ** **

Discouraged:

```
<h1 class="hw">Homework 0</h1>
<p class="hw">Due Fri</p>
<p class="hw">Late cutoff Sun</p>
```

vs

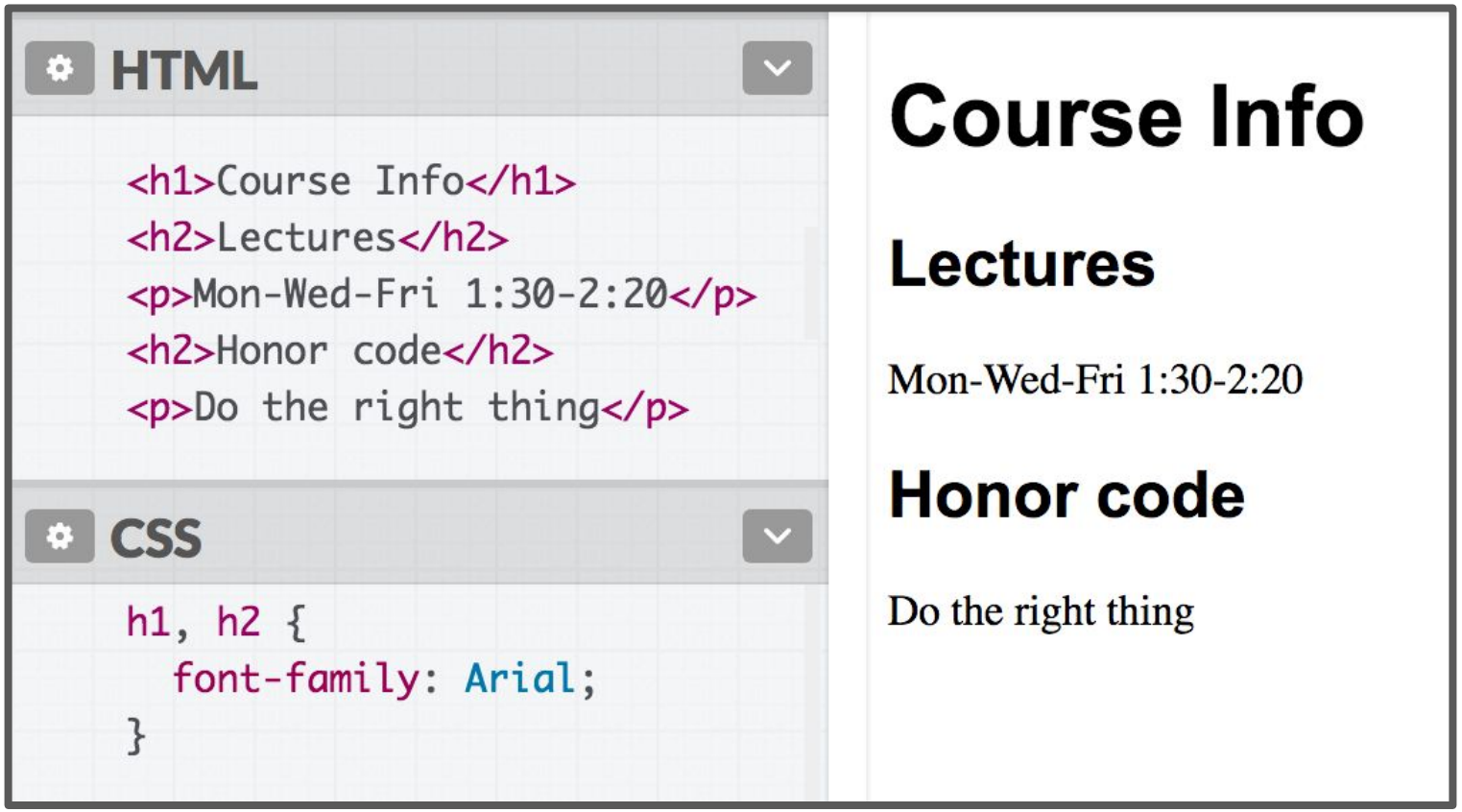
Preferred:

```
<div class="hw">
  <h1>Homework 0</h1>
  <p>Due Fri</p>
  <p>Late cutoff Sun</p>
</div>
```

Instead of applying a class to several adjacent elements, wrap the group in a `<div>` container and style the contents via descendent selectors.

selector, selector (comma)

Syntax	Example	Example described
selector, selector	**`h2, div`**	**`<h2>`** elements and **`<div>`**s



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

Grouping selectors

2 Common bugs:

`p.abc` **vs** `p .abc`

`p .abc` **vs** `p, .abc`

- A <p> element with the **abc** class **vs**
  An element with the **abc** class that descends from **<p>**

- An element with the **abc** class that descends from **<p>** **vs**
  All **<p>** elements *and* all elements with the **abc** class

Combining selectors

You can combine selectors:

```
#main li.important strong {
  color: red;
}
```

Q: What does this select?

Grouping selectors

Q: What does this select?

```
#main li.important strong {
  color: red;
}
```

A: Read from right to left:

- `<strong>` tags that are children of `<li>` tags that have an "`important`" class that are children of the element with the "`main`" id.

Colliding styles

When styles collide, the most specific rule wins (specificity)

```
div strong { color: red; }
strong { color: blue; }

<div>
   <strong>What color am I?</strong>
</div>
```

Colliding styles

When styles collide, the most specific rule wins (specificity)

```
div strong { color: red; }
strong { color: blue; }

<div>
   <strong>What color am I?</strong>
</div>
```

Colliding styles

Specificity precedence rules (details):

- `id`s are more specific than `class`es
- `class`es are more specific than element names
- Style rules that directly target elements are more specific than style rules that are inherited

Colliding styles

- If elements have the same specificity, the later rule wins.

```
strong { color: red; }
strong { color: blue; }

<div>
   <strong>What color am I?</strong>
</div>
```

Aside: The process of figuring out what rule applies to a given element is called the cascade. This is where the "C" in *Cascading* Style Sheets comes from.

Inheritance

We saw earlier that CSS styles are inherited from parent to child.

Instead of selecting all elements individually:

```
a, h1, p, strong {
  font-family: Helvetica;
}
```

You can style the parent and the children will inherit the styles.

```
body {
  font-family: Helvetica;
}
```

You can override this style via specificity:

```
h1, h2 {
  font-family: Consolas;
}
```

Inheritance

While many CSS styles are inherited from parent to child, **not all CSS properties are inherited**.

```
a {
  display: block;
  font-family: Arial;
}
```

```
<a href="/home">
  Back to <em>Home</em>
</a>
```

 inherits the font-family property, but not display:

Back to *Home*

Inheritance

While many CSS styles are inherited from parent to child, **not all CSS properties are inherited**.

- There's no rule for what properties are inherited or not; the inheritance behavior defined in the CSS spec.
- You can look it up via MDN, e.g.
  `font-family`: Inherited yes
  `display`: Inherited no
- Generally text-related properties are inherited and layout-related properties are not.
- (You can also change this via the `inherit` CSS property, which is somewhat esoteric and not often use)

<a> colors?

Hmm, MDN says color is inherited… but if I set the body color to deeppink, links don't change color:

CSS

```
body {
  color: deeppink;
  font-family: Helvetica;
}
```

HTML

```
<h1>Chocolate</h1>
<p>
  <a href="https://www.ghirardelli.com/">Ghiradelli</a>
  is not overrated
</p>
```

<a> inherits font-family…

Why doesn't <a> inherit color?

(Codepen)

User agent styles

This is because the browser has its own default styles:

- Browser loads its own default stylesheet on every webpage
- Not governed by spec, but there are recommendations

```
<!DOCTYPE html>
<html>
  <head>
    <title>CS 193X</title>
    <!--
      NOT TOTALLY ACCURATE: This isn't actually injected
      in the HTML, but it is loaded silently!
    -->
    <link rel="stylesheet" href="user-agent-style.css" />
  </head>
```

<a> colors?

So to style <a> links, we have to override the browser default link style by explicitly setting a color:

CSS

```css
body {
  color: deeppink;
  font-family: Helvetica;
}

a {
  color: deeppink;
}
```

HTML

```html
<h1>Chocolate</h1>
<p>
  <a href="https://www.ghirardelli.com/">Ghiradelli</a>
  is not overrated
</p>
```

Link-related CSS

Since we're on the topic of links:

- How do we style **visited** links differently from **unvisited**?

CSS pseudo-classes

pseudo-classes: special keywords you can append to selectors, specifying a *state* or *property* of the selector

Syntax	Explanation
`a`	All anchor tags (links) in all states
`a:visited`	A visited link
`a:link`	An unvisited link
`a:hover`	The style when you hover over a link
`a:active`	The style when you have "activated" a link (downclick)

There are more **pseudo-classes** than this; have a look!

Before we move on:
A few style notes

Why not `<div>` everywhere?

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

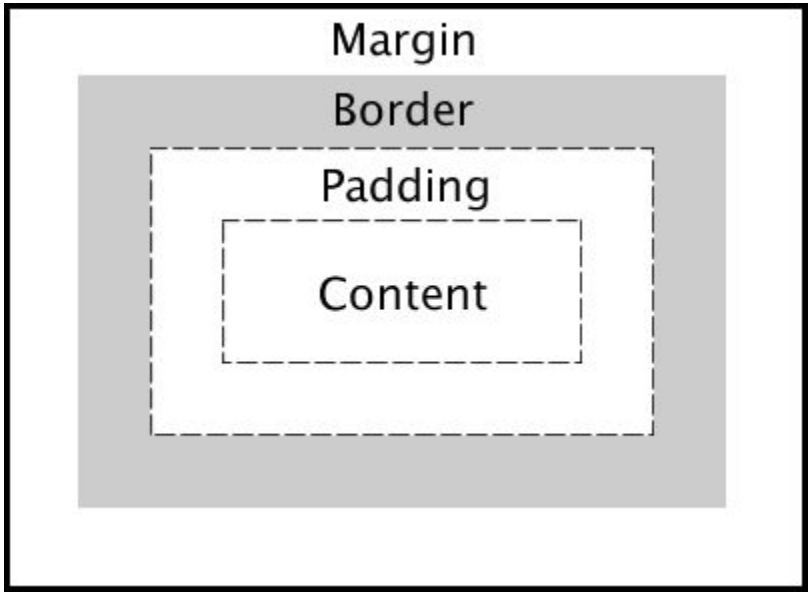
- Is this a good idea?
- Why does HTML have `id`s when you have `class`es?
- Why does HTML have `<p>`, `<h1>`, `<strong>`, etc. when you have `<div>`, `<span>`, `class`, and `id`?

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

```
<p>
  Welcome to CS193X: Web Programmi
  In this class, you will learn mod
</p>
```

CSS

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

Welcome to CS193X: Web Programming Fundamentals! In this class, you will learn modern full-stack web development techniques.

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 in the next couple lectures.

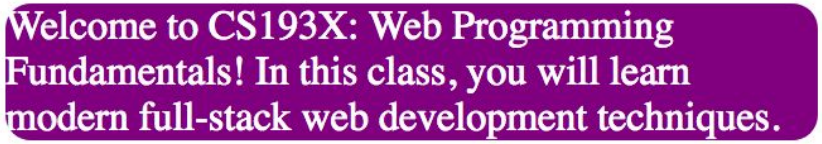
Rounded border

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

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

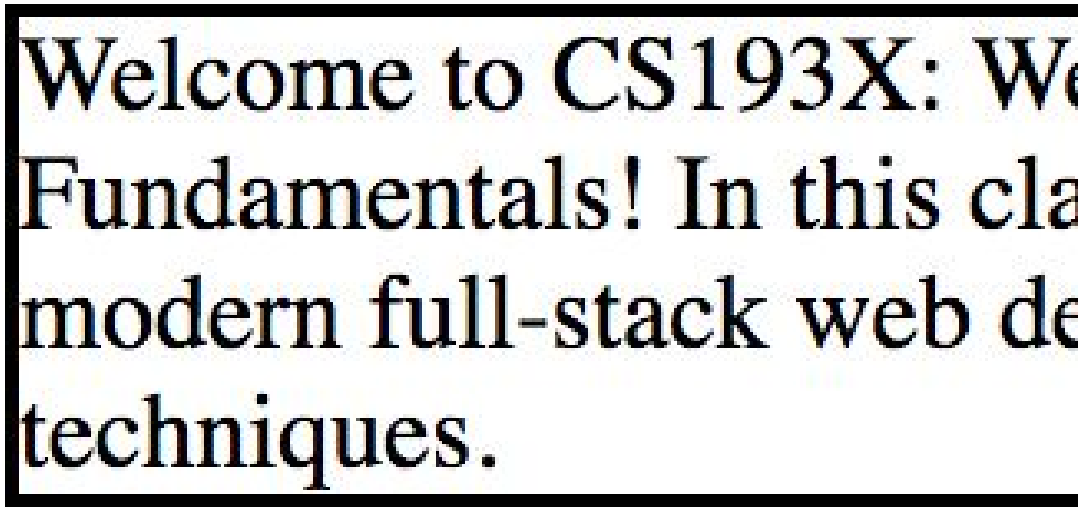
You don't actually need to set a border to use border-radius.

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



Borders look a little squished

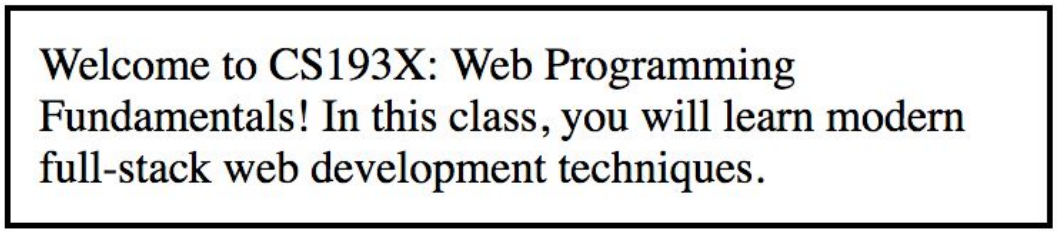
When we add a border to an element, it sits flush against the text:



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

padding

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



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|left|bottom|right
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:

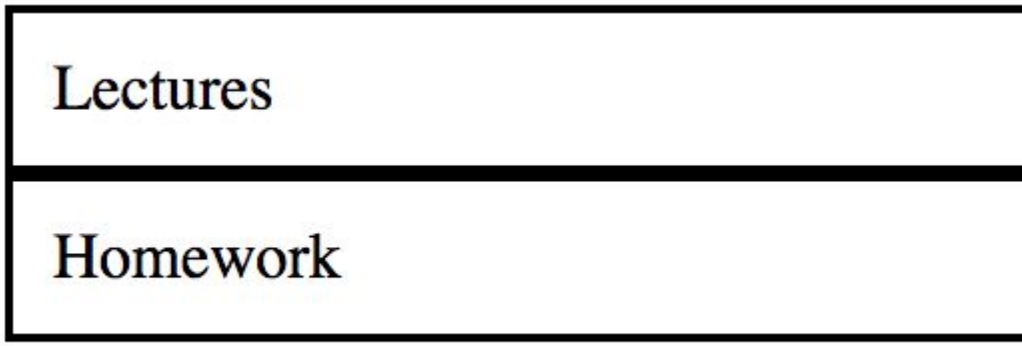
HTML

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

CSS

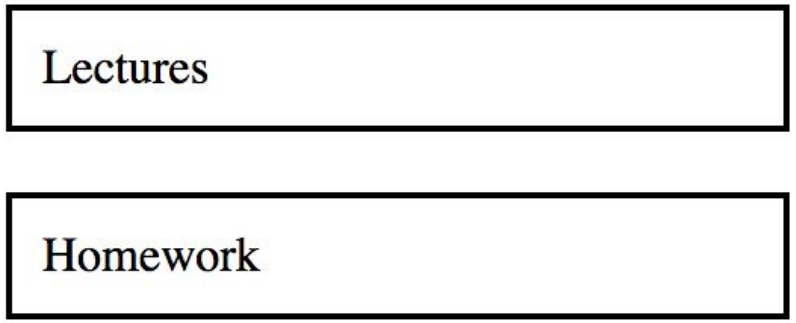
```
div {
  border: 2px solid black;
  padding: 10px;
}
```

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

More box model:
Next time!

Overflow slides

We didn't get to these in lecture

margin

Actually, why doesn't this:

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

Lectures

Homework

Look more like this?

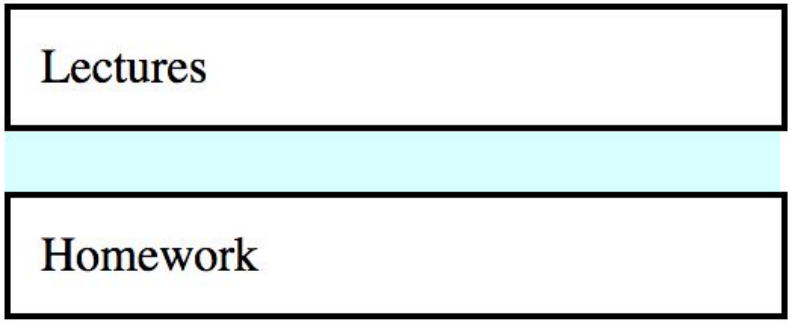
Lectures

Homework

margin

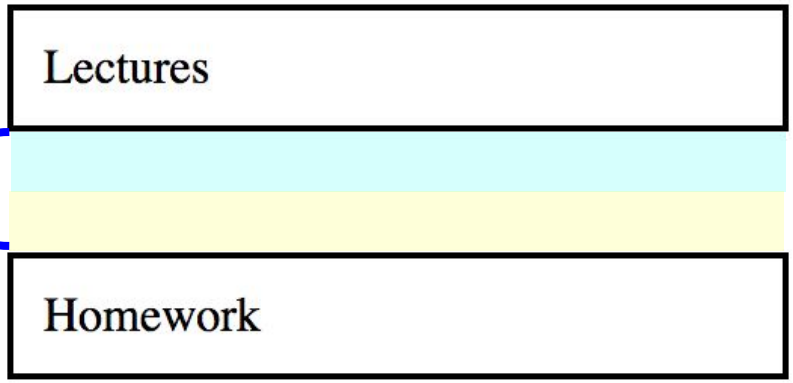
Actually, why doesn't this:

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



...look more like this?

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



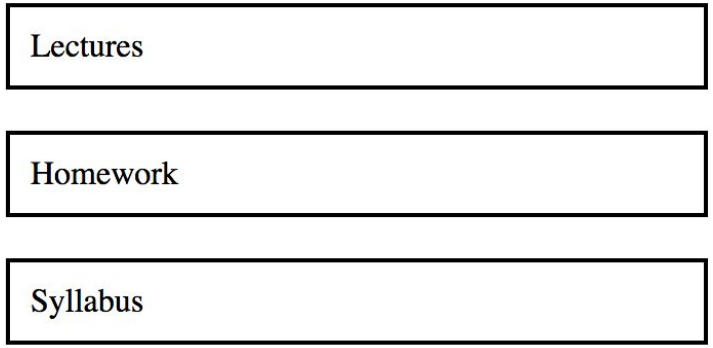
`margin` collapsing

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.

- `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;
}
```

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.

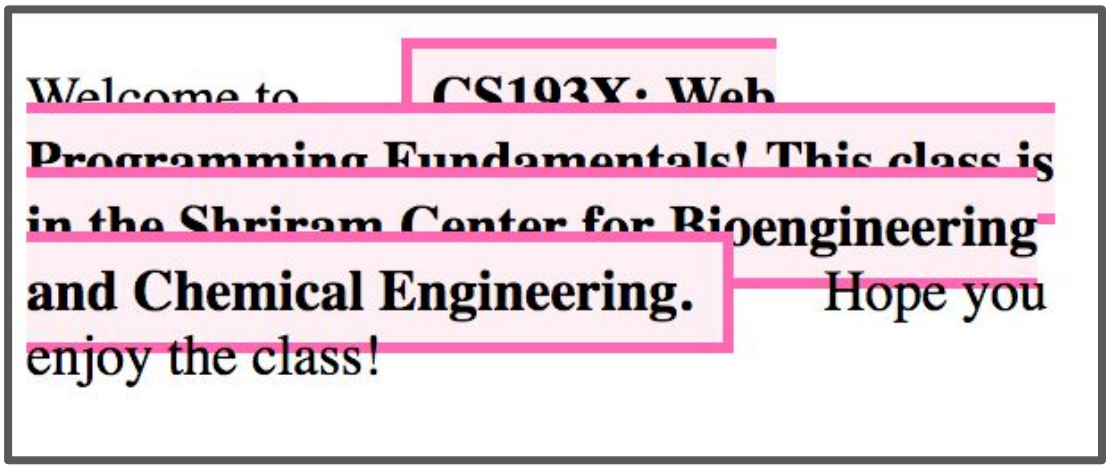
CSS

```
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.

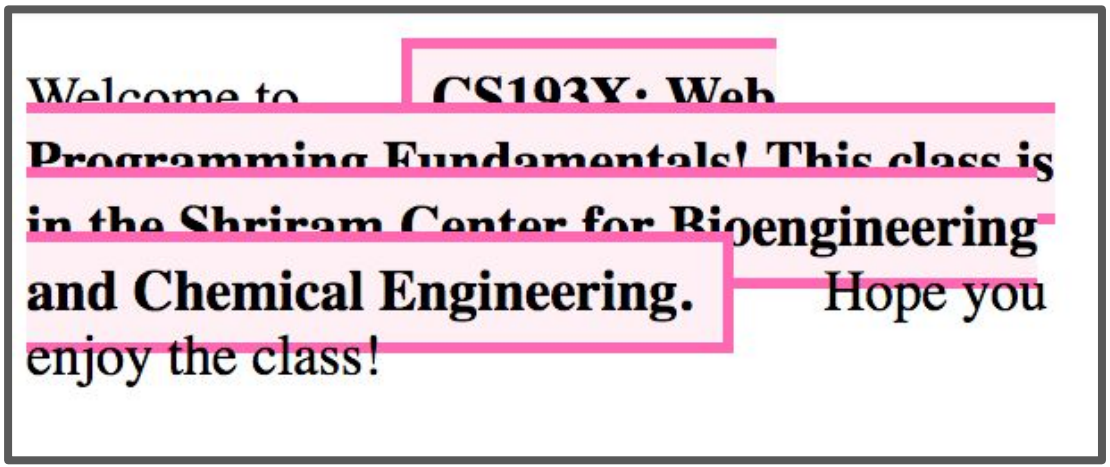
CSS

```
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>
```

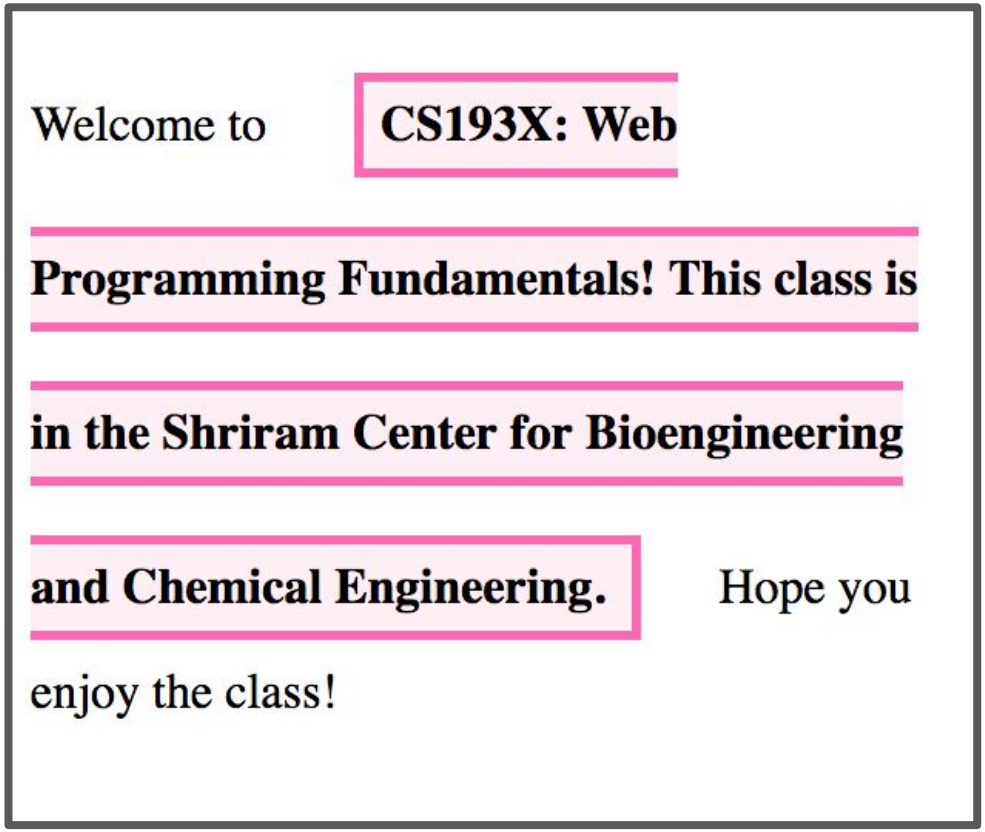


Let's change the line height to view this more clearly...

Inline element box model

CSS

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

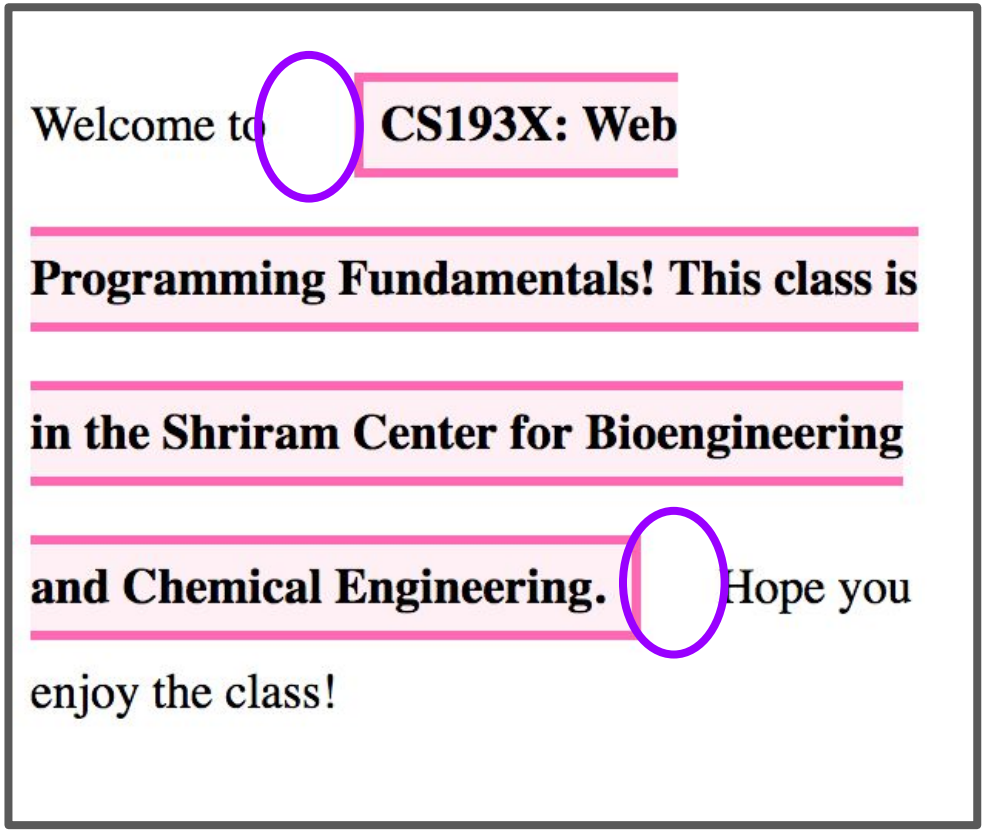


(Codepen)

Inline element box model

CSS

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

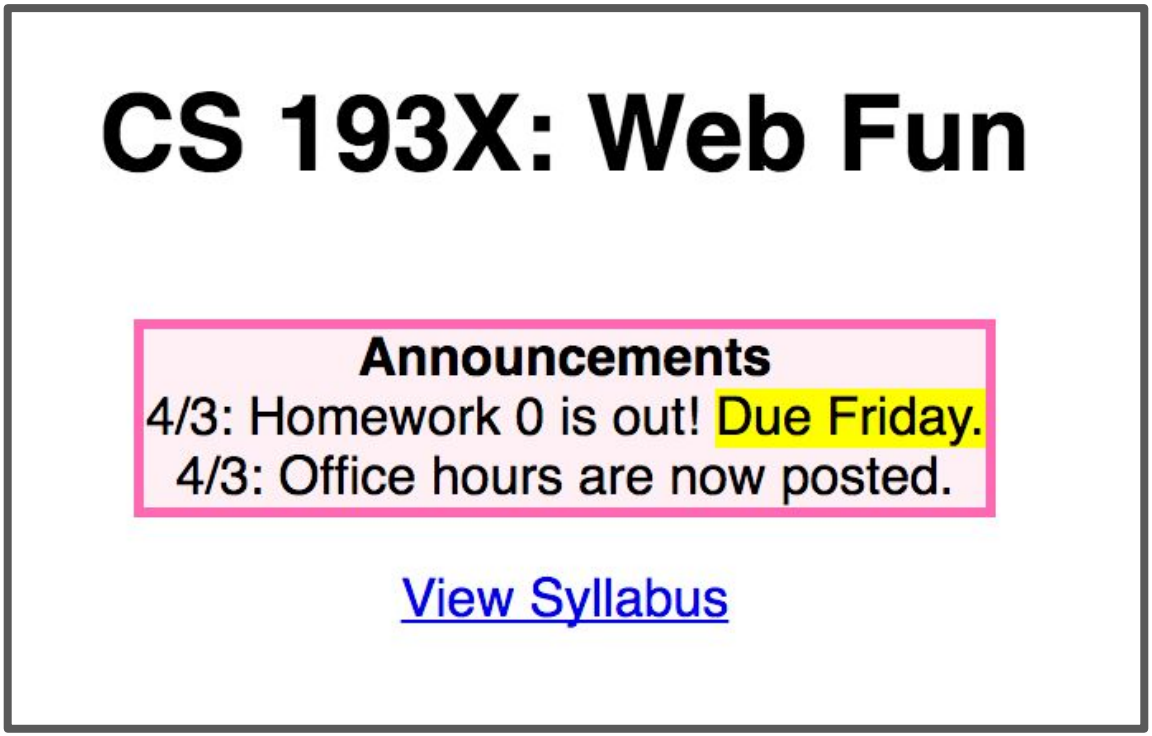


- **margin** is to the left and right of the inline element
  - `margin-top` and `margin-bottom` are **ignored**
- use **`line-height`** to manage space between lines

(Codepen)

The CSS Box Model

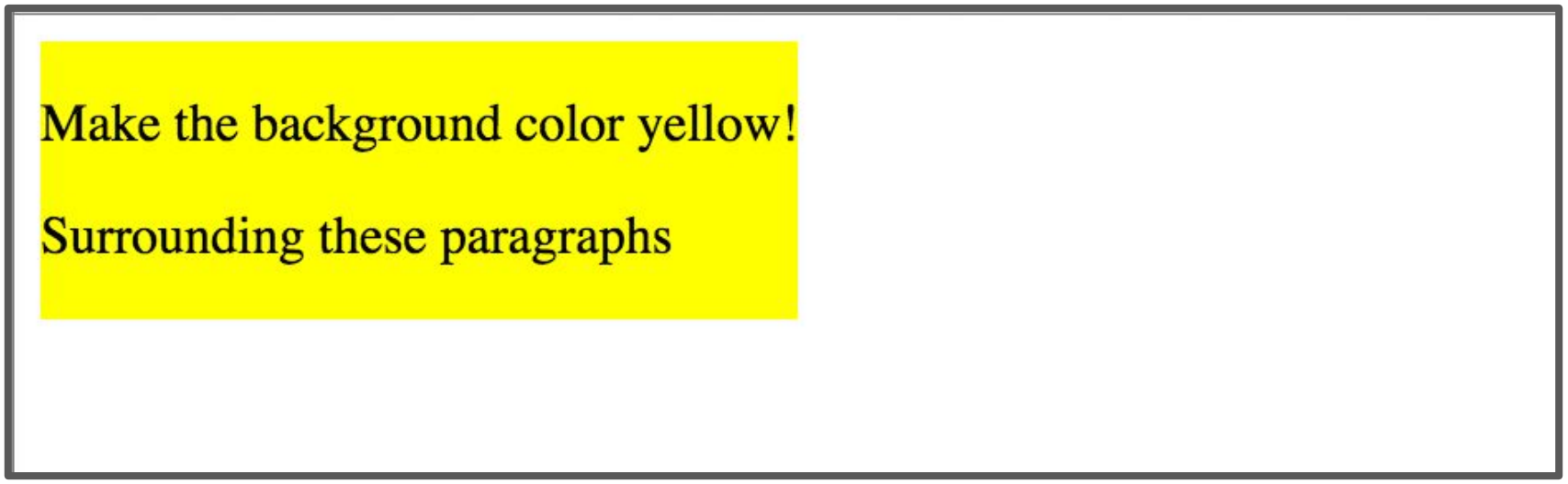
Let's revisit our Course web page example:



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?

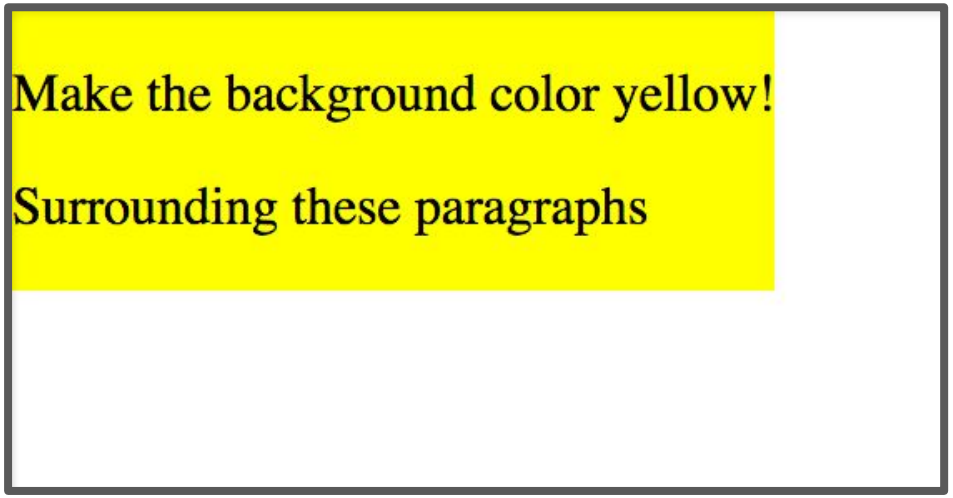
We can use the browser's Page Inspector to help us figure it out!

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:

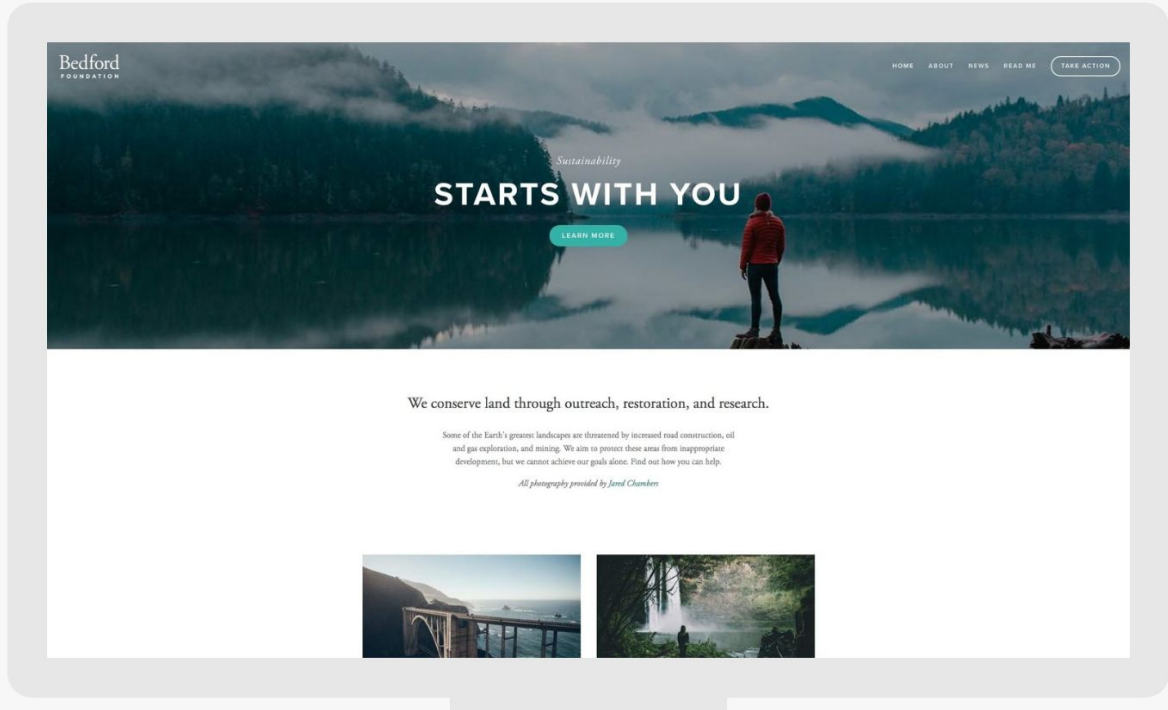
- **block vs inline** and the "natural" layout of the page, depending on the element type
- **`classes` and `ids`** and how to specify specific elements and groups of elements
- **`div` and `span`** and how to create generic elements
- **The CSS box model** and how every element is shaped like a box,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

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

Layout exercise

Squarespace template

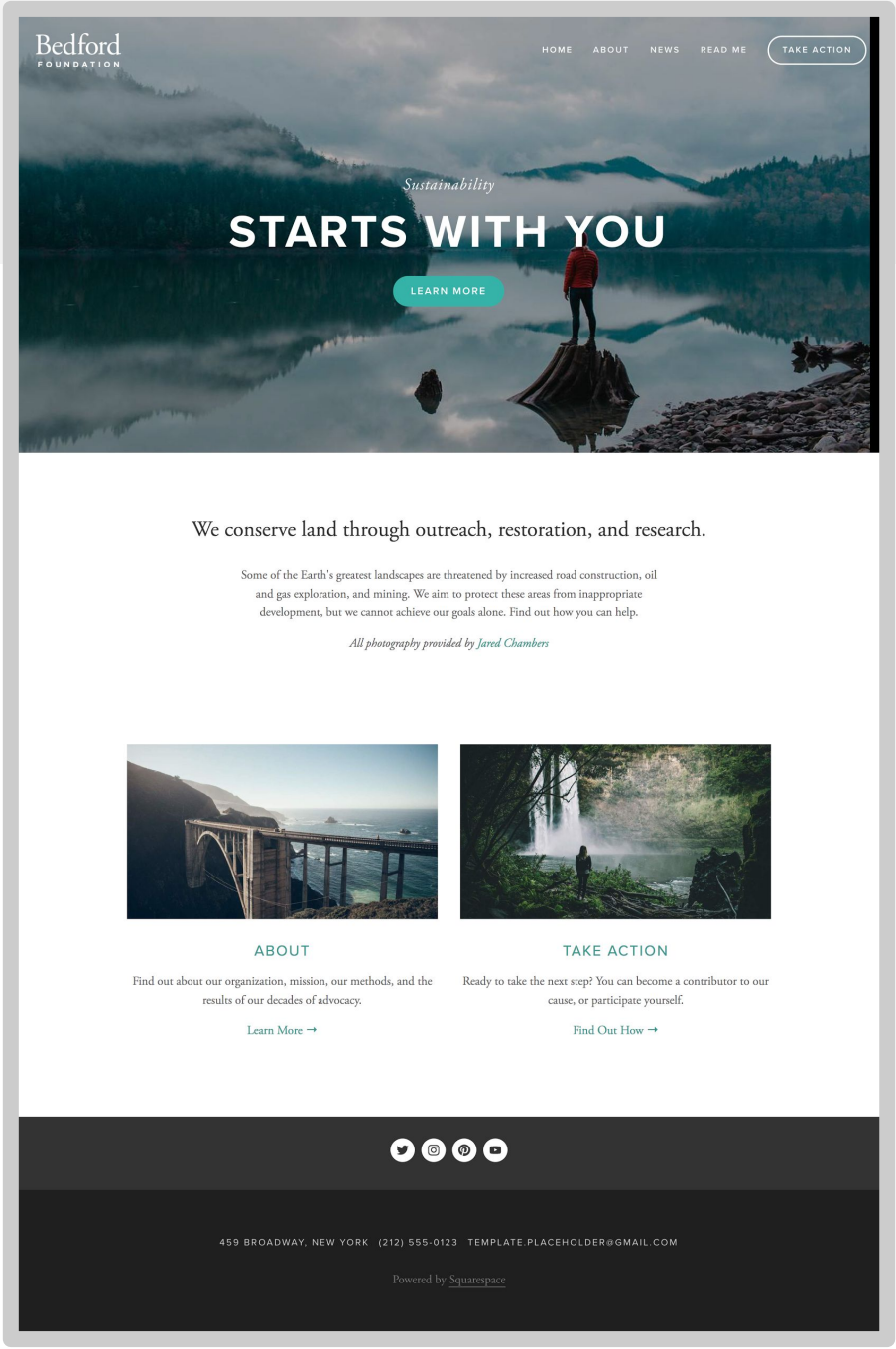
Squarespace's most popular template looks like this:



Do we know enough to make something like that?

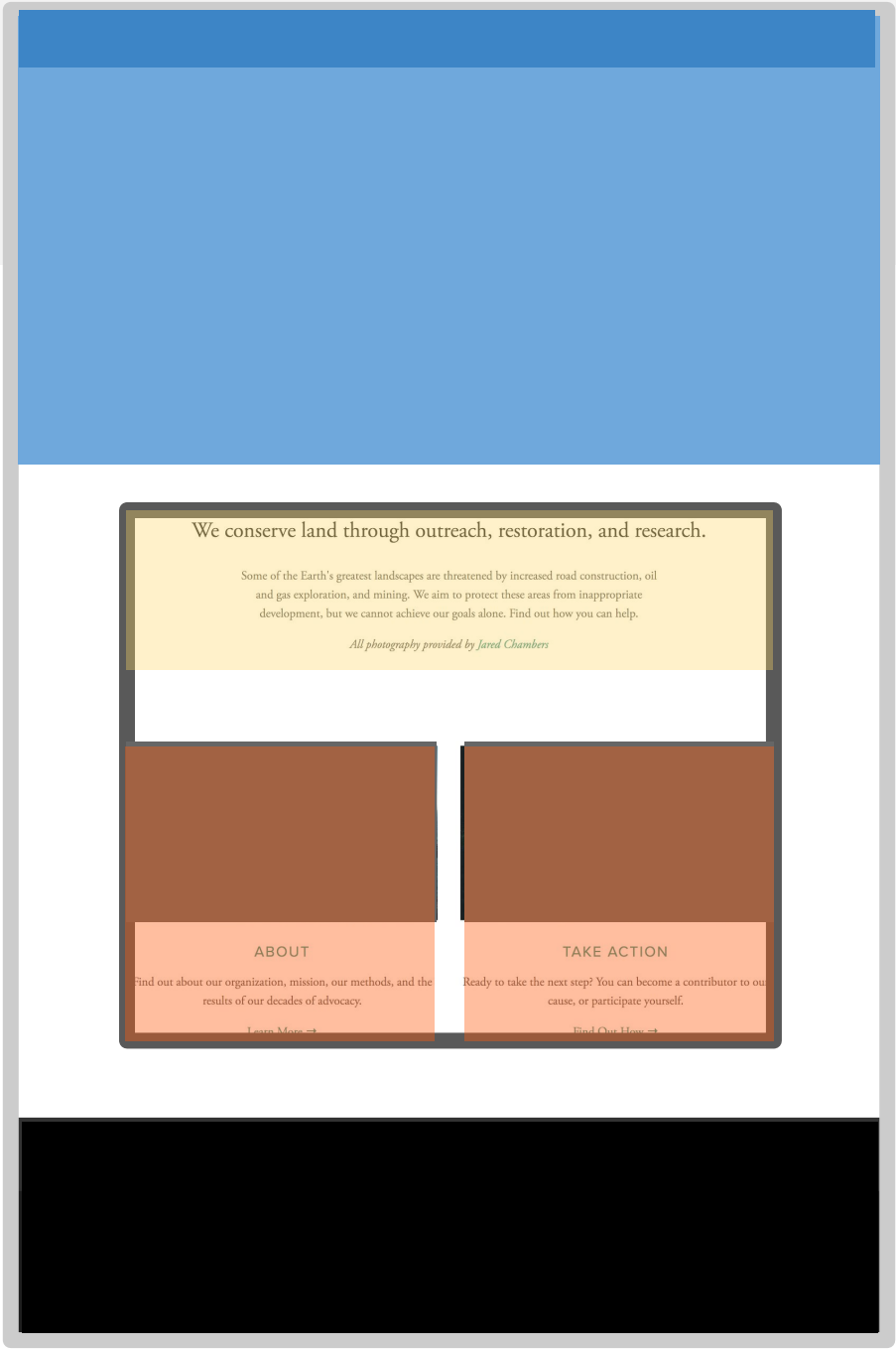
Basic shape

Begin visualizing the layout in terms of boxes:



Basic shape

Begin visualizing the layout in terms of boxes:

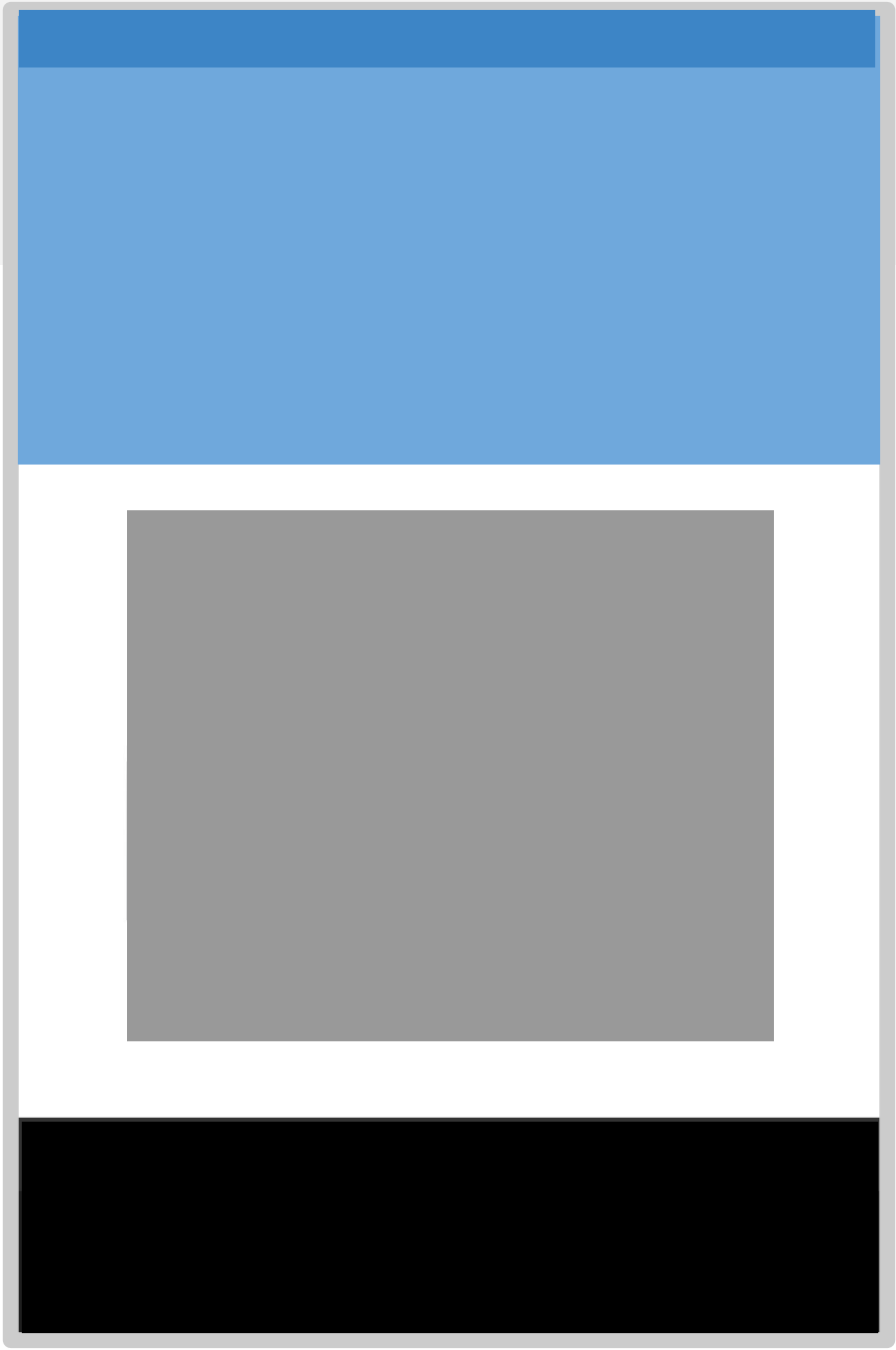


Basic shape

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.

Header

Navbar:

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

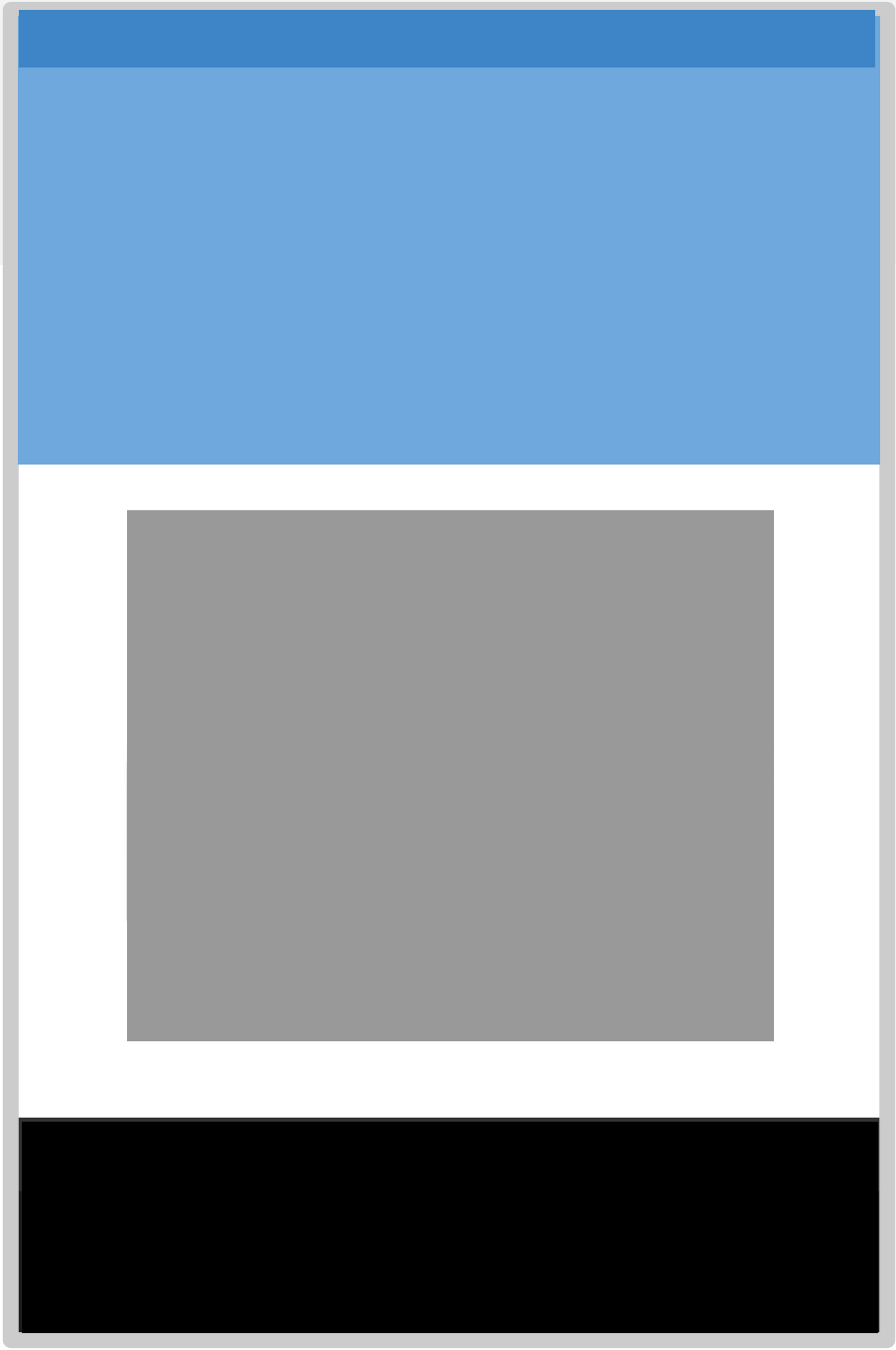
Header:

- 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>`

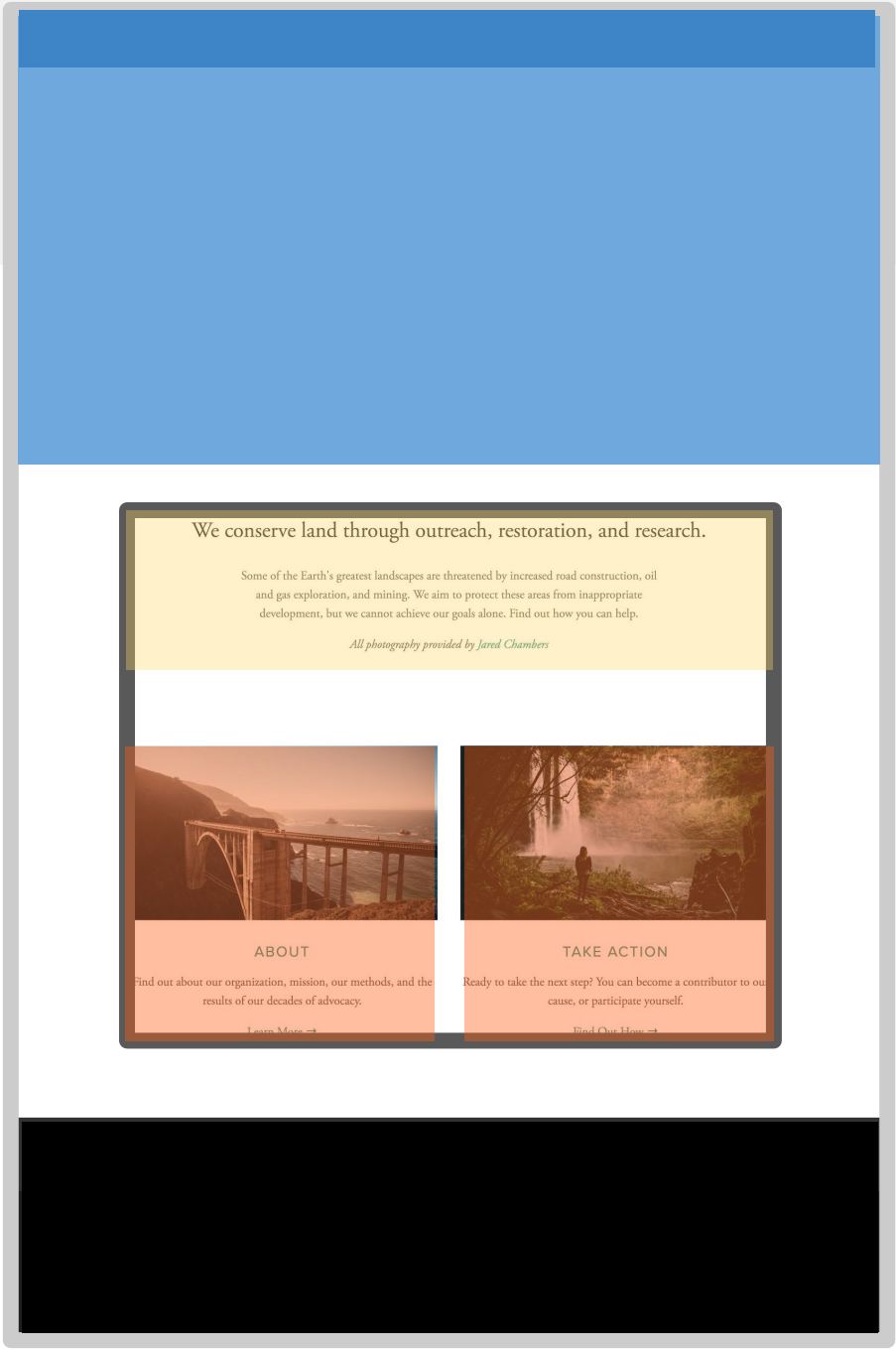
Main contents

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>`

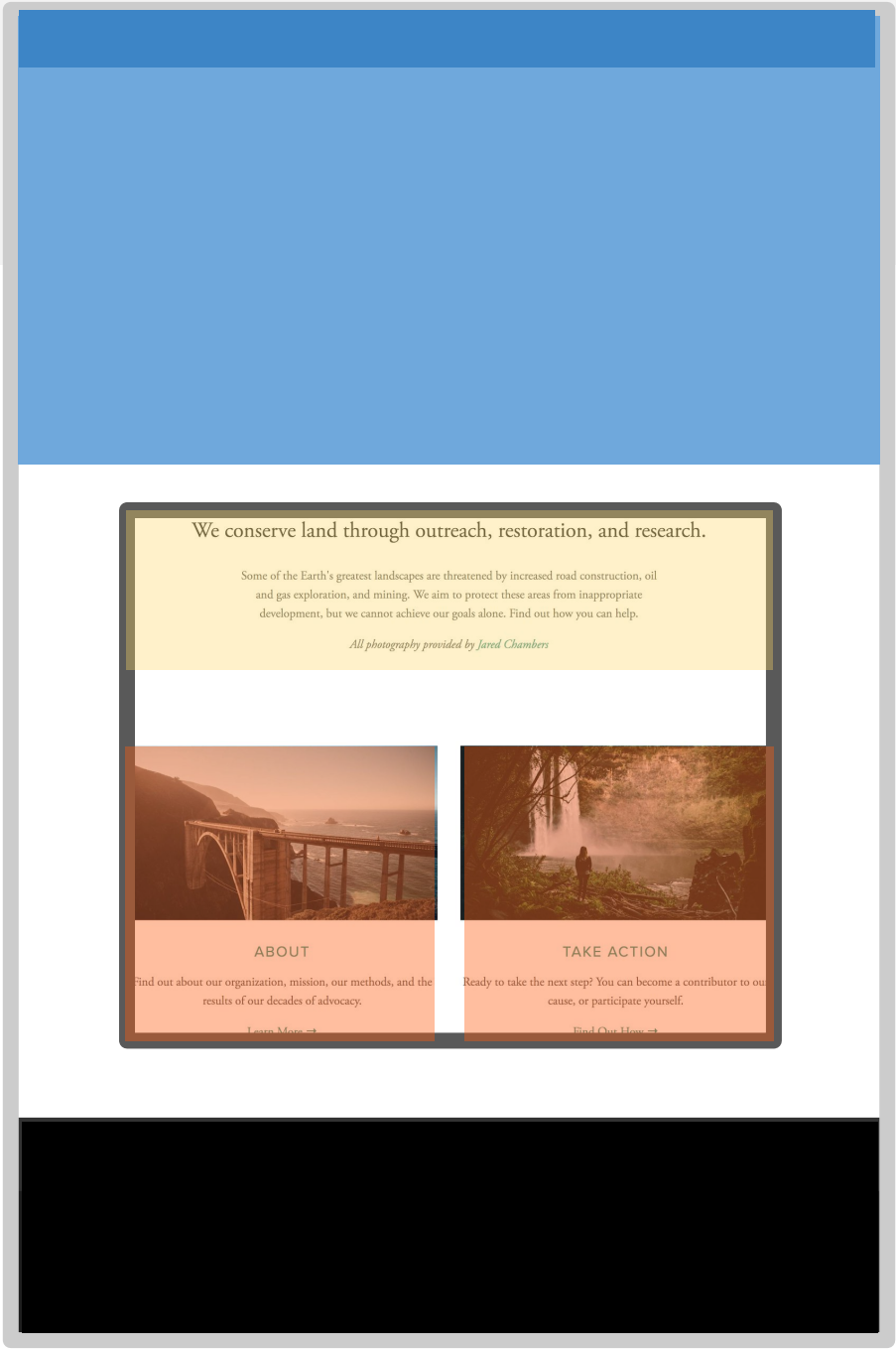


Main contents

Orange box:

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

This is where we get stuck.



Next time:
Flexbox to the rescue!