

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**.

<p>

HTML is awesome!!!

</p>

Some HTML elements

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

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

Moby Dick

Or, the Whale

Paragraph: **p**

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

Call me Ishmael.

Line break: **br**

```
since feeling is first<br/>  
who pays any attention<br/>  
to the syntax of things
```

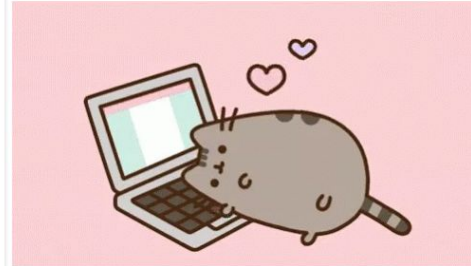
since feeling is first
who pays any attention
to the syntax of things

Some HTML elements

Image: **img**

```

```



Link: **a** (note: not **link**)

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

[click here!](https://www.google.com)

Strong (bold): **strong** (note: don't use **b**)

```
<strong>Be BOLD</strong>
```

Be BOLD

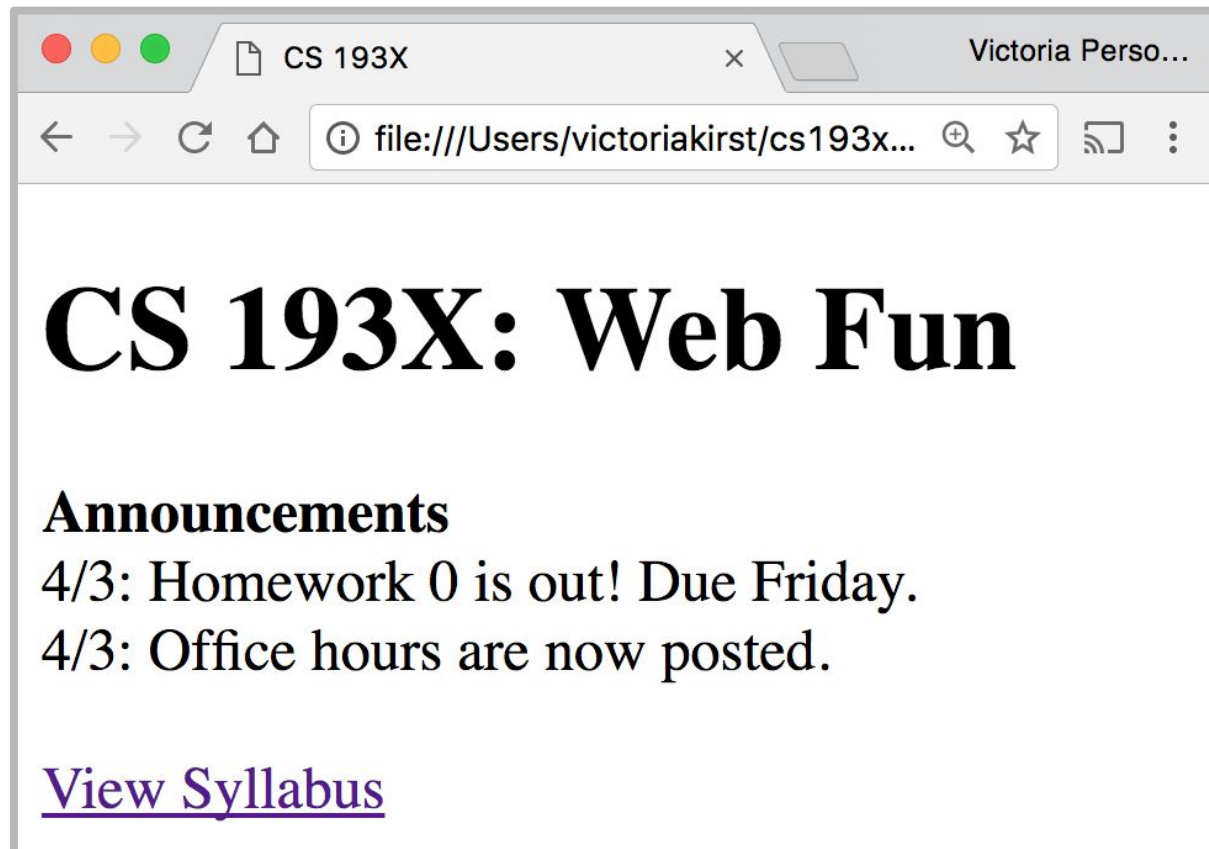
Emphasis (italic): **em** (note: don't use **i**)

```
He's my <em>brother</em> and all
```

He's my *brother* and all

Recall: Course web page

We wrote [some HTML](#) to make the following page:



That was weird

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

```
<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 `` 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

```
selector {  
    property: value;  
}
```

Some CSS properties

Font face: **font-family**

```
h1 {  
  font-family: Helvetica;  
}
```

Moby Dick

Font color: **color**

```
h1 {  
  color: green;  
}
```

Moby Dick

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;  
}
```

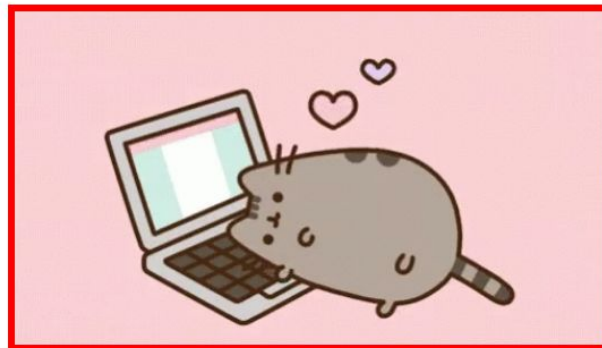
Moby Dick

Assign a **background-color** to **body** to make the page a different color.

Some CSS properties

Border: **border** ([border shorthand syntax](#))

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



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

```
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()

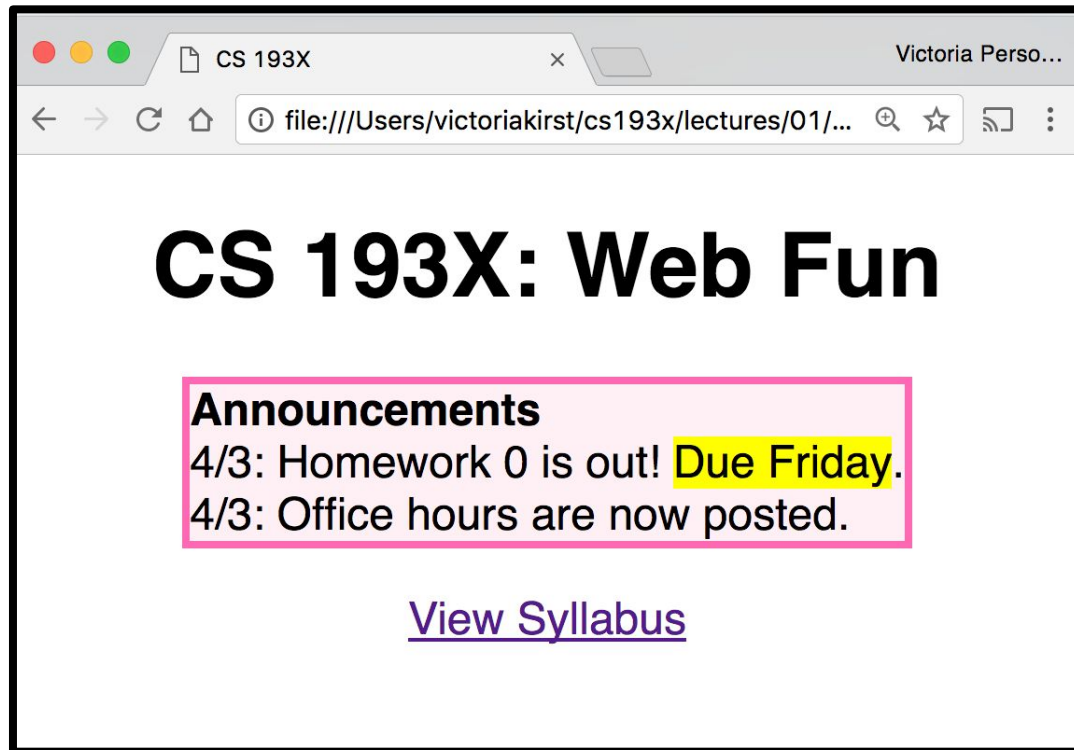
```
color: rgb(34, 12, 64);
```

```
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:



❖ [CodePen link: Follow along!](#) ❖

Exercise: Course web page

Let's write some CSS to style our page:

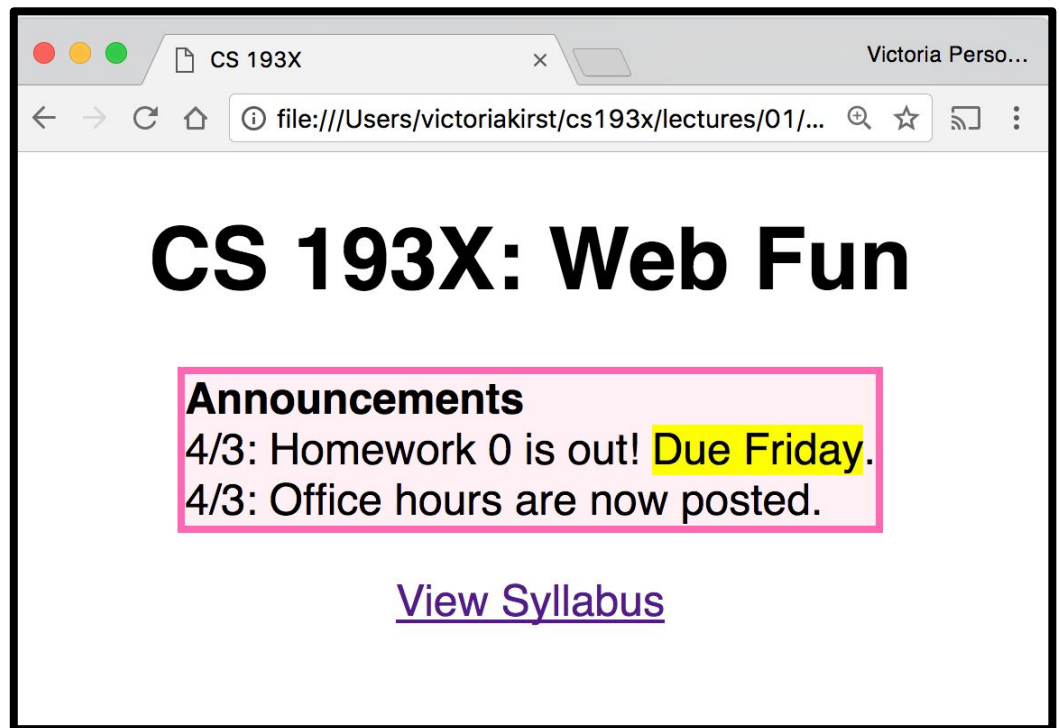
Font face: Helvetica

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;  
}
```

Produces:

CS 193X: Web Fun

Announcements

4/3: Homework 0 is out! Due Friday.

4/3: Office hours are now posted.

[View Syllabus](#)

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...**

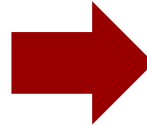
- `text-align: center;` didn't work on the `<a>` tag
- The box was reaaaaaally wide!
- How to center the box?!
- How do you highlight?!

How do we get from this...

CS 193X: Web Fun

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

[View Syllabus](#)



... to this?

CS 193X: Web Fun

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

[View Syllabus](#)

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**.

<p>

HTML is awesome!!!

</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:

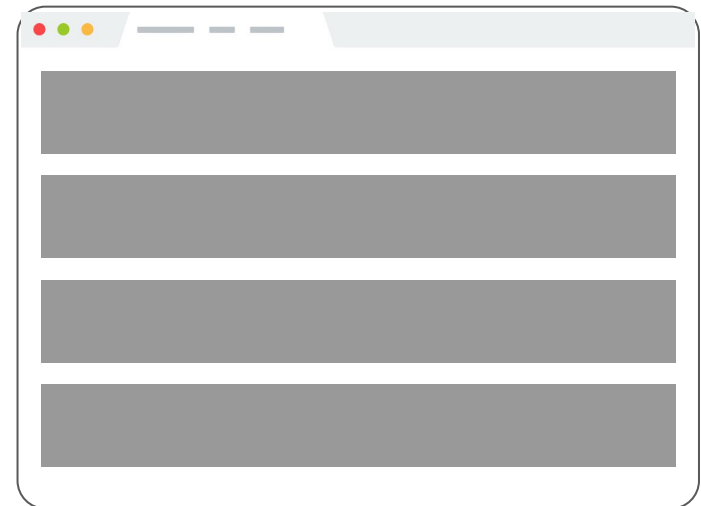
1. **block**: large blocks of content, has height and width
`<p>`, `<h1>`, `<blockquote>`, ``, ``, `<table>`
2. **inline**: small amount of content, no height or width
`<a>`, ``, ``, `
`
 - a. **inline block**: inline content with height and width
``
3. **metadata**: information about the page, usually not visible
`<title>`, `<meta>`

Block elements

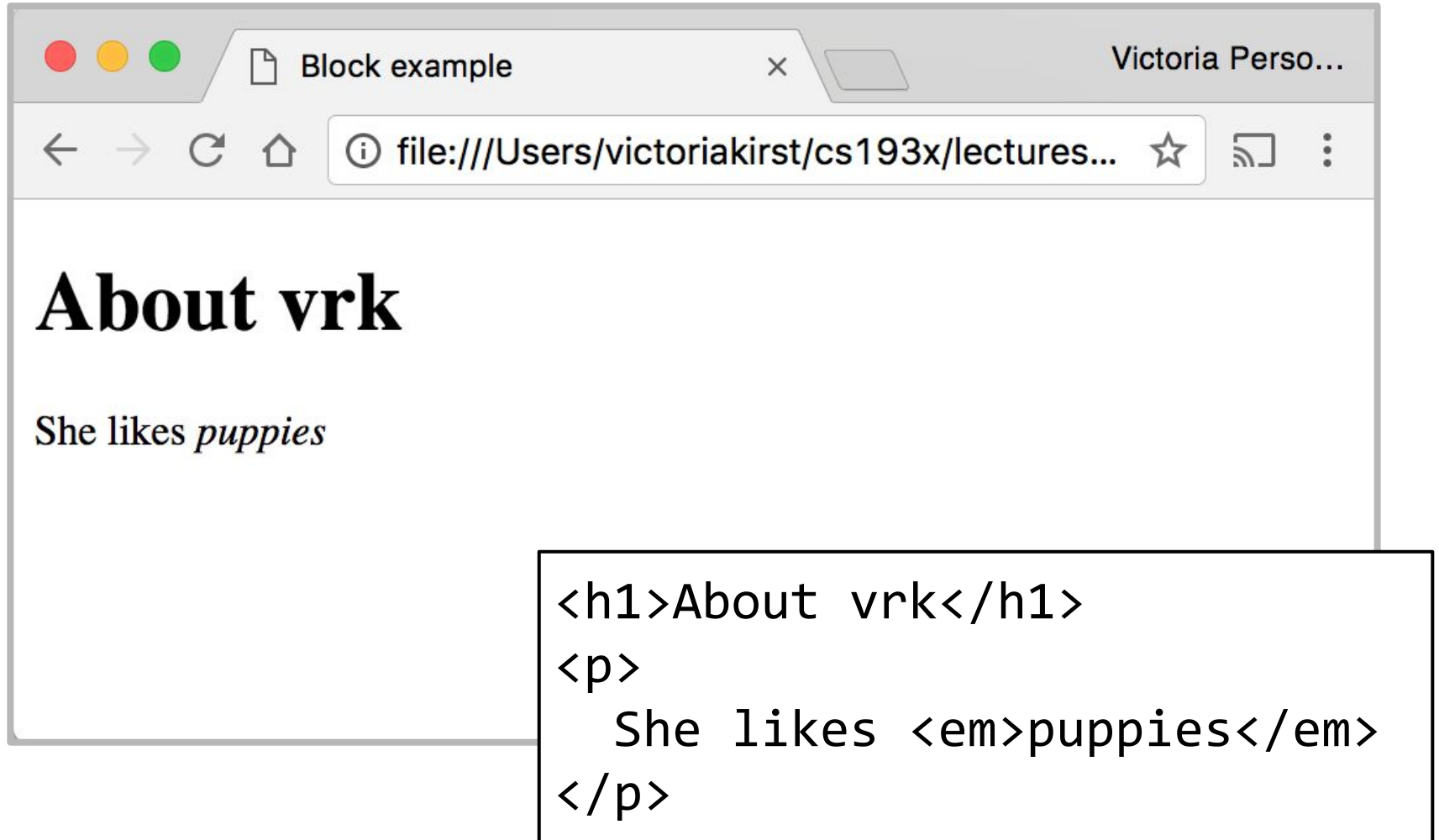
Examples:

`<p>`, `<h1>`, `<blockquote>`, ``, ``, `<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



Example: Block

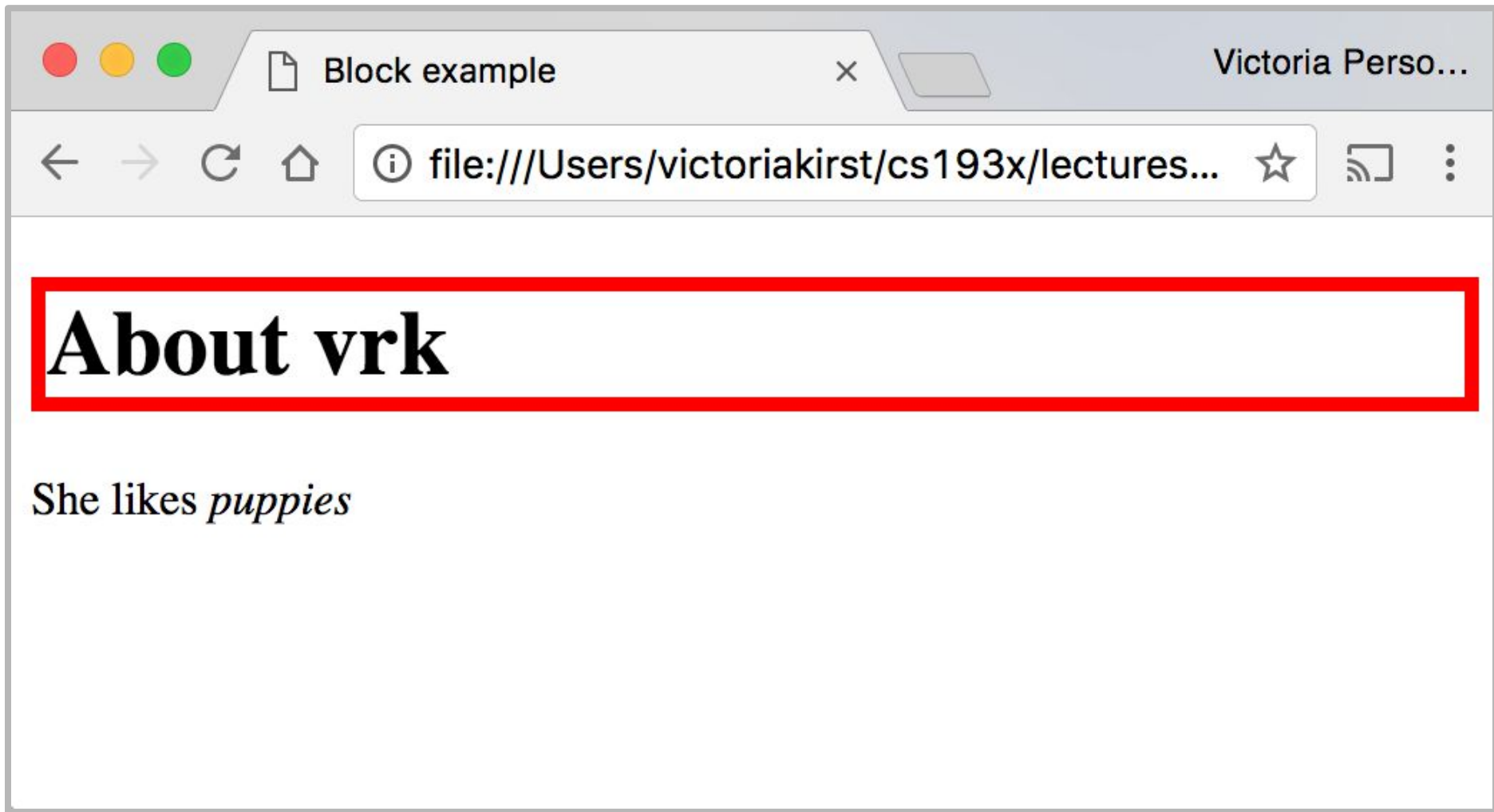


**Q: What does this
look like in the
browser?**

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



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



([Codepen](#))

Block-level:

extends the full width of the page

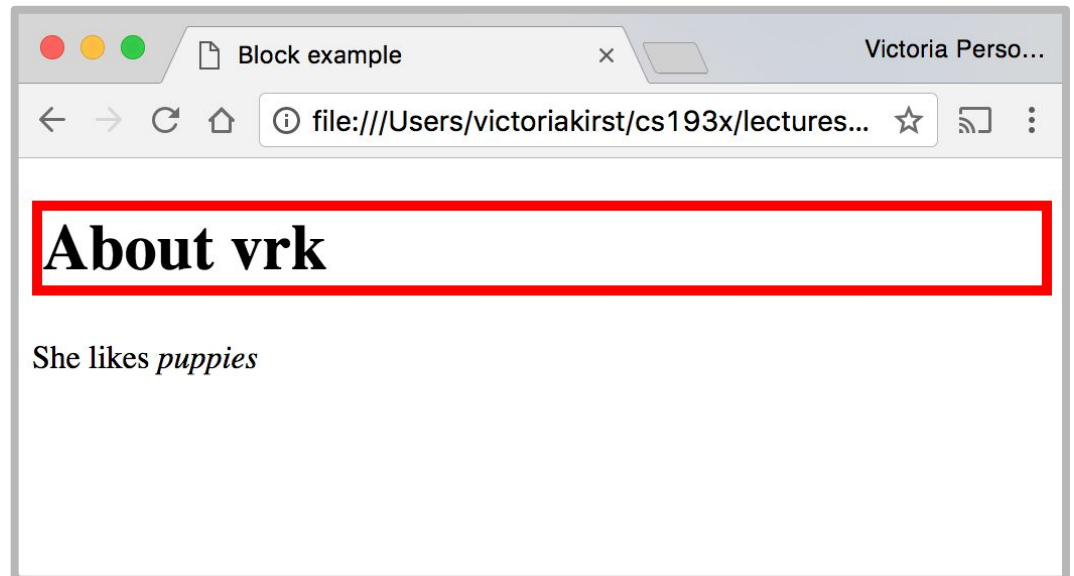
```
h1 {  
  border: 5px solid red;  
}
```

```
<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

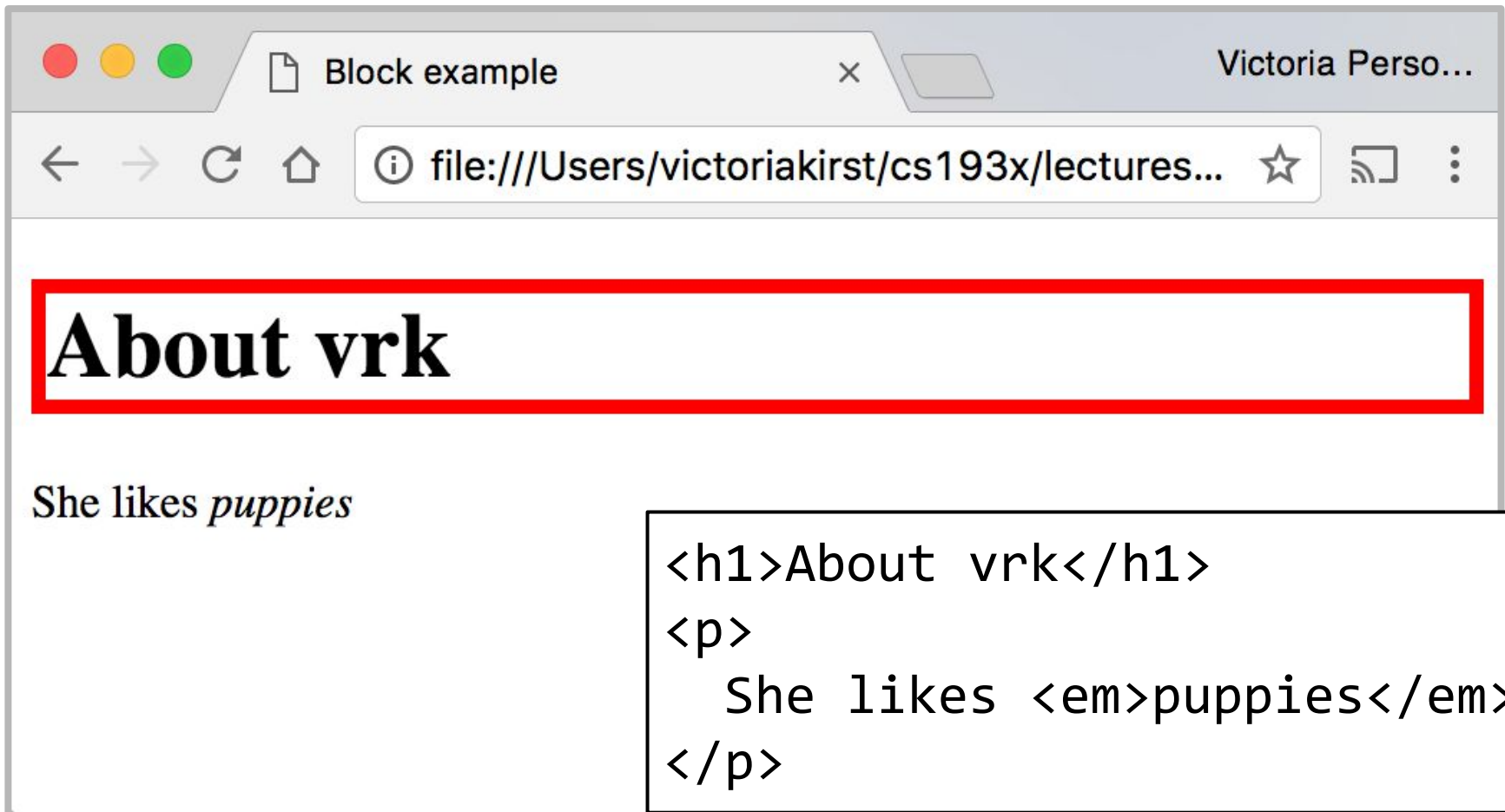
Note how block-level elements (**h1**, **p**) flow top to bottom

See: [Codepen](#)

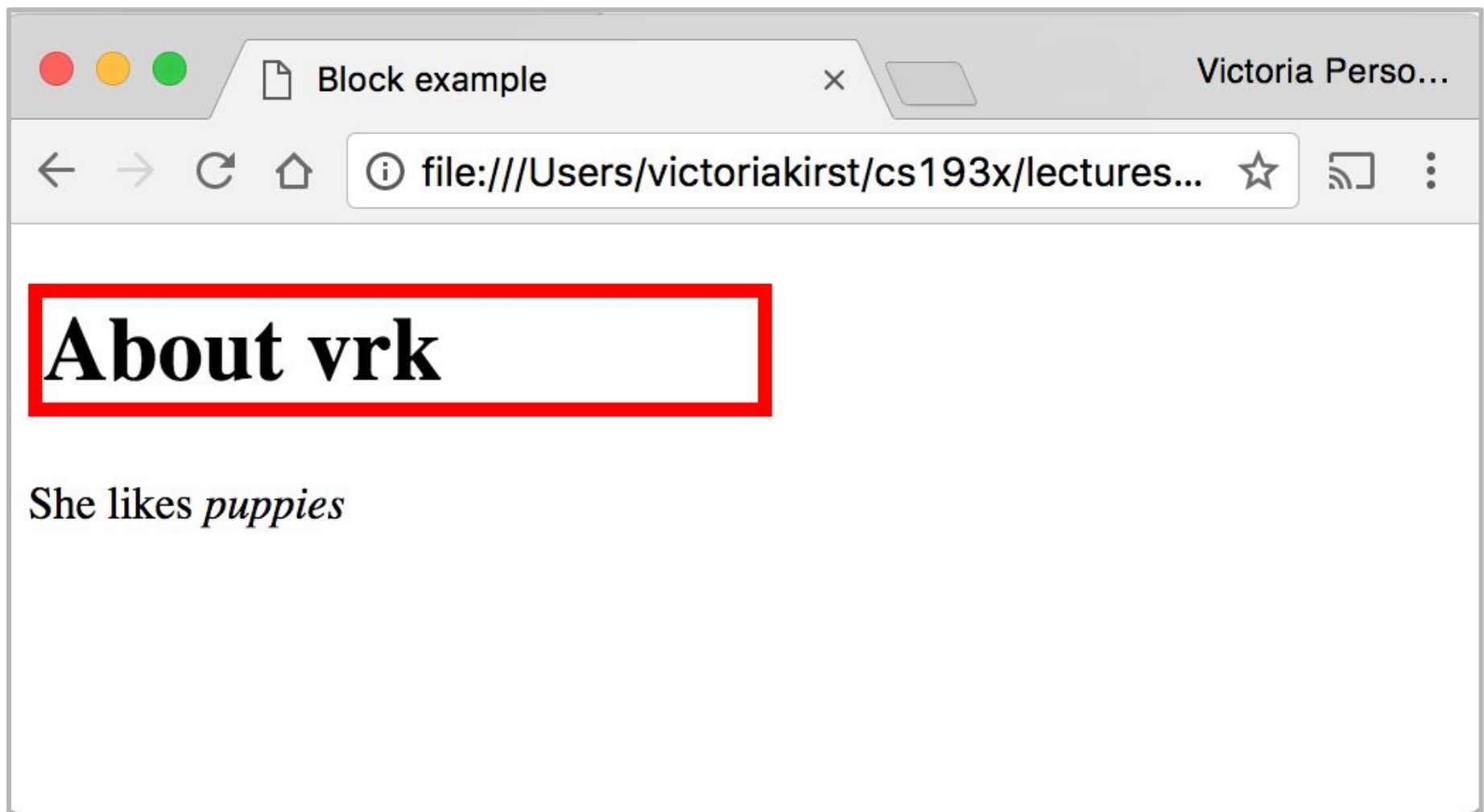


**Q: What does this
look like in the
browser?**

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



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

([Codepen](#))

Block-level

width can be modified

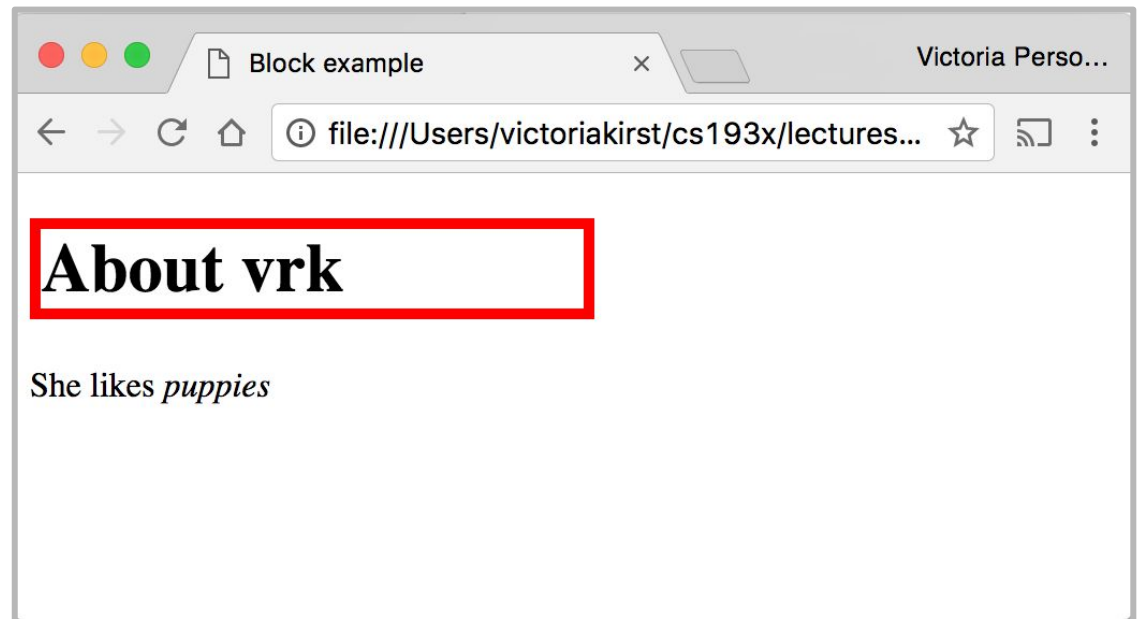
```
h1 {  
  border: 5px solid red;  
  width: 50%;  
}
```

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

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

Block-level elements
still flow top to
bottom

See: [Codepen](#)

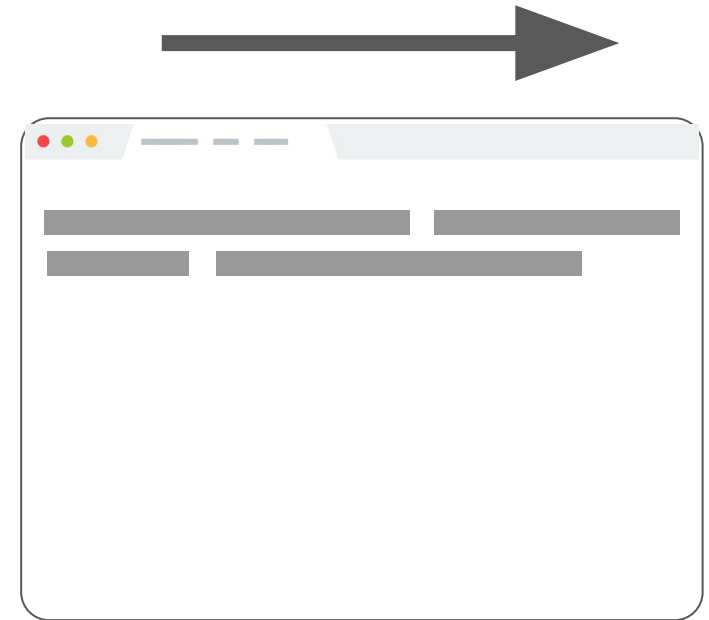


Inline elements

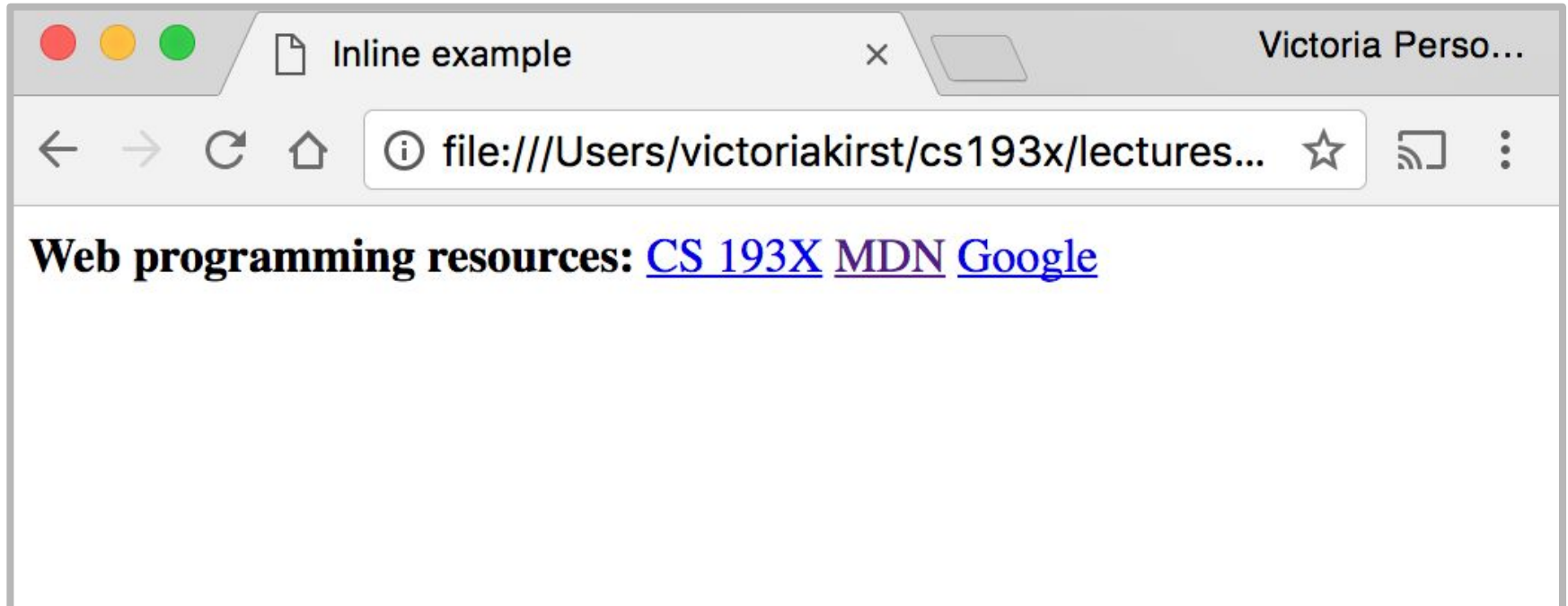
Examples:

`<a>`, ``, ``, `
`

- 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



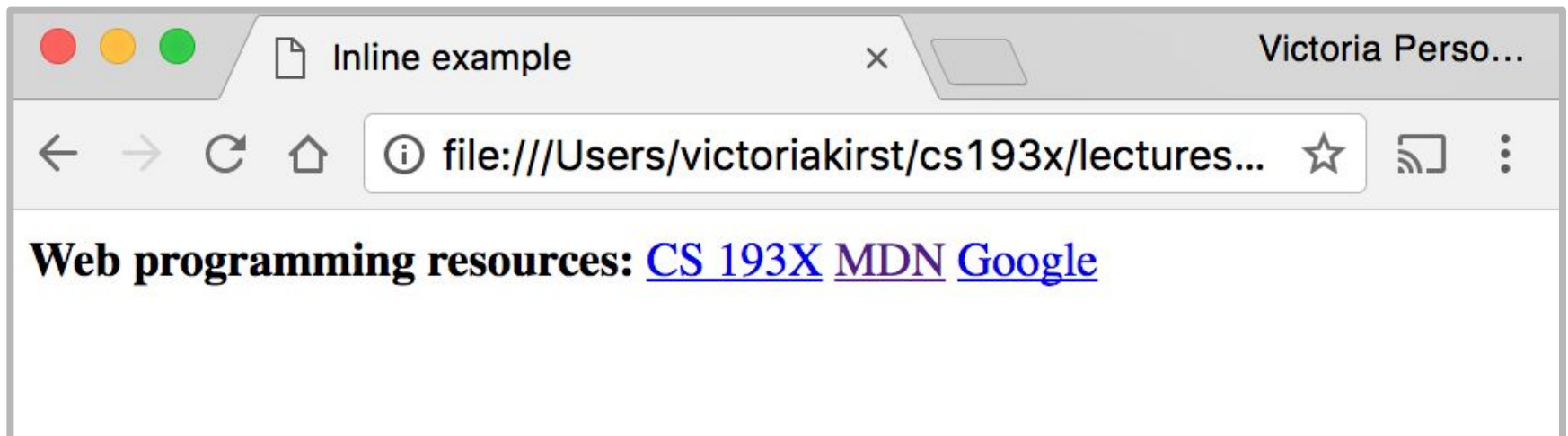
Example: Inline



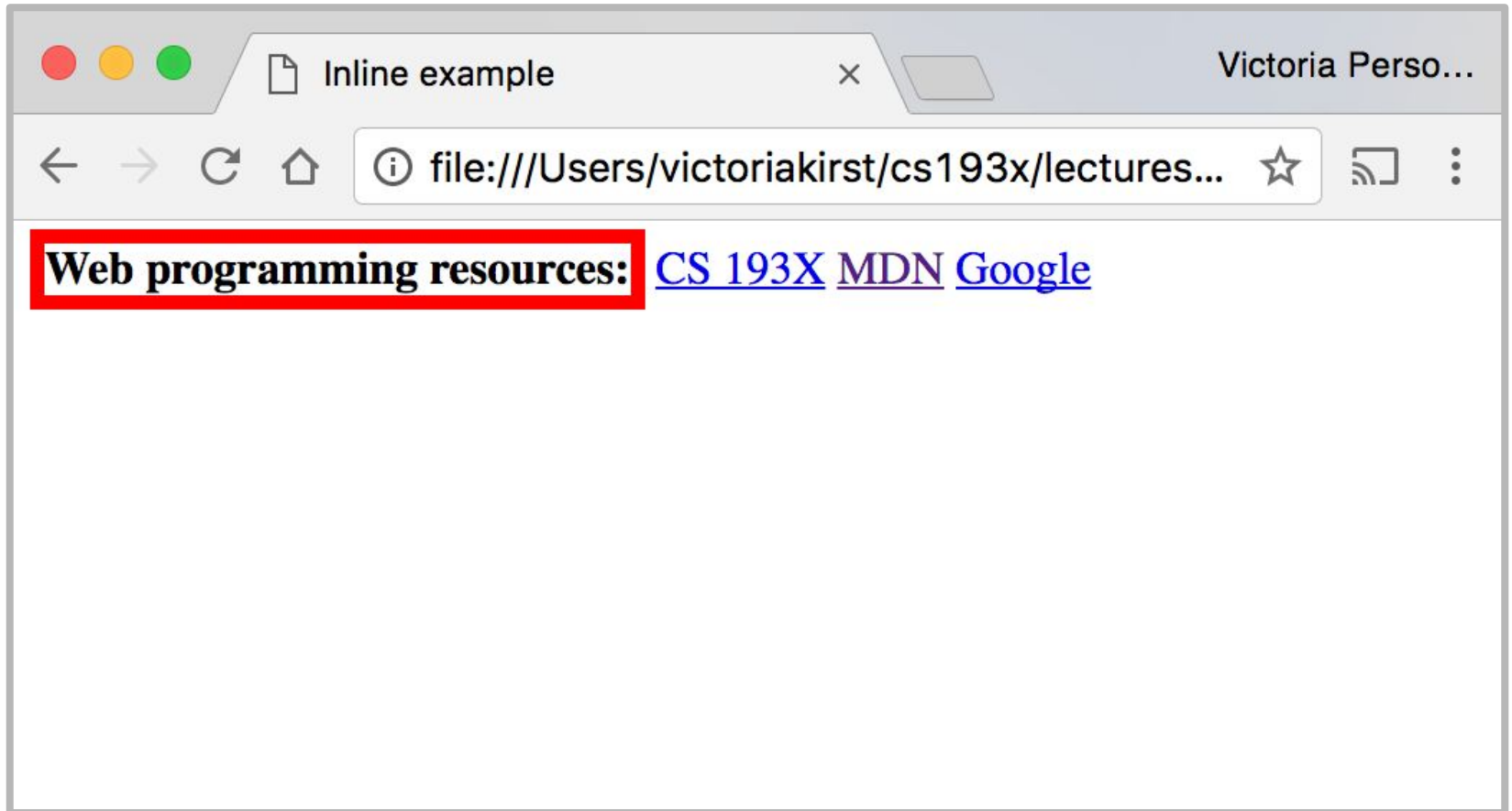
```
<strong>Web programming resources:</strong>  
<a href="http://cs193x.stanford.edu">CS 193X</a>  
<a href="https://developer.mozilla.org/en-US/">MDN</a>  
<a href="http://google.com">Google</a>
```

**Q: What does this
look like in the
browser?**

```
strong {  
  border: 5px solid red;  
  width: 1000px;  
}
```



```
<strong>Web programming resources:</strong>  
<a href="http://cs193x.stanford.edu">CS 193X</a>  
<a href="https://developer.mozilla.org/en-US/">MDN</a>  
<a href="http://google.com">Google</a>
```



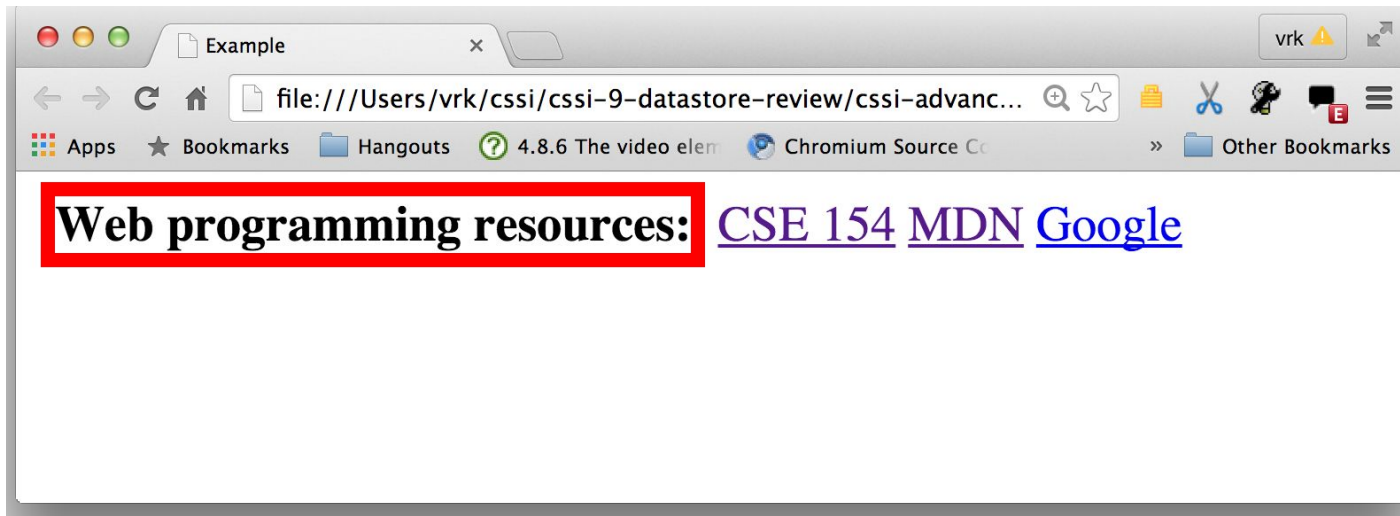
([Codepen](#))

Inline elements ignore width

width cannot be modified

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

```
<strong>Web programming reso  
<a href="http://cs193x.stanf  
<a href="https://developer.m  
<a href="http://google.com">
```

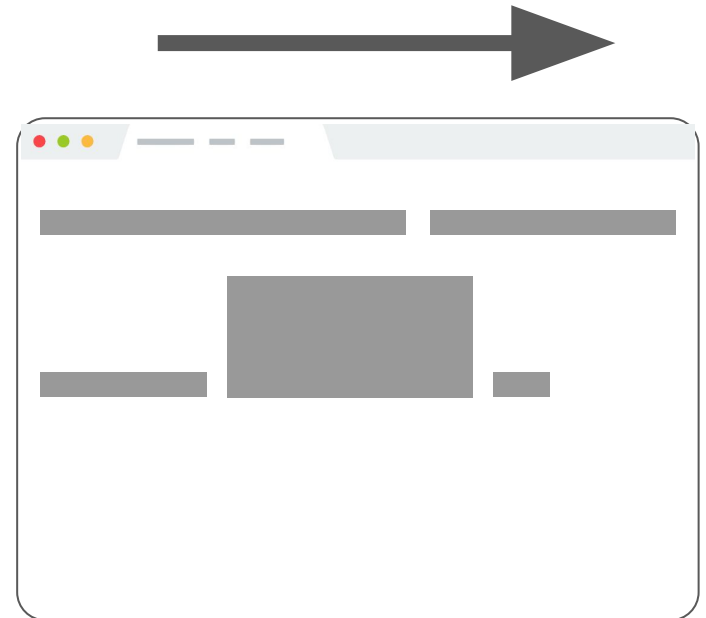


Cannot set **width** on inline element, so it is ignored ([Codepen](#))

inline-block

Examples: ``, 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;  
}
```

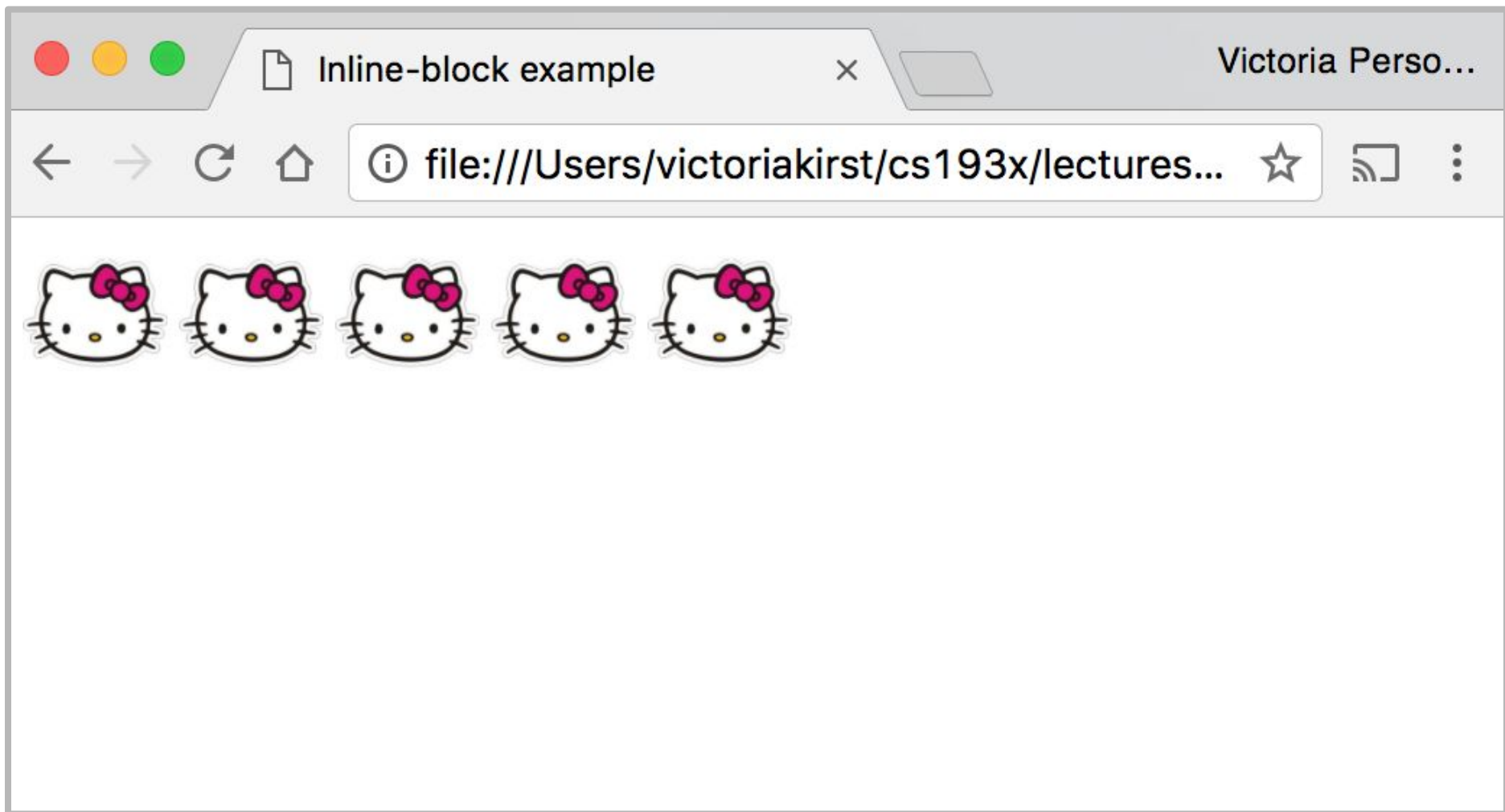
Q: What does this look like in the browser?

```
  
  
  
  

```

<http://i.imgur.com/WJToVGv.jpg> =





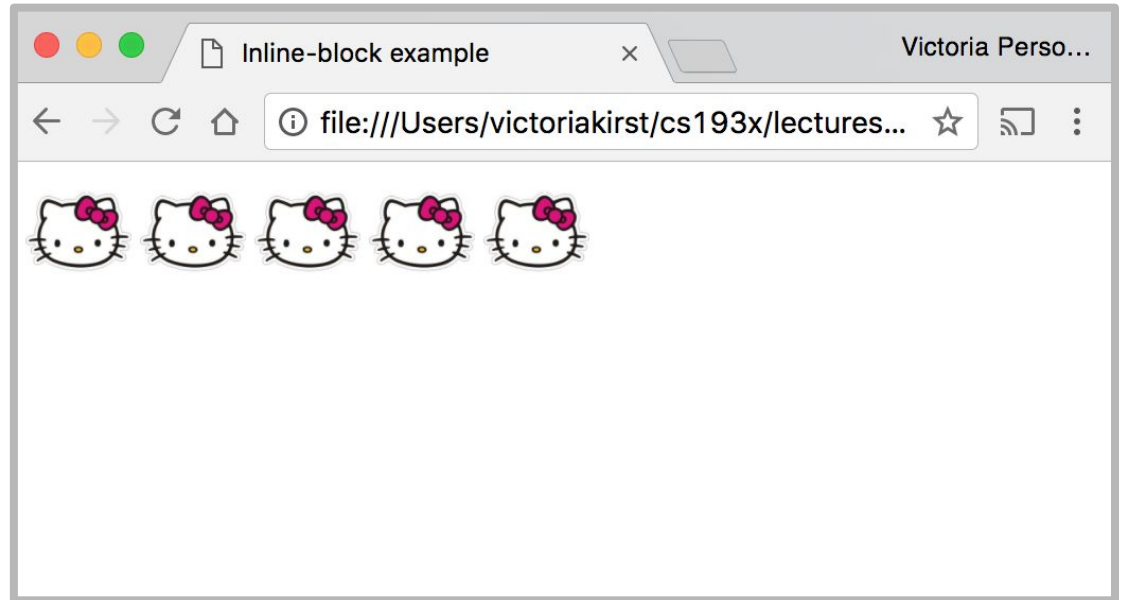
([Codepen](#))

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](#))

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



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

```
  
  
  
  

```

The display CSS property

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

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

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

Possible values for `display`:

- `block`
- `inline`
- `inline-block`
- some others: [link](#)

Review

1. **block**: flows **top-to-bottom**; **has** height and width
<p>, <h1>, <blockquote>, , , <table>
2. **inline**: flows **left-to-right**; **does not have** height and width
<a>, , ,

 - a. **inline block**: flows **left-to-right**; **has** height and width
equal to size of the content

Questions?

Moral of the story:

If your CSS isn't working, see if you're trying to apply block-level properties to inline elements

h1 vs strong mystery

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

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

CS 193X: Web Fun

Announcements 4/3: Homework 0 is out!

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

CS 193X: Web Fun

Announcements
4/3: Homework 0 is out!

h1 vs strong demystified!

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

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

CS 193X: Web Fun

Announcements 4/3: Homework 0 is out!

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

CS 193X: Web Fun

Announcements
4/3: Homework 0 is out!

**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?**

```
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](#)...

text-align mystery

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.

Initial value

start, or a nameless value that acts as left if **direction** is ltr, right if **direction** is rtl if start is not supported by the browser.

Applies to

block containers

([source](#))

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; }**

HTML

```
<p>
  <a href="http://cs193x.stanford.ed
    View Syllabus
  </a>
</p>
```

CSS

```
p { /* works! */
  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](#)

Box size mystery

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

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

CS 193X: Web Fun

Announcements

4/3: Homework 0 is out! Due Friday.

4/3: Office hours are now posted.

[View Syllabus](#)

Why?

How do we fix this?

Box size mystery

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

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

CS 193X: Web Fun

Announcements

4/3: Homework 0 is out! Due Friday.

4/3: Office hours are now posted.

[View Syllabus](#)

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;  
}
```

CS 193X: Web Fun

Announcements

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

[View Syllabus](#)

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!)

Centering the box

We can also center the box by centering the body tag, since p is now inline-block.

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

CS 193X: Web Fun

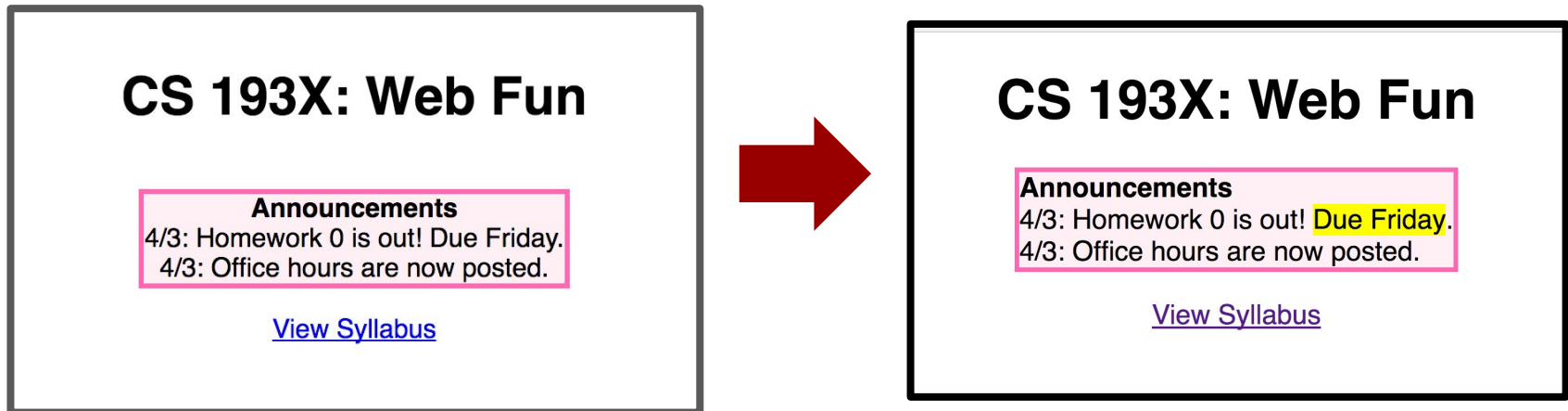
Announcements

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

[View Syllabus](#)

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

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

CSS

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

CS 193X: Web Fun

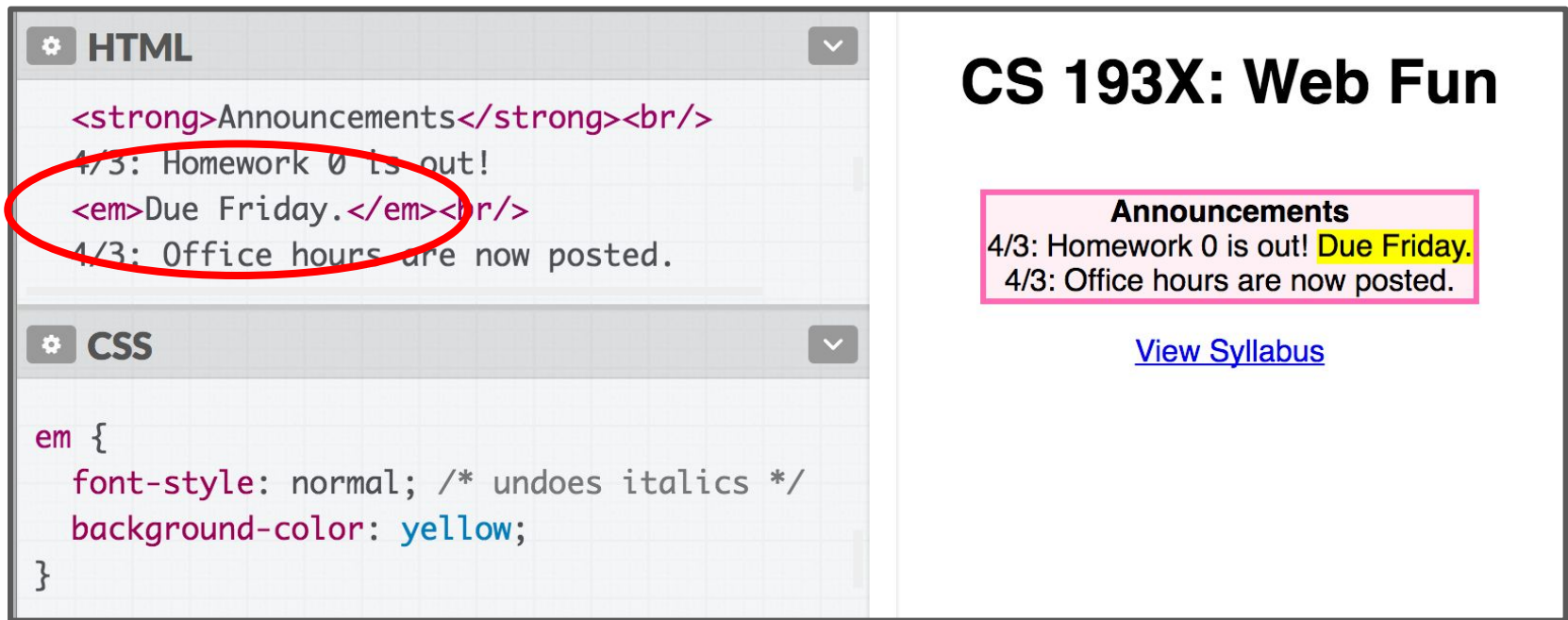
Announcements
4/3: Homework 0 is out! Due Friday.
4/3: Office hours are now posted.

[View Syllabus](#)

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**:



The screenshot shows a web editor interface. On the left, there are two panels: 'HTML' and 'CSS'. The 'HTML' panel contains the following code:

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

The 'CSS' panel contains the following code:

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

On the right, the rendered preview shows the output of the code. It features a heading 'CS 193X: Web Fun' and a section titled 'Announcements'. The text '4/3: Homework 0 is out! Due Friday.' is highlighted in yellow, and '4/3: Office hours are now posted.' is below it. A link 'View Syllabus' is at the bottom of the preview.

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 ``?

What are they?

<div> and

Two generic tags with no intended purpose or style:

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

 in action

We can use as a generic inline HTML container:

HTML

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

CSS

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

CS 193X: Web Fun

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

[View Syllabus](#)

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?

CS 193X: Web Fun

Announcements

4/3: Homework 0 is out! Due Friday.

4/3: Office hours are now posted.

[View Syllabus](#)

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>
```


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>
```

```
.hw {  
    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:
`HW1`
- Often used with span and div to create generic elements: e.g. `` is like creating a "highlight" element

Other selectors:
Next time!

Overflow slides

(we didn't cover these)

element.className

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

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

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

Homework 0

Due Fri

Late cutoff Sun

Lectures

Apr 3: Syllabus

Apr 5: HTML+CSS

Descendent selector

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

HTML

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

CSS

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

Homework 0

Due Fri

Late cutoff Sun

Lectures

Apr 3: Syllabus

Apr 5: HTML+CSS

Descendent selector

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

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

The screenshot shows a web development tool interface. The top panel is labeled 'HTML' and contains the following code:

```
<div class="hw">
  <div>
    <p>
      HW0: <strong>Due Friday</strong>
    </p>
  </div>
  HW1 out <strong>Monday</strong>
</div>
```

The bottom panel is labeled 'CSS' and contains the following code:

```
.hw strong {
  color: red;
}
```

On the right side, there is a preview of the rendered HTML. It shows two lines of text: 'HW0: Due Friday' and 'HW1 out Monday'. The words 'Due' and 'Monday' are highlighted in red, demonstrating the effect of the CSS rule.

Descendent selector

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

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
<i>selector, selector</i>	h2, div	<h2> elements and <div>s

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

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

Course Info

Lectures

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

Honor code

Do the right thing

Selector summary

Example	Example described
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	 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:

- `` tags that are children of `` 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](#)):

- `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.

```
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;  
}
```

 inherits the
font-family property,
but not display:

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

[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)

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.

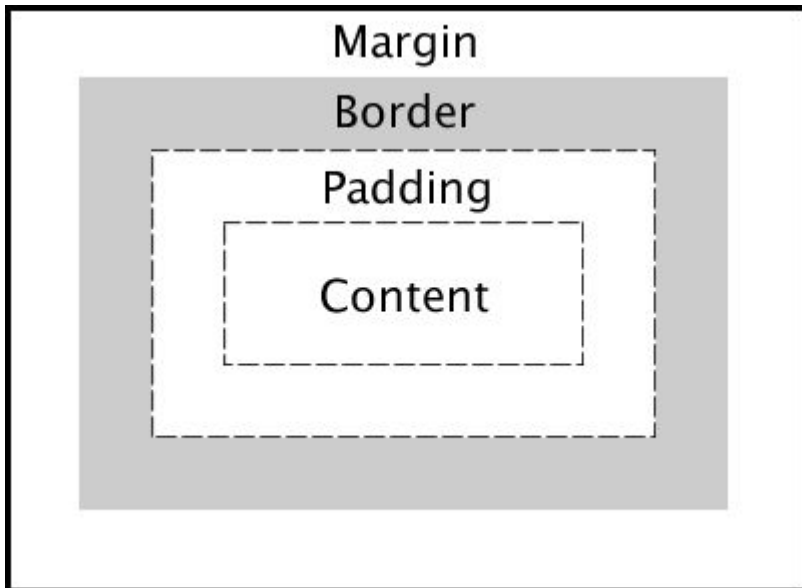
- Is this a good idea?
- Why does HTML have `ids` when you have `classes`?
- Why does HTML have `<p>`, `<h1>`, ``, etc. when you have `<div>`, ``, `class`, and `id`?

CSS Box Model

The CSS Box Model

Every element is composed of 4 layers:

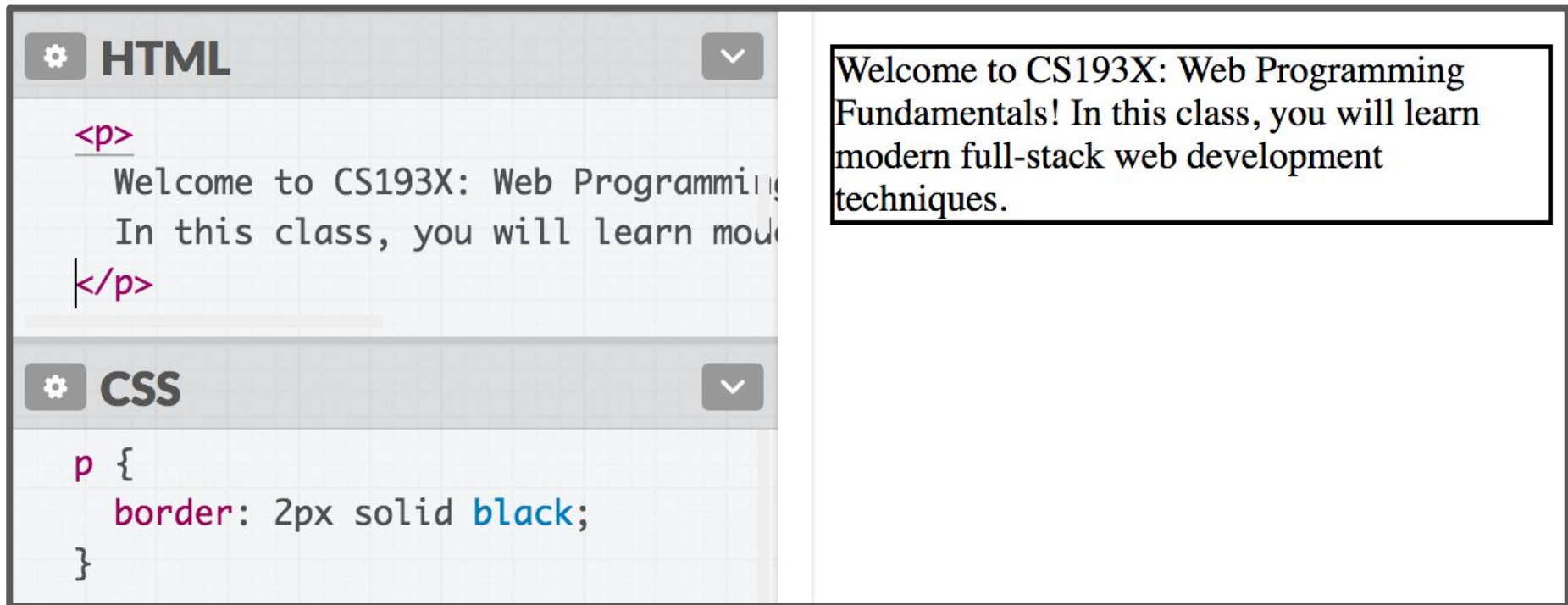
- the element's content
- the **border** around the element's content
- **padding** space between the content and border (inside)
- a **margin** clears the area around border (outside)



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

- It can be used on inline elements but it [behaves differently](#)

border



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; (all styles)`

`border-width: 3px;`

`border-color: purple;`

border

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:

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

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

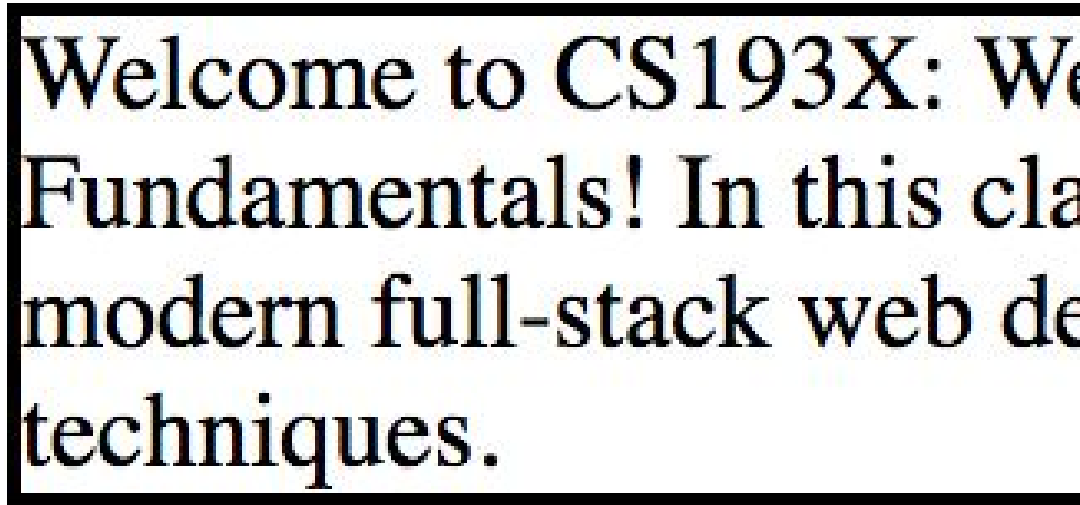
```
p {  
  background-color: purple;  
  border-radius: 10px;  
  color: white;  
}
```

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

Borders look a little squished

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

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



Welcome to CS193X: Web Fundamentals! In this class, we'll learn modern full-stack web development techniques.

padding

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

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

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:



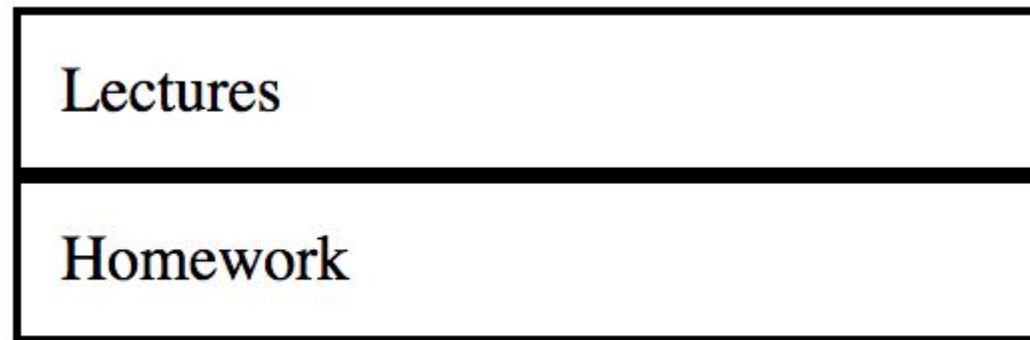
The screenshot shows a code editor with two panels. The left panel, titled 'HTML', contains the following code:

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

The right panel, titled 'CSS', contains the following code:

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

Q: How do we add space between multiple elements?



margin

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

Lectures

Homework

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

The CSS Box Model

Let's revisit our Course web page example:

CS 193X: Web Fun

Announcements

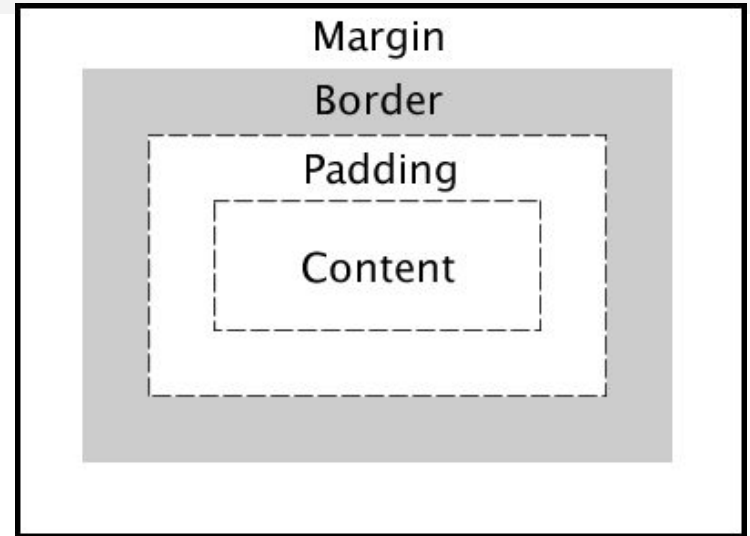
4/3: Homework 0 is out! Due Friday.

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?**

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

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

Make the background color yellow!

Surrounding these paragraphs

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

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