CS193X: Web Programming Fundamentals

Spring 2017

Victoria Kirst
(vrk@stanford.edu)
Today's schedule

Announcements:
- Office Hours now posted

Schedule:
- HTML and CSS
- Inline vs block
- Classes and Ids
HW0 Reminders

HW0 still due this Friday!

A few tips:

- Please don't make your repository public
  - If you do, I will just make it private again

- Don't forget to submit your homework via the Google Form linked at the bottom of the HW0 spec

- You can update your HW0 GitHub repository/published page without submitting the Google Form again 😊 ☐ (but multiple submissions are OK)
Waitlist??

- If you have an access code and have not enrolled: Please do so ASAP

- If you do not have an access code yet: Please email me!
Suggestion: Bring your laptop!

- Bring your laptop to lecture so you can follow along with the lecture slides and check out the live examples.
- I will be using CodePen in lecture, which lets you livestream the code I write, which might be hard to see on the projector screen.

(But, y'know, don't look ahead for the answers to lecture questions and then pretend like you knew them all along.)
HTML and CSS
Quick review
Recall: HTML

**HTML (Hypertext Markup Language)**
- Describes the **content** and **structure** of a web page; not a programming language.
- Made up of building blocks called **elements**.

```html
<p>
  HTML is <em>awesome!!!</em>
  <img src="puppy.png" />
</p>
```
Some HTML elements

Top-level heading: `h1`, `h2`, ... `h6`

```
<h1>Moby Dick</h1>
<h2>Or, the Whale</h2>
```

Paragraph: `p`

```
<p>Call me Ishmael.</p>
```

Line break: `br`

```
since feeling is first
who pays any attention
to the syntax of things
```

Call me Ishmael.

since feeling is first
who pays any attention
to the syntax of things
Some HTML elements

Image: `<img src="pusheen.gif" />

Link: `<a href="google.com">click here!</a>`

Strong (bold): `<strong>Be BOLD</strong>`

Emphasis (italic): `<em>He's my brother</em> and all`
Recall: Course web page

We wrote some HTML to make the following page:

CS 193X: Web Fun

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

View Syllabus
That was weird

- We saw that HTML whitespace collapses into one space...

```html
<h1>CS 193X: Web Fun</h1>
<strong>Announcements</strong>
<br/>
4/3: Homework 0 is out!
<br/>
```

- Except weirdly the `<h1>` heading was on a line of its own, and `<strong>` was not.

Hmmm… strange…
Oh well, it works! Let's move on!!!
CSS
Recall: CSS

**CSS:** Cascading Style Sheets
- Describes the **appearance** and **layout** of a web page
- Composed of CSS **rules**, which define sets of styles

```css
selector {  
  property: value;
}
```
Some CSS properties

Font face: `font-family`

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

Font color: `color`

```
body {
  color: green;
}
```

Note that `color` always refers to `font` color, and there's no way to make it mean anything other than font color.

Background color: `background-color`

```
body {
  background-color: pink;
}
```

Assign a `background-color` to `body` to make the page a different color.
Some CSS properties

**Border:** `border` ([border shorthand syntax](https://developer.mozilla.org/en-US/docs/Web/CSS/Border))

```css
img {
  border: 3px solid red;
}
```

**Text alignment:** `text-align` *(note: don't use `<center>`)*

```css
p {
  text-align: center;
}
```

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

140 predefined names (list)
color: black;

Hex values
color: #00ff00;
color: #0f0;
color: #00ff0080;

rgb() and rgba()
color: rgb(34, 12, 64);
color: rgba(0, 0, 0, 0.5);

- The "a" in rgba stands for alpha channel and is a transparency value
- Prefer more descriptive:
  1. Predefined name
  2. rgb / rgba
  3. Hex
Exercise: Course web page

Let's write some CSS to style our page:

CS 193X: Web Fun

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

View Syllabus

CodePen link: Follow along!
Let's write some CSS to style our page:

**Font face**: Helvetica

**Border**: hotpink 3px

**Background color**: lavenderblush

**Highlight**: yellow

- Box is centered
- Header and link are centered
- Box contents are left-aligned

**CodePen link**: Follow along!
Solution?!

body {  
  font-family: Helvetica;
}

h1 {  
  text-align: center;
}
a {  
  text-align: center;
}
p {  
  border: 3px solid hotpink;
  background-color: lavenderblush;
}
CSS exercise debrief

We used some **key techniques:**

- Add invisible containers in HTML to select groups of elements in CSS.
- Apply styles to parent/ancestor element to style parent and all its children. (Will talk more about this later.)
CSS exercise debrief

But we encountered more weirdness...
- text-align: center; didn't work on the `<a>` tag
- The box was reaaaaaally wide!
- How to center the box?!
- How do you highlight?!
Q: Why is HTML/CSS so bizarre??
A: There is one crucial set of rules we haven't learned yet...

**block vs inline display**
What is HTML?

HTML (Hypertext Markup Language)
- Describes the **content** and **structure** of a web page
- Made up of building blocks called **elements**.

```html
<p>
    HTML is <em>awesome!!!</em>
    <img src="puppy.png" />
</p>
```

And there are 3 basic types.
Types of HTML elements

Each HTML element is categorized by the HTML spec into one of three-ish categories:

1. **block**: large blocks of content, has height and width
   - `<p>`, `<h1>`, `<blockquote>`, `<ol>`, `<ul>`, `<table>`

2. **inline**: small amount of content, no height or width
   - `<a>`, `<em>`, `<strong>`, `<br>`
     a. **inline block**: inline content with height and width
        - `<img>`

3. **metadata**: information about the page, usually not visible
   - `<title>`, `<meta>`
Block elements

Examples:
<p>, <h1>, <blockquote>, <ol>, <ul>, <table>

- Take up the full width of the page (flows top to bottom)
- Have a height and width
- Can have block or inline elements as children
About vrk

She likes puppies
Q: What does this look like in the browser?

```html
h1 {
    border: 5px solid red;
}

<h1>About vrk</h1>
<p>
    She likes <em>puppies</em>
</p>
```
About vrk

She likes *puppies*
Block-level: extends the full width of the page

```html
h1 {
  border: 5px solid red;
}
```

```html
<h1>About vrk</h1>
<p>
  She likes <em>puppies</em>
</p>
```

<h1>is block-level, so it extends the full width of the page by default</h1>

Note how block-level elements (<h1>, p) flow top to bottom

See: Codepen
Q: What does this look like in the browser?

```css
h1 {
    border: 5px solid red;
    width: 50%;
}
```

```
<h1>About vrk</h1>
<p>She likes <em>puppies</em></p>
```

About vrk

She likes *puppies*
About vrk

She likes puppies
Block-level width can be modified

```css
h1 {
  border: 5px solid red;
  width: 50%;
}
```

```
<h1>About vrk</h1>
<p>
  She likes <em>puppies</em>
</p>
```

<h1>is block-level, so its <strong>width</strong> can be modified</h1>

Block-level elements still flow top to bottom

See: <a>Codepen</a>
Inline elements

Examples:
<a>, <em>, <strong>, <br>

- Take up only as much width as needed (flows left to right)
- **Cannot** have height and width
- **Cannot** have a block element child
- **Cannot** be positioned (i.e. CSS properties like float and position do not apply to inline elements)
  - Must position its **containing block element** instead
Web programming resources: CS 193X MDN Google

Web programming resources: CS 193X MDN Google
Inline elements ignore width
width cannot be modified

```css
strong {
  border: 5px solid red;
  width: 1000px;
  /* Will not work; strong is inline! */
}
```

web programming resources:
<a href="http://cs193x.stanford.edu">CS 193X</a>
<a href="http://google.com">Google</a>

Cannot set width on inline element, so it is ignored (Codepen)
inline-block

Examples: `<img>`, any element with `display: inline-block;`

- Width is the size of the content, i.e. it takes only as much space as needed (flows left to right)
- **Can** have height and width
- **Can** have a block element as a child
- **Can** be positioned (i.e. CSS properties like float and position apply)
Example: Inline-block

```
img {
    width: 50px;
}
```

Q: What does this look like in the browser?

```
<img src="http://i.imgur.com/WJToVGv.jpg" />
<img src="http://i.imgur.com/WJToVGv.jpg" />
<img src="http://i.imgur.com/WJToVGv.jpg" />
<img src="http://i.imgur.com/WJToVGv.jpg" />
<img src="http://i.imgur.com/WJToVGv.jpg" />
```

http://i.imgur.com/WJToVGv.jpg =
**Inline-block**
Has width and height; flows left to right

Can set `width` on inline-block element, so image width is set to 50px. ([Codepen](http://i.imgur.com/WJToVGv.jpg)

`inline-block` flows left to right, so images are right next to each other.

```
img {
  width: 50px;
}
```

```
<img src="http://i.imgur.com/WJToVGv.jpg" />
<img src="http://i.imgur.com/WJToVGv.jpg" />
<img src="http://i.imgur.com/WJToVGv.jpg" />
<img src="http://i.imgur.com/WJToVGv.jpg" />
<img src="http://i.imgur.com/WJToVGv.jpg" />
```
The **display** CSS property

You can change an element's default rendering type by changing the **display** property. Examples:

```css
p {
    display: inline;
}
```

```css
a {
    display: block;
}
```

Possible values for display:

- block
- inline
- inline-block
- some others: link
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>`

Questions?
Moral of the story:
If your CSS isn't working, see if you're trying to apply block-level properties to inline elements.
Recall: Weirdly the `<h1>` heading was on a line of its own, and `<strong>` was not. -- **Why?**
h1 vs strong demystified!

Recall: Weirdly the <h1> heading was on a line of its own, and <strong> was not. -- Why?

Because h1 is a block-level element, and strong is an inline-level element.
text-align mystery

Recall: We couldn't set `text-align: center;` on the `<a>` tag directly, but we could center `<h1>`. Why?

```html
h1 { /* works! */
  text-align: center;
}

a { /* fails :(
  text-align: center;
}
```

CS 193X: Web Fun

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

[View Syllabus](#)

Let's try looking at the [MDN description of text-align](#) ...
**Summary**

The **text-align** CSS property describes how inline content like text is aligned in its parent block element. **text-align** does not control the alignment of block elements, only their inline content.

<table>
<thead>
<tr>
<th>Initial value</th>
<th>start, or a nameless value that acts as <code>left</code> if <code>direction</code> is <code>ltr</code>, <code>right</code> if <code>direction</code> is <code>rtl</code> if <code>start</code> is not supported by the browser.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applies to</td>
<td><strong>block containers</strong></td>
</tr>
</tbody>
</table>
text-align demystified!

**Why?** From the *spec*, *can't apply text-align to an inline element*; must apply text-align to its block container, or set `a { display: block; }`
Box size mystery

Recall: The pink box we put around the announcements extended the entirety of the page.

Why?
How do we fix this?
Box size mystery

Recall: The pink box we put around the announcements extended the entirety of the page.

Why? Because p is block-level, so width == width of the page

How do we fix this?
Box size mystery: demystified!

Recall: The pink box we put around the announcements extended the entirety of the page.

```
p {
    border: 3px solid hotpink;
    background-color: lavenderblush;
    display: inline-block;
}
```

Why? Because p is block-level, so width == width of the page

How do we fix this? Change display to inline-block (though now the space above the box has increased… will address later!)
We can also center the box by centering the body tag, since p is now inline-block.

```css
body {
    text-align: center;
}

p {
    border: 3px solid hotpink;
    background-color: lavenderblush;
    display: inline-block;
}
```
Highlight mystery

Recall: We didn't know how to select a random snippet of text to change its background.

How do we fix this?
Highlight: demystified!

We can select a random segment of text by wrapping it in an inline element:

```html
<br/>
4/3: Homework 0 is out!
<br/>
4/3: Office hours are now posted.
```

```css
em {
  font-style: normal; /* undoes italics */
  background-color: yellow;
}
```

Hmmm… but wouldn't it be better to have a "highlight" element?
Highlight: demystified!

We can select a random segment of text by wrapping it in an inline element:

```
<strong>Announcements</strong><br/>
4/3: Homework 0 is out!<br/>
<em>Due Friday.</em><br/>
4/3: Office hours are now posted.
```

```
em {
    font-style: normal; /* undoes italics */
    background-color: yellow;
}
```

Hmmm… but wouldn't it be better to have a "highlight" element? How do we make a generic HTML element?
Have you heard of `<div>` and `<span>`? What are they?
<div> and <span>  
Two generic tags with no intended purpose or style:  
- `<div>`: a generic **block** element  
- `<span>`: a generic **inline** element
We can use `<span>` as a generic inline HTML container:
Multiple generic containers?

But won't we often want multiple generic containers?

How do we distinguish two generic containers?

In other words, how do we select a subset of elements instead of all elements on the page?
CSS Selectors: Classes and Ids
Classes and ids

There are 3 basic types of CSS selectors:

<table>
<thead>
<tr>
<th>Element selector (this is the one we've been using)</th>
<th>p</th>
<th>All &lt;p&gt; elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>✨ ID selector ✨</td>
<td>#abc</td>
<td>element with id=&quot;abc&quot;</td>
</tr>
<tr>
<td>✨ Class selector ✨</td>
<td>.abc</td>
<td>elements with class=&quot;abc&quot;</td>
</tr>
</tbody>
</table>

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

<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>

.homework {color: hotpink;}
#title {color: purple;}

Homework

HW0 is due Friday.
HW1 goes out Monday.
All homework due at 11:59pm.
More on class and id

- **class** and **id** are special HTML attributes that can be used on any HTML element
  - **class**: Used on 1 or more elements; identifies a **collection** of elements
  - **id**: Used on exactly 1 element per page; identifies **one unique** element
- Can apply multiple classes by space-separating them: `<span class="hw new">HW1</span>`
- Often used with span and div to create generic elements: e.g. `<span class="highlight">is like creating a "highlight" element</span>`
Other selectors:
Next time!
Overflow slides
(we didn't cover these)
**element.className**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Example</th>
<th>Example described</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>element.className</code></td>
<td><code>p.abc</code></td>
<td><code>&lt;p&gt;</code> elements with abc class</td>
</tr>
</tbody>
</table>

**HTML**

```html
<h1 class="hw">Homework 0</h1>
<p class="hw">Due Fri</p>
<p class="hw">Late cutoff Sun</p>
<h1>Lectures</h1>
<p>Apr 3: Syllabus</p>
<p>Apr 5: HTML+CSS</p>
```

**CSS**

```css
.p.hw {
  color: green;
}
```

**Homework 0**

- Due Fri
- Late cutoff Sun

**Lectures**

- Apr 3: Syllabus
- Apr 5: HTML+CSS
## Descendent selector

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Example</th>
<th>Example described</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>selector selector</code></td>
<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>
</tbody>
</table>

### HTML

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

### CSS

```css
.hw p { 
    color: green;
}
```

### Homework 0

Due Fri

Late cutoff Sun

### Lectures

Apr 3: Syllabus

Apr 5: HTML+CSS
Descendent selector

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Example</th>
<th>Example described</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>selector selector</code></td>
<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>
</tbody>
</table>

**Note:** The element does not have to be a direct child. The descendant may be nested many layers in.
Descendent selector

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Example</th>
<th>Example described</th>
</tr>
</thead>
<tbody>
<tr>
<td>selector selector</td>
<td>div strong</td>
<td><code>&lt;strong&gt;</code> elements that are descendants of a <code>&lt;div&gt;</code></td>
</tr>
</tbody>
</table>

**Discouraged:**

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

**Preferred:**

```
<div class="hw">
  <h1>Homework 0</h1>
  <p>Due Fri</p>
  <p>Due Fri</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)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Example</th>
<th>Example described</th>
</tr>
</thead>
<tbody>
<tr>
<td>selector, selector</td>
<td>h2, div</td>
<td>&lt;h2&gt; elements and &lt;div&gt;s</td>
</tr>
</tbody>
</table>

**HTML**

```html
<h1>Course Info</h1>
<h2>Lectures</h2>
<p>Mon-Wed-Fri 1:30-2:20</p>
<h2>Honor code</h2>
<p>Do the right thing</p>
```

**CSS**

```css
h1, h2 {
  font-family: Arial;
}
```

**Course Info**

**Lectures**

Mon-Wed-Fri 1:30-2:20

**Honor code**

Do the right thing
## Selector summary

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

```html
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*)

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

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

Specificity precedence rules (details):
● ids are more specific than classes
● classes are more specific than element names
● elements are more specific than children of those elements
Colliding styles

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

```html
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:

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

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

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

You can override this style via specificity:

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

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

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

<em>
    inherits the font-family property, but not display:
</em>

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

<em>Back to Home</em>
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)
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 ids when you have classes?
- 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)

You should mostly consider the box model properties for **block-level** elements!

- It can be used on inline elements but it **behaves differently**
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;`
Can also specify each border individually:

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

And can set each property individually:

- `border-style: dotted;`  
  `(all styles)`
- `border-width: 3px;`
- `border-color: purple;`
Can also specify each border individually:

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

And can set each property individually:

- `border-style: dotted;` *(all styles)*
- `border-width: 3px;`
- `border-color: purple;`

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:

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

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

```css
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 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
  ```
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**

- Can specify margin-top, margin-bottom, margin-left, margin-right
- There's also a **shorthand**:
  
  ```
  div {
    border: 2px solid black;
    margin: 10px;
    padding: 10px;
  }
  ```

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

**margin** is the space between the border and other elements.
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]
The CSS Box Model

- width by default refers to content-width, which is almost never what you want
  - Use box-sizing: border-box; to calculate width based on the border size
- Margin collapsing is something to know about and consider
- When in doubt, use the browser's Page Inspector to see what's going on

(Also, the Box Model works a little differently for inline elements)
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>
```

```css
div {
  display: inline-block;
  background-color: yellow;
}
```
Q: Why is there a white space around the box?

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

Make the background color yellow!

Surrounding these paragraphs