Today's schedule

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

Wednesday
- Mobile layouts
- CSS wrap-up: units, fonts

Friday
- Beginning JavaScript
HW1 released

Homework 1 is out today!
- Due Mon Apr 17 Extended to Wed Apr 19
- Details here
Quick review
## Selector summary

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>p</code></td>
<td>All <code>&lt;p&gt;</code> elements</td>
</tr>
<tr>
<td><code>.abc</code></td>
<td>All elements with the <code>abc</code> class, i.e. <code>class=&quot;abc&quot;</code></td>
</tr>
<tr>
<td><code>#abc</code></td>
<td>Element with the <code>abc</code> id, i.e. <code>id=&quot;abc&quot;</code></td>
</tr>
<tr>
<td><code>p.abc</code></td>
<td><code>&lt;p&gt;</code> elements with <code>abc</code> class</td>
</tr>
<tr>
<td><code>p#abc</code></td>
<td><code>&lt;p&gt;</code> element with <code>abc</code> id (p is redundant)</td>
</tr>
<tr>
<td><code>div strong</code></td>
<td><code>&lt;strong&gt;</code> elements that are descendants of a <code>&lt;div&gt;</code></td>
</tr>
<tr>
<td><code>h2, div</code></td>
<td><code>&lt;h2&gt;</code> elements and <code>&lt;div&gt;</code>s</td>
</tr>
</tbody>
</table>
The CSS Box Model

Every element is composed of 4 layers:

- the element's content
- the **border** around the element's content
- **padding** space between the content and border (inside)
- a **margin** clears the area around border (outside)
<div>s look a little squished

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

Q: How do we add space between multiple elements?
margin is the space between the border and other elements.
- Can specify margin-top, margin-bottom, margin-left, margin-right
- There's also a shorthand:
  margin: 2px 4px 3px 1px; <- top|right|bottom|left
  margin: 10px 2px; <- top+bottom|left+right
Back where we left off!
Actually, why doesn't this:

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

Look more like this?

- Lectures
- Homework

- Lectures
- Homework
margin

Actually, why doesn't this:

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

...look more like this?

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

Lectures

Homework

Lectures

Homework
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)

```css
img { margin-top: -50px; }
```
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?
Q: Does the box model apply to inline elements as well?
A: Yes, but the box is **shaped differently**.
Box model for inline elements?

Q: Does the box model apply to inline elements as well?
A: Yes, but the box is shaped differently.

Let's change the line height to view this more clearly...
Inline element box model

Welcome to CS193X: Web Programming Fundamentals! This class is in the Shriram Center for Bioengineering and Chemical Engineering. Hope you enjoy the class!

(Codepen)
- **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
Let's revisit our Course web page example:

**CS 193X: Web Fun**

**Announcements**
- 4/3: Homework 0 is out! **Due Friday.**
- 4/3: Office hours are now posted.

[View Syllabus]
Q: What does this look like in the browser?

```html
<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.

```css
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:

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

Begin visualizing the layout in terms of boxes:
Basic shape

Begin visualizing the layout in terms of boxes:
Basic shape

Begin visualizing the layout in terms of boxes:

Let's first try making this layout!

Codepen Link
## Content Sectioning elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;p&gt;</code></td>
<td>Paragraph (<a href="https://developer.mozilla.org">mdn</a>)</td>
</tr>
<tr>
<td><code>&lt;h1&gt;</code>-<code>&lt;h6&gt;</code></td>
<td>Section headings (<a href="https://developer.mozilla.org">mdn</a>)</td>
</tr>
<tr>
<td><code>&lt;article&gt;</code></td>
<td>A document, page, or site (<a href="https://developer.mozilla.org">mdn</a>)</td>
</tr>
<tr>
<td></td>
<td>This is usually a root container element after body.</td>
</tr>
<tr>
<td><code>&lt;section&gt;</code></td>
<td>Generic section of a document (<a href="https://developer.mozilla.org">mdn</a>)</td>
</tr>
<tr>
<td><code>&lt;header&gt;</code></td>
<td>Introductory section of a document (<a href="https://developer.mozilla.org">mdn</a>)</td>
</tr>
<tr>
<td><code>&lt;footer&gt;</code></td>
<td>Footer at end of a document or section (<a href="https://developer.mozilla.org">mdn</a>)</td>
</tr>
<tr>
<td><code>&lt;nav&gt;</code></td>
<td>Navigational section (<a href="https://developer.mozilla.org">mdn</a>)</td>
</tr>
</tbody>
</table>

These elements do not "do" anything; they are basically more descriptive `<div>`s. Makes your HTML more readable. See [MDN](https://developer.mozilla.org) for more info.
Content Sectioning elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;p&gt;</code></td>
<td>Paragraph (<a href="https://developer.mozilla.org">mdn</a>)</td>
</tr>
<tr>
<td><code>&lt;h1&gt;</code>–<code>&lt;h6&gt;</code></td>
<td>Section headings (<a href="https://developer.mozilla.org">mdn</a>)</td>
</tr>
<tr>
<td><code>&lt;article&gt;</code></td>
<td>A document, page, or site (<a href="https://developer.mozilla.org">mdn</a>) This is usually a root container element after <code>&lt;body&gt;</code></td>
</tr>
<tr>
<td><code>&lt;section&gt;</code></td>
<td>Generic section of a document (<a href="https://developer.mozilla.org">mdn</a>)</td>
</tr>
<tr>
<td><code>&lt;header&gt;</code></td>
<td>Introductory section (<a href="https://developer.mozilla.org">mdn</a>)</td>
</tr>
<tr>
<td><code>&lt;footer&gt;</code></td>
<td>Footer at end of a document (<a href="https://developer.mozilla.org">mdn</a>)</td>
</tr>
<tr>
<td><code>&lt;nav&gt;</code></td>
<td>Navigational section (<a href="https://developer.mozilla.org">mdn</a>)</td>
</tr>
</tbody>
</table>

These elements do not "do" anything, they are basically more descriptive `<div>`s. Makes your HTML more readable. See [MDN](https://developer.mozilla.org) for more info.

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

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

Header:
- Height: 400px;
- Background: lightskyblue
- <header>
Main section

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

Footer:
- Height: 100px
- Background: Black
- <footer>
Main contents

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

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

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

This is where we get stuck.
Flexbox
CSS layout so far

Block layout:
Laying out large sections of a page

Inline layout:
Laying out text and other inline content within a section
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 solves all sorts of problems.

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

  - Split-screen
  - Sidebar
  - Sticky footer
  - Centering
  - Fluid grid
  - Collection grid
  - Equal-height modules
Flex layout solves all sorts of layout problems.
- Here are some examples of layouts that are easy to create with flex layout (and really difficult otherwise):

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.

```
id=flex-container

class=flex-item
```
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](https://codepen.io)
```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>
```
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:

```css
#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:

```css
#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:

```css
#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:

```css
#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:

```css
#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:

```css
#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:

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

Same rules apply with multiple flex items:

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

Same rules apply with multiple flex items:

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

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

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

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

```css
#flex-container {
    display: flex;
    justify-content: space-around;
    align-items: center;
}
```
And you can also lay out columns instead of rows:

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

And you can also lay out columns instead of rows:

```css
#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:

```css
#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:

```css
#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:

```css
#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>
</html>
```

```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>
<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>
    <span class="flex-item"></span>
  </div>
</body>
</html>
```
Recall: block layouts

If #flex-container was not display: flex:

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

Why does this change when display: flex?
Why do inline elements suddenly seem to have height and width?
More next time!