

CS193X: Web Programming Fundamentals

Spring 2017

Victoria Kirst
(vrk@stanford.edu)

Today's schedule

- Syllabus
- Course Info
- Browsers! The Internet!
- A little bit about HTML
- Homework 0 assigned and due **this Friday 4/7**

Check out the course website for all this and more:

<https://cs193x.stanford.edu>

Syllabus

What is CS193X?

Web Programming "Fundamentals"

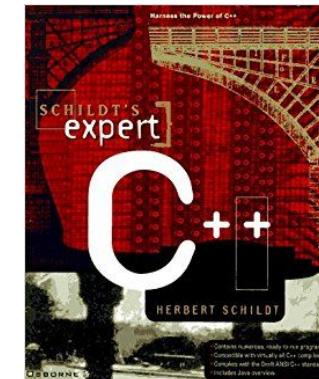
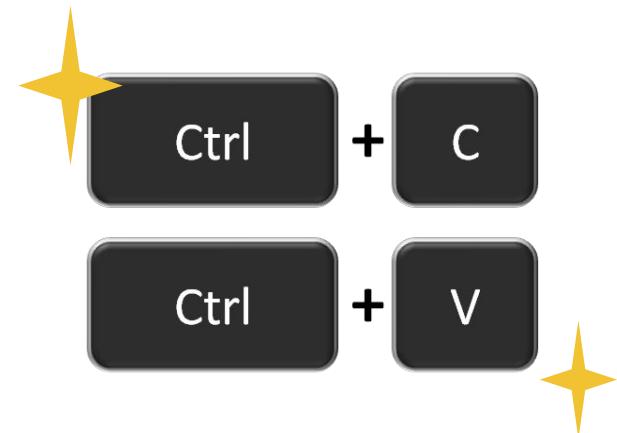
- An introduction to web programming

Q: What does that mean, exactly?

Who are you?

You are:

- A copy/paste programmer of JavaScript, HTML, CSS
(or you've never used these languages)
- A good programmer in at least one real* programming language (Java, C++, etc)
- Frustrated
(maybe)



*In case it's unclear, I'm being facetious

Frustrated?

Every beginner CSS tutorial makes CSS look trivially easy:

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



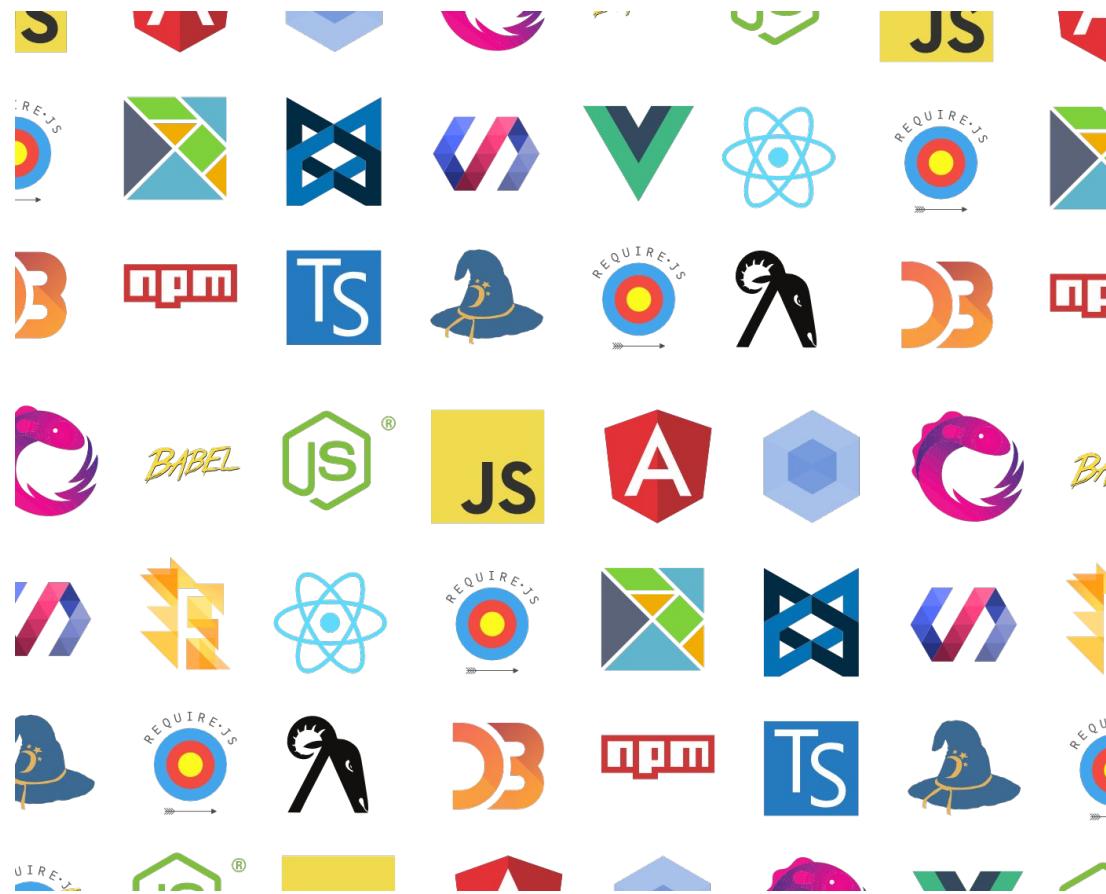
But then when you try to write CSS, literally nothing works:

CSS
IS
AWESOME

A rectangular box containing the text "CSS IS AWESOME" in a bold, sans-serif font. The text is stacked vertically: "CSS" on top, "IS" in the middle, and "AWESOME" at the bottom. The box has a thin black border.

Frustrated?

You want to learn JavaScript...



...but you're
overwhelmed by all
the frameworks,
libraries, tools, etc
and have no idea
where to start.

What is CS193X?

Web Programming "Fundamentals"

- An **opinionated, hopefully frustration-free** introduction
to web programming

What is CS193X?

Opinionated:

- There are many ways to do things on the web: can't learn them all at once!
- CS193x: What I think you need to know as a beginner

Hopefully frustration-free:

- We will go slowly through the essential concepts and speed through the obvious stuff
- You are **not** expected to fill in the gaps via Google and StackOverflow

CS193X Goals

If you never take another web programming class again, you will leave CS193X with the following skills:

- Create **attractive, small scale web sites or apps** that at least mostly work on phones
- Have the **vocabulary and background knowledge** to understand technical writing/discussions about the web (e.g. web API documentation; random blog posts)
- Have the **foundation** to pursue the areas of web programming that you're interested in (if you choose)

(CS193X Non-goals)

CS193X is **not** a class to take to learn how to code.

- We are enforcing a 106B prereq. HW2 involves recursion!

CS193X is **not** a class that will turn you into a senior frontend/backend developer.

- Nor is any class; software takes years of experience to develop expertise.

CS193X is **not** a class that will teach you all there is to know about web programming.

- In particular, we will **not** teach how to go from fundamentals to "production-level" architecture.

CS193X, in detail

- HTML
- CSS
- JavaScript
- Backend basics
 - Server on NodeJS + Express
 - Database via MongoDB and Mongoose

(Uh...)

CS193X, in detail

- HTML
- CSS
- JavaScript
- Backend basics
 - Server on NodeJS + Express
 - Database via MongoDB and Mongoose

(Uh...

- a) How is this an "opinionated" list of topics?
- b) How is this different from CS142?)

CS193X, in detail

- HTML
- CSS
- JavaScript
- Backend basics
 - Server on NodeJS + Express
 - Database via MongoDB and Mongoose

(Uh...

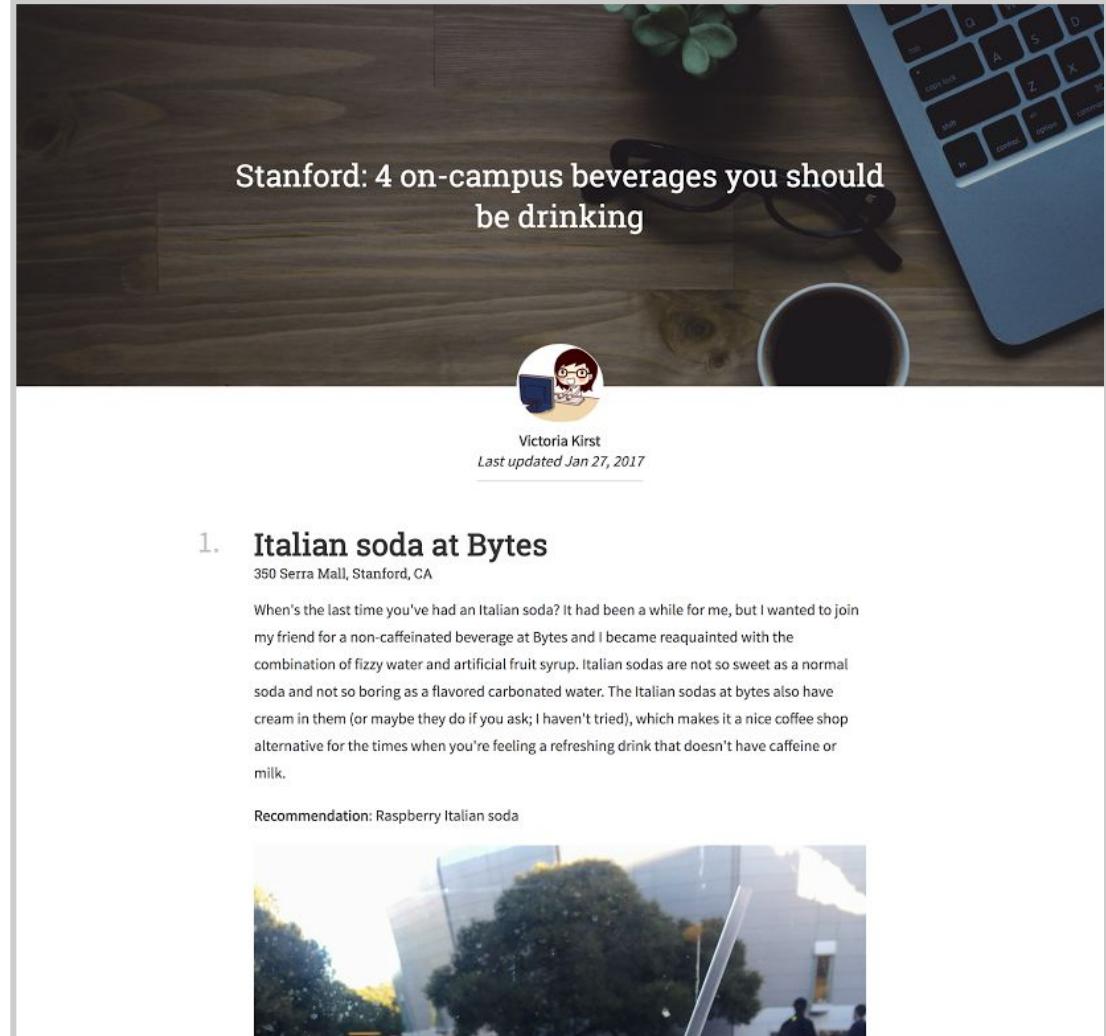
- a) How is this an "opinionated" list of topics?
- b) How is this different from CS142?)

→ **The difference is in the focus and the content itself**

CS193X: CSS, applied

HW1 will ask you to make a webpage that looks like this-ish:

(Note: HW1 is not released yet.)



The screenshot shows a webpage with a wooden background. At the top, there's a title: "Stanford: 4 on-campus beverages you should be drinking". Below the title is a small circular profile picture of a person with glasses. To the right of the profile picture, the name "Victoria Kirst" is displayed, followed by the text "Last updated Jan 27, 2017". A horizontal line separates this header from the main content. The main content is a numbered list: "1. Italian soda at Bytes". Underneath the list, it says "350 Serra Mall, Stanford, CA". The text describes the Italian soda at Bytes, mentioning its combination of fizzy water and artificial fruit syrup, noting it's not as sweet as a normal soda and not as boring as flavored carbonated water. It also mentions that the sodas have cream in them. Below this text is another horizontal line. Underneath the line, it says "Recommendation: Raspberry Italian soda". At the bottom of the page, there's a small image of a modern building with glass windows and green trees in front of it.

Stanford: 4 on-campus beverages you should be drinking

 Victoria Kirst
Last updated Jan 27, 2017

1. **Italian soda at Bytes**
350 Serra Mall, Stanford, CA

When's the last time you've had an Italian soda? It had been a while for me, but I wanted to join my friend for a non-caffeinated beverage at Bytes and I became reacquainted with the combination of fizzy water and artificial fruit syrup. Italian sodas are not so sweet as a normal soda and not so boring as a flavored carbonated water. The Italian sodas at bytes also have cream in them (or maybe they do if you ask; I haven't tried), which makes it a nice coffee shop alternative for the times when you're feeling a refreshing drink that doesn't have caffeine or milk.

Recommendation: Raspberry Italian soda



CS193X: CSS

HTML (~1 day)

- Key concepts: inline, block, inline-block

CSS (~1.5 weeks)

- Multiple rendering styles: natural, flex, positioned, float
- Mobile layouts
- Transforms and animations
- **FYI: No libraries or compiled CSS**

CS193X: Modern JS / ES6+

Later in the quarter, we will read and write JavaScript that looks sort of like this:

```
(async () => {
  let choice = 'e';
  do {
    choice = await askQuestion('Enter choice');
    await processChoice(choice);
  } while (choice != 'e');
})();
```

CS193X: Modern JS / ES6+

JavaScript (~5 weeks)

- JavaScript classes
- Relevant functional programming
 - Lambdas
 - Generator functions and async/await
 - "Fat arrow" vs function
 - Closures
- Creating and using Promises
- Understanding the Event Loop
- Modules and encapsulation

NO frontend framework; minimal libraries
No Angular/React/JQuery/etc

CS193X: JavaScript, applied

HW2 will ask you to make a webpage that looks like this-ish:

Pick A Dog And We'll Reveal A Deep Truth About You

This is your pupper of truth.

Pick a pup:

Below the main image are three smaller images of dogs, each with a yellow border around its photo frame. The first image shows a golden retriever puppy. The second image shows a brown dog. The third image shows another golden retriever puppy. There are also other partially visible images at the bottom.

CS193X: JavaScript, applied

And HW2 will also ask you to write two small Chrome extensions:

(Note: HW2 is also not released yet.)

Extensions

 Evil Extension 1.0
Permissions Reload (⌘R)
 Allow in incognito Allow access to file URLs

 Twitter Gardener 1.0
Add some positive affirmations to the Twitter feed you're viewing.
Permissions Reload (⌘R)
 Allow in incognito Allow access to file URLs

CS193X: Baby's first backend

CS193X coverage of server-side programming will be light.

Backend stack:

NodeJS + Express + MongoDB via Mongoose ([~3 weeks](#))

- What is a server
- What is npm
- How to serve static web pages
- How to server JSON via REST APIs
- Writing to and loading from a database
- Authentication via OAuth2 (i.e. login via Gmail account)

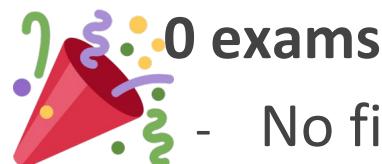
CS193X Structure

"Homework 0" + 6 homeworks

- Each homework will be a standalone web page or a very small standalone web app
- Each homework will have a multiple choice "mini-homework" attached to it

1 final project

- Open-ended! Details to come.
- ~1 week in scope



- No final, no midterm, no exams



CS193X Structure

"Homework 0" + 6 homeworks

- Each homework will be a standalone web page or a very small standalone web app
- Each homework will have a multiple choice "mini-homework" attached to it

1 final project

- Open-ended! Details to come.
- ~1 week in scope

0 exams

- No final, no midterm, no exams



Also CS193X
does **not** count
for any CS
credit, not even
elective!

CS142 vs CS193X summarized

CS193X: ([syllabus](#))

Content: In-depth coverage of CSS and JavaScript; no frontend framework; shallow coverage of backend

Homework: Multiple small standalone apps

Final: Open-ended final project

CS? Counts for [no CS credit](#)

CS142: ([syllabus](#))

Content: Moves fast through CSS and raw JavaScript; uses AngularJS; deeper dive into backend

Homework: Building a large photo sharing app

Final: Has a midterm and final

CS? Counts for [CS elective](#)

Course info

Disclaimer

This is the first ever offering of CS193X, meaning:

- **Everything is subject to change.** Including everything I've just told you and everything I'm about to tell you.
- **There will be all the mistakes of a new course!**
 - Bugs in homework
 - Awkward lectures
 - Things that are too hard / too easy

Please be patient with us! We are also soliciting your constructive feedback.

Grades

Homework: 60%

Mini-HWs: 5%

Final Project: 35%

- **Mini-Homeworks:** multiple-choice quizzes to complete before starting the regular assignment. Can retry as many times as you want without penalty
- **Final Project:** Details to come later in the quarter.

Lateness policy

- Every homework may be submitted up to 48 hours after the deadline, without penalty.
- Homework submitted on time will receive a small bonus to their homework score.
- Submissions are **not accepted** beyond the 48-hour grace period. The grace period is strictly enforced.
- The final project must be turned in on time.

Browser and Text editor/IDE

- **Text editor:** You can use whatever you want. We recommend [Atom](#).
- **Browser:** Your code must work on [Chrome](#), as that is what your TAs will use when grading your homework. It will not be tested in any other browser.
- **Homework turn-in:** We are using GitHub Classroom for assignment turnin.

Complete [Homework 0](#) to get all set up with your homework workflow in CS193X!

Honor code

The web platform allows you to **view the source code** of any website you visit.

- Great for learning!
- But what about Honor Code?

Honor code

- **DON'T** look at other people's CS193X solutions
- **DON'T** publish homework source code publicly on GitHub, StackOverflow, personal web page, etc.
- **OK** to look at other website's code for inspiration (though it should rarely be necessary in this class)
- **OK** to look at StackOverflow / Google / etc for help (though it should rarely be necessary in this class)
- **OK** to share a webpage you made in CS193X to show off the webpage itself
- **DON'T** share a webpage you made in CS193X with the intent to share the code

([See full description](#))

Honor code

- **DON'T** look at other people's CS193X solutions
- **DON'T** publish homework source code publicly on GitHub, StackOverflow, personal web page, etc.
- **OK** to look at other website's code for inspiration (though it should rarely be necessary in this class)
- **OK** to look at StackOverflow / Google (though it should rarely be necessary in this class)
- **OK** to share a webpage you made if you did not intent to share the code
- **DON'T** share a webpage you made if you did not intent to share the code

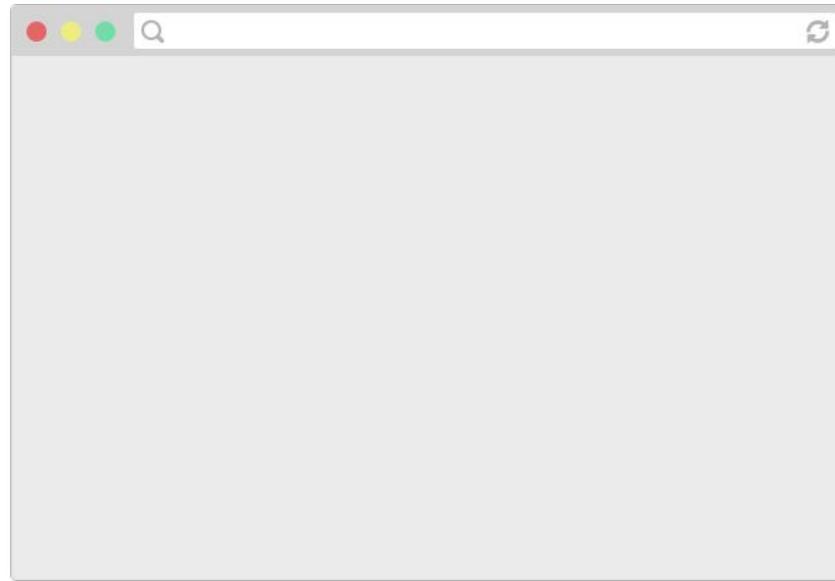
tl;dr: If your intentions are good, it's probably OK.

([See full description](#))

Questions?

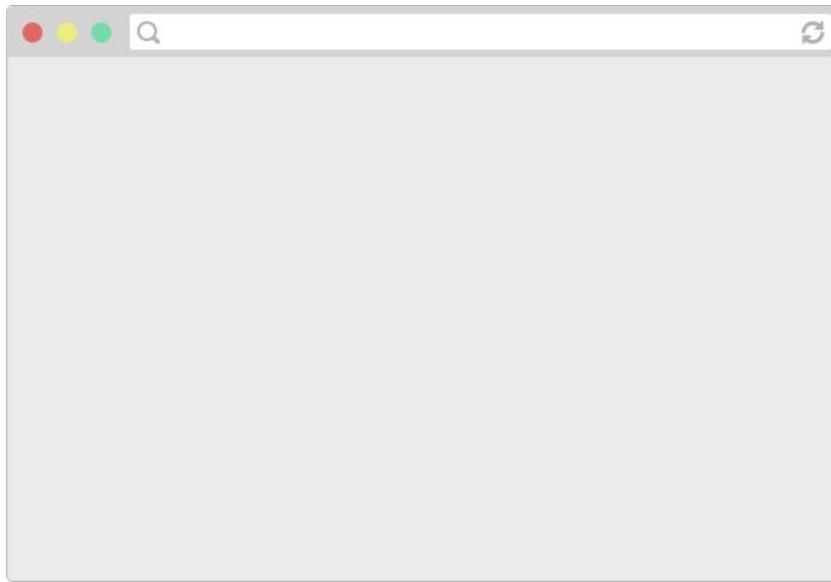
Browsers!
The Internet!
The web!

How do web pages work?



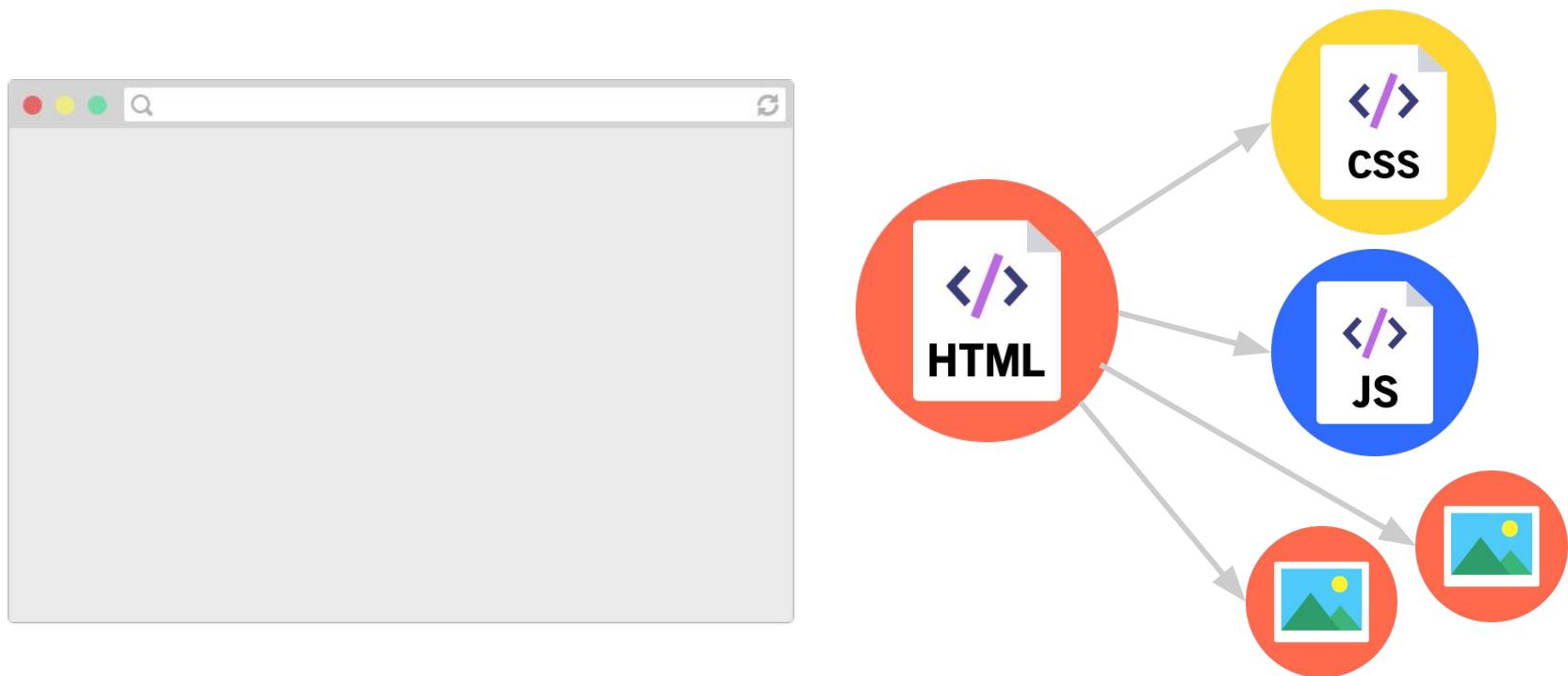
Browsers are applications that can display web pages.
E.g. Chrome, Firefox, Safari, Internet Explorer, Edge, etc.

How do web pages work?



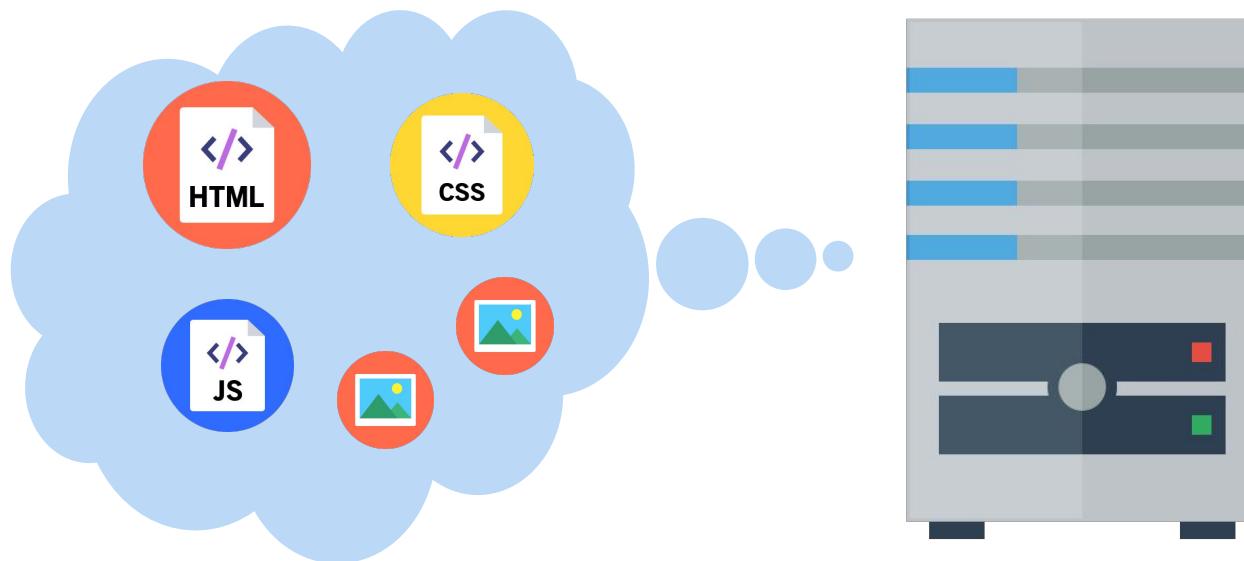
Web pages are written in a markup language called **HTML**, so browsers display a web page by reading and interpreting its HTML.

How do web pages work?



The HTML file might link to other resources, like images, videos, as well as **JavaScript** and **CSS** (stylesheet) files, which the browser then also loads.

How do web pages work?

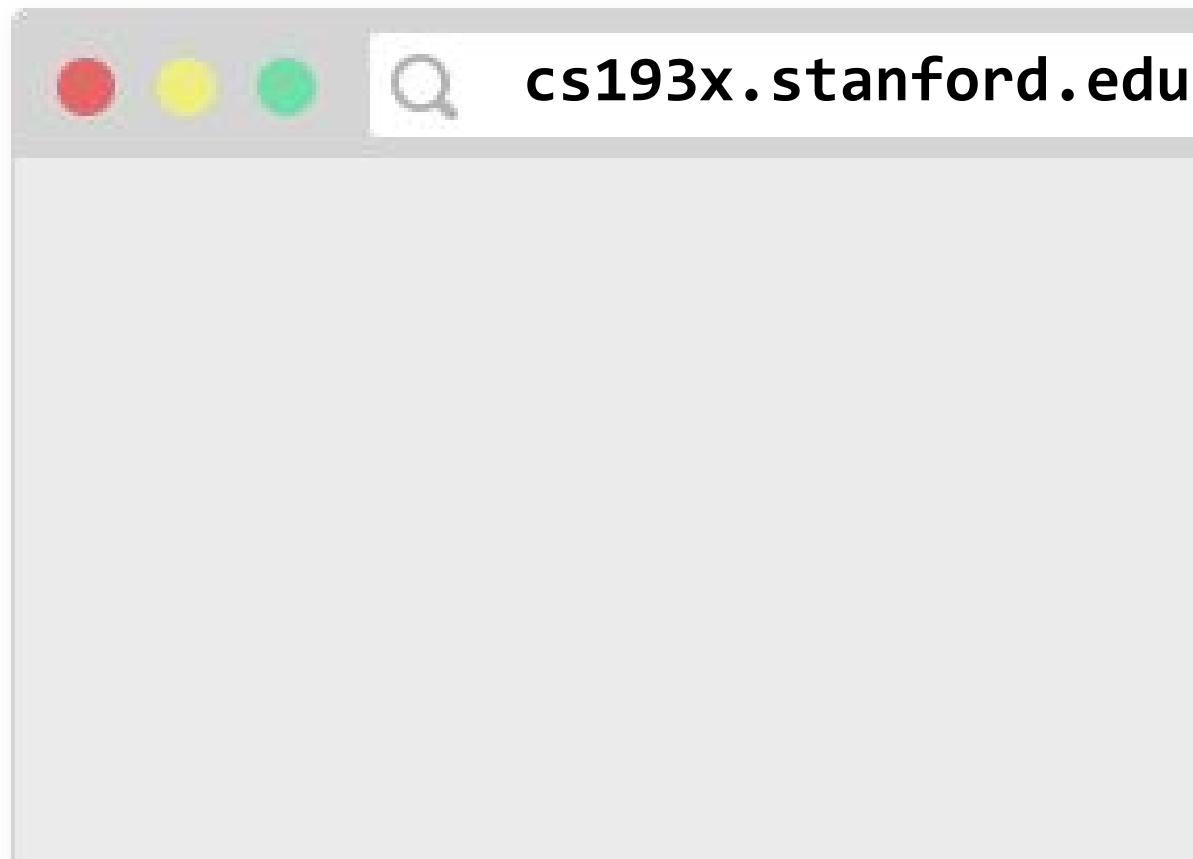


A **web server** is a program running on a computer that delivers web pages in response to requests.

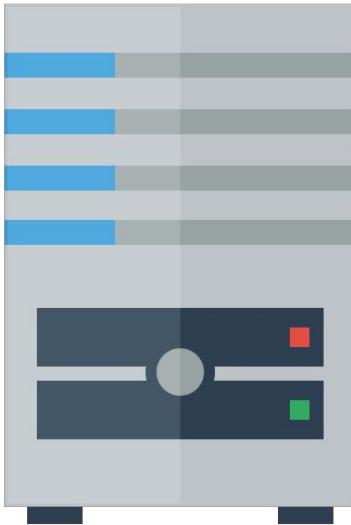
It either stores or generates the web page returned.

How do web pages work?

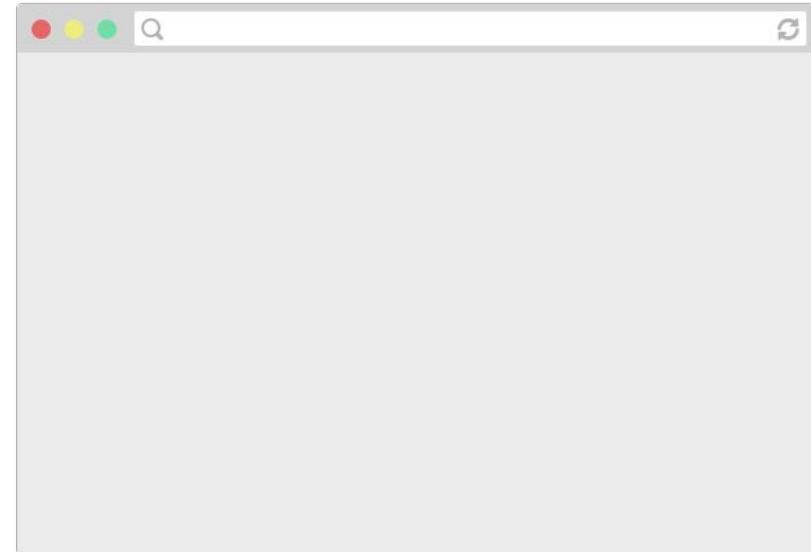
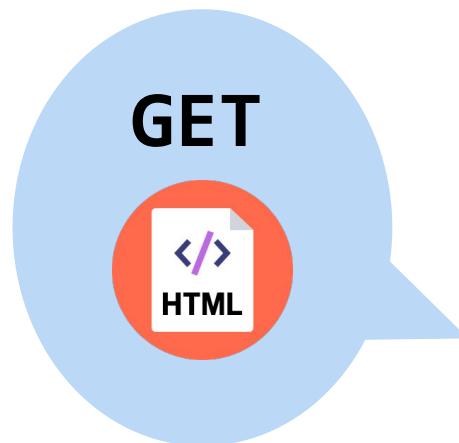
1. You type
in a URL,
which is the
address of
the HTML
file on the
internet.



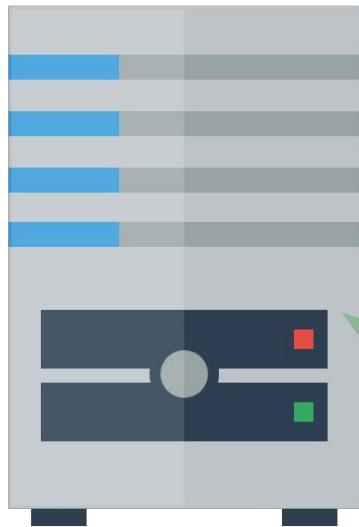
How do web pages work?



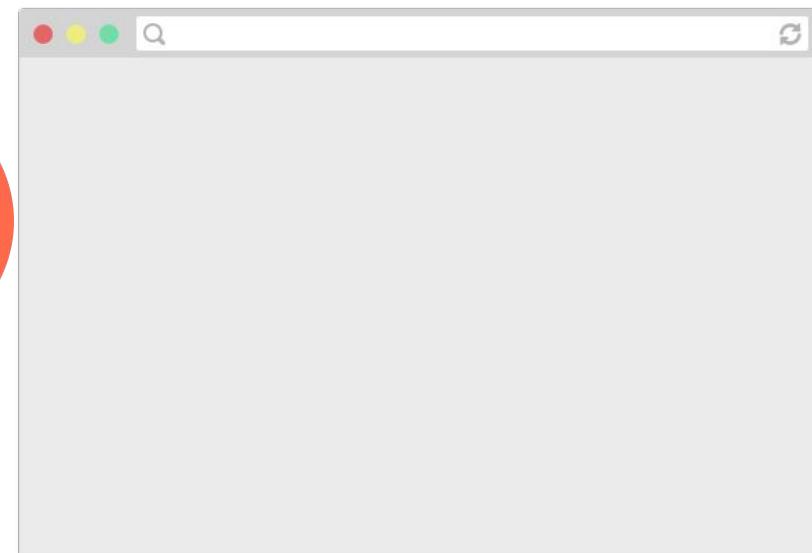
2. The browser asks the web server that hosts the document to send that document.



How do web pages work?



3. The web server responds to the browser with HTML file that was requested.

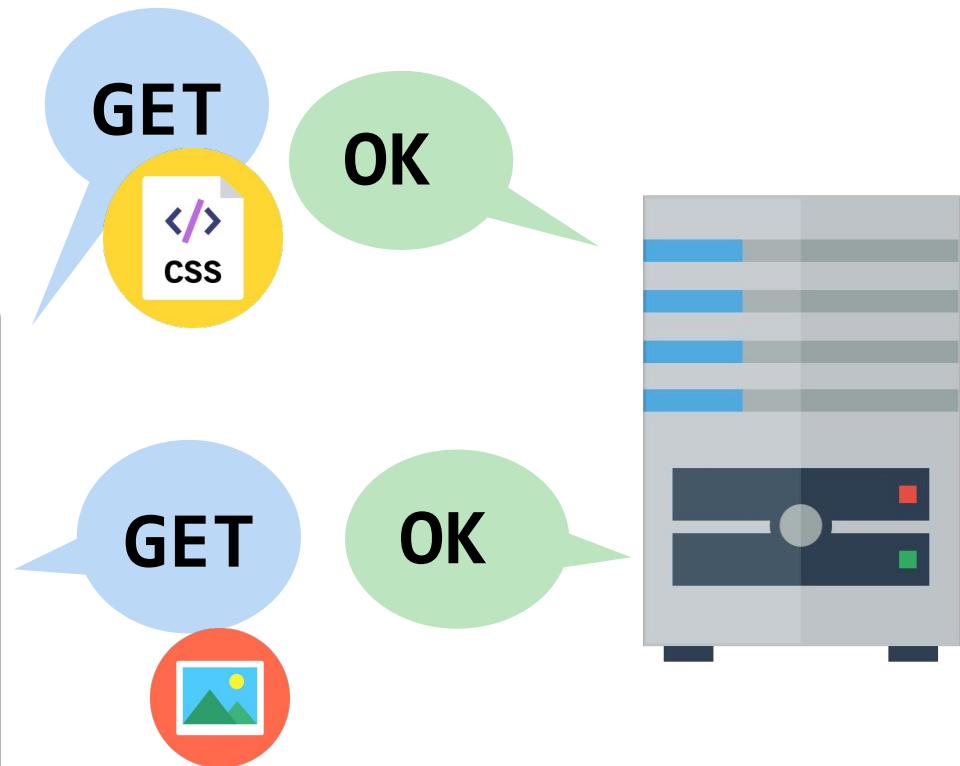
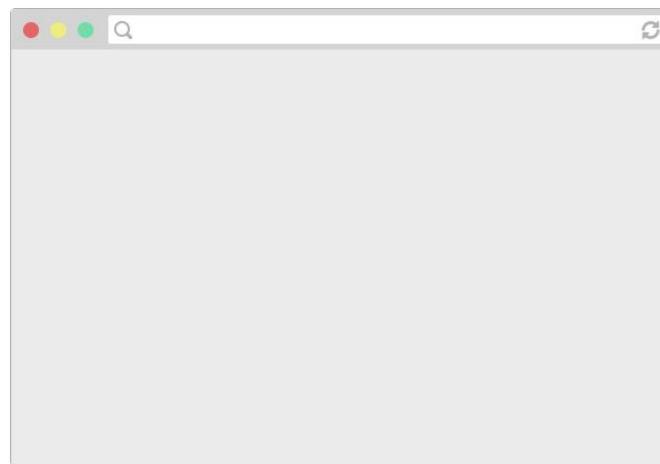


How do web pages work?

4. The browser reads the HTML, sees the embedded resources and asks the server for those as well.



...



How do web pages work?

5. The web page is loaded when all the resources are fetched and displayed.



P.S.

(That was obviously very hand-wavy. We'll get more detailed when we talk about servers later in the quarter.)

The screenshot shows a web browser window for the URL `cs193x.stanford.edu`. The page has a light gray header bar with the address and a search icon. Below the header is a navigation bar with links for SYLLABUS, INFO, STAFF, LECTURES, and HOMEWORK. The main content area features a dark background with white line-art icons of various computer components like monitors, keyboards, and servers. Overlaid on this background are the course details:

Stanford CS Department
CS193X: Spring 2017

Welcome to CS193X: Web Programming Fundamentals! This is an opinionated introduction to web programming. In this class, you will learn modern full-stack web development techniques without use of a frontend framework.

- [Prereqs](#) CS106B; Unix/command-line proficiency or CS1U
- [Lectures](#) Mon-Wed-Fri, 1:30-2:20 in Shriram Center 104
- [Final exam](#) No final exam.
- [Final project](#) Yes, there will be one. Details TBA.

CS193X is limited enrollment. This is the first offering of CS193X, so enrollment is limited and by application only. It will be offered again sometime in the 2017-2018 school year.

Course staff

HTML and CSS

HTML and CSS strategy

Assumption: Most people have cursory familiarity with HTML and CSS. Therefore we will:

- **Speed through** the obvious stuff
- **Skip** self-explanatory syntax
- **Skip** the parts you can look up easily through Google

★ **Therefore, be aggressive with questions!** ★

What is 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 <**emem**>
 <**img** **src**=**"puppy.png"** />

</**p**>

Basic HTML page structure

(i.e. copy/paste boilerplate)

```
<!DOCTYPE html>
<html>
  <head>
    <title>CS 193X</title>
  </head>

  <body>
    ... contents of the page...
  </body>
</html>
```

Saved in a *filename.html* file.

Basic HTML page structure

(i.e. copy/paste boilerplate)

Metadata that
doesn't appear in
the viewport of
the browser

```
<!DOCTYPE html>
<html>
  <head>
    <title>CS 193X</title>
  </head>
  <body>
    ... contents of the page...
  </body>
</html>
```

Contents that
render in the
viewport of the
browser

E.g. **<title>**
shows up as the
name of the tab

HTML elements

<p>

HTML is awesome!!!

</p>

- An element usually has start and ending tags (<p> and </p>)
 - **content:** stuff in between start and end tags
- An element can be self-closing (img)
- An element can have attributes (src="puppy.jpg")
- Elements can contain other elements (p contains em and img)

Some HTML elements

(to place within <body>)

Top-level heading h1, h2, ... h6	<h1>Moby Dick</h1>
Paragraph	<p>Call me Ishmael.</p>
Line break	since feeling is first who pays any attention
Image	
Link	click here!
Strong (bold)	Be BOLD
Emphasis (italic)	He 's my brother and all

Exercise: Course web page

Let's write some HTML to make the following page:

The screenshot shows a web browser window with a light gray header bar. In the top left are red, yellow, and green window control buttons. The title bar in the center contains a white folder icon followed by the text "CS 193X". To the right of the title bar is a close button (an "x"). On the far right of the header bar, the text "Victoria Perso..." is visible. Below the header bar is a toolbar with several icons: back, forward, refresh, home, and search. The search bar contains the URL "file:///Users/victoriakirst/cs193x...". To the right of the search bar are icons for a magnifying glass, a star, a square with a circle, and three dots. The main content area of the browser displays the following HTML code:

```
CS 193X: Web Fun

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

View Syllabus
```

Exercise: Course web page

HTML boilerplate

```
<!DOCTYPE html>
<html>
  <head>
    <title>CS 193X</title>
  </head>

  <body>
    ...
  </body>
</html>
```

Plaintext contents of the page

CS 193X: Web Fun

Announcements

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

4/3: Office hours are now posted.

[View Syllabus](#)

Solution

```
<!DOCTYPE html>
<html>
  <head>
    <title>CS 193X</title>
  </head>
  <body>
    <h1>CS 193X: Web Fun</h1>
    <strong>Announcements</strong><br/>
    4/3: Homework 0 is out!<br/>
    4/3: Office hours are now posted.<br/>
    <br/>
    <a href="http://cs193x.stanford.edu/syllabus">
      View Syllabus
    </a>
  </body>
</html>
```

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

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;  
}
```

CSS

A CSS file is composed of **style rules**:

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

selector: Specifies the HTML element(s) to style.

property: The name of the CSS style.

value: The value for the CSS style.

Saved in a *filename.css* file.

CSS

// NOT REAL CSS

```
fork {  
  color: gold;  
}
```

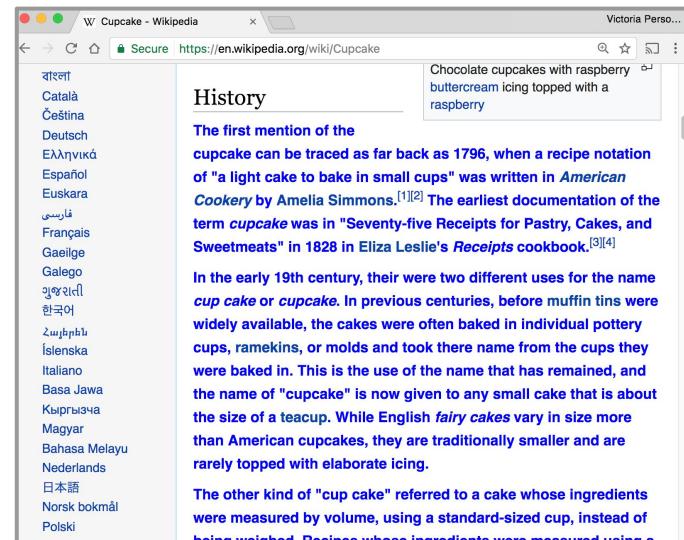
"All forks on the table
should be gold"



CSS

```
p {  
    color: blue;  
    font-weight: bold;  
}
```

"All <p> elements on the page
should be blue and bold"



Linking CSS in HTML

(i.e. copy/paste boilerplate)

```
<!DOCTYPE html>
<html>
  <head>
    <title>CS 193X</title>
    <link rel="stylesheet" href="filename.css" />
  </head>

  <body>
    ... contents of the page...
  </body>
</html>
```

Some CSS properties

There are over [500 CSS properties](#)! Here are a few:

Font face (mdn)	font-family: Helvetica;
Font color (mdn)	color: gray;
Background color (mdn)	background-color: red;
Border (mdn)	border: 3px solid green;
Text alignment (mdn)	text-align: center;

Aside: [Mozilla Developer Network](#) (MDN) is the best reference for HTML elements and CSS properties

- The actual W3 spec is very hard to read (meant for browser developers, not web developers)

Main ways to define CSS colors:

140 predefined names ([list](#))

```
color: black;
```

rgb() and rgba()

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

```
color: rgba(0, 0, 0, 0.5);
```

Hex values

```
color: #00ff00;
```

```
color: #0f0;
```

```
color: #00ff0080;
```

- The "a" stands for **alpha channel** and is a **transparency** value
- Generally prefer more descriptive over less:
 1. Predefined name
 2. rgb / rgba
 3. Hex

Exercise: Course web page

Let's write some CSS to style our page:

Font face: Helvetica

Border: hotpink 3px

Background color:

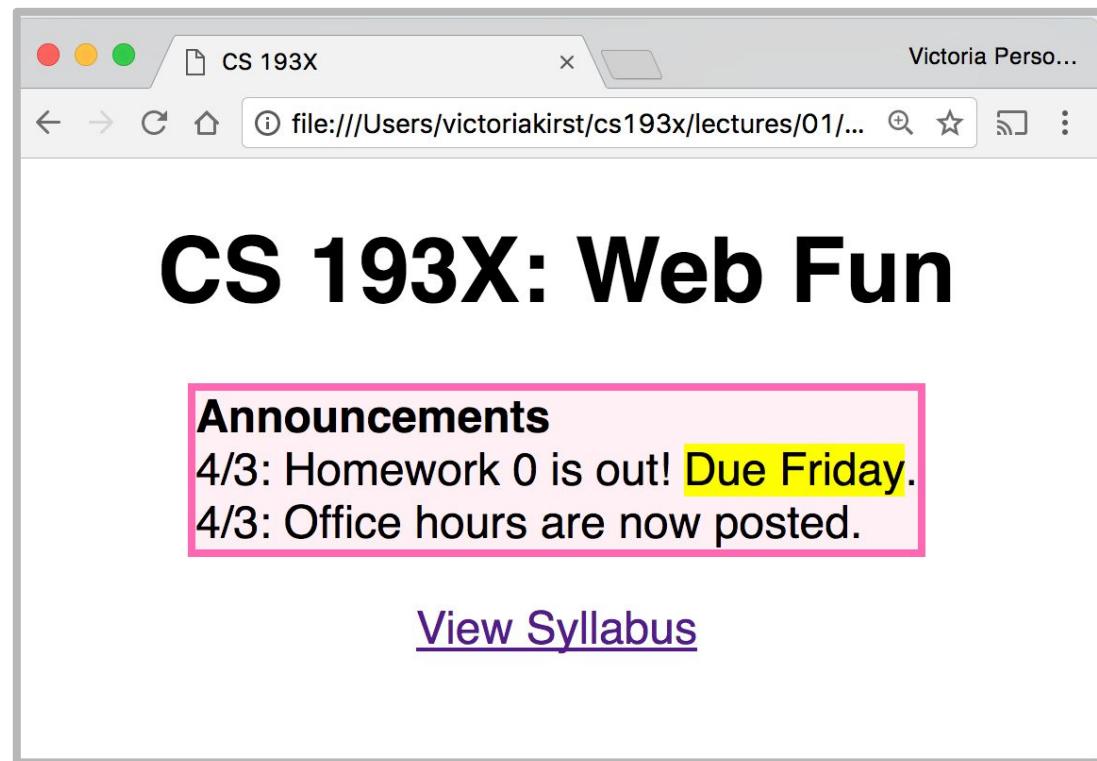
lavenderblush

Highlight: yellow

- Box is **centered**

- Header and link are
centered

- Box contents are
left-aligned



CSS exercise debrief

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.)

But we encountered **more weirdness...**

- Couldn't set `text-align: center;` to the `<a>` or `` tags directly, but could center `<p>` and `<h1>`
- Had to set a `width` on the box to make it hug the text ... any other way to do this?
- How to center the box?! How do you highlight?!

Q: Why is HTML/CSS
so bizarre??

A: Next time!

Homework 0 is
out now, due this Friday
April 7

Overflow slides

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 **< 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:

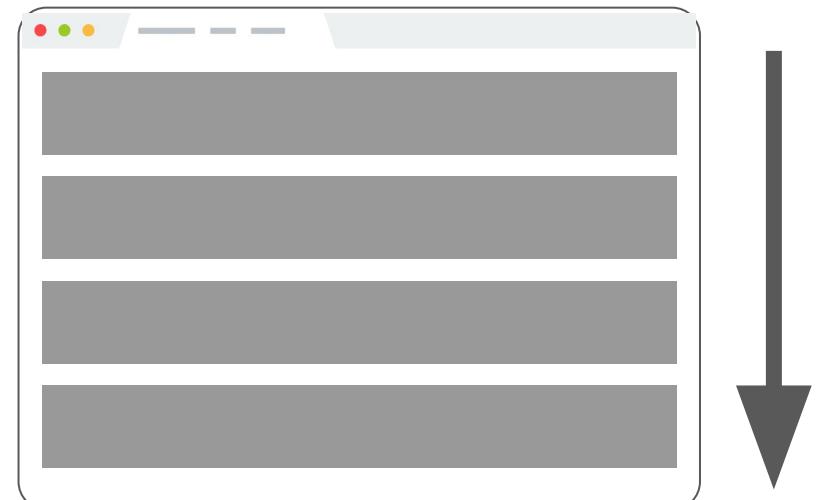
1. **block**: large blocks of content, has height and width
`<p>`, `<h1>`, `<blockquote>`, ``, ``, `<table>`
2. **inline**: small amount of content, no height or width
`<a>`, ``, ``, `
`
 - a. **inline block**: inline content with height and width
``
3. metadata: information about the page, usually not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

Block elements

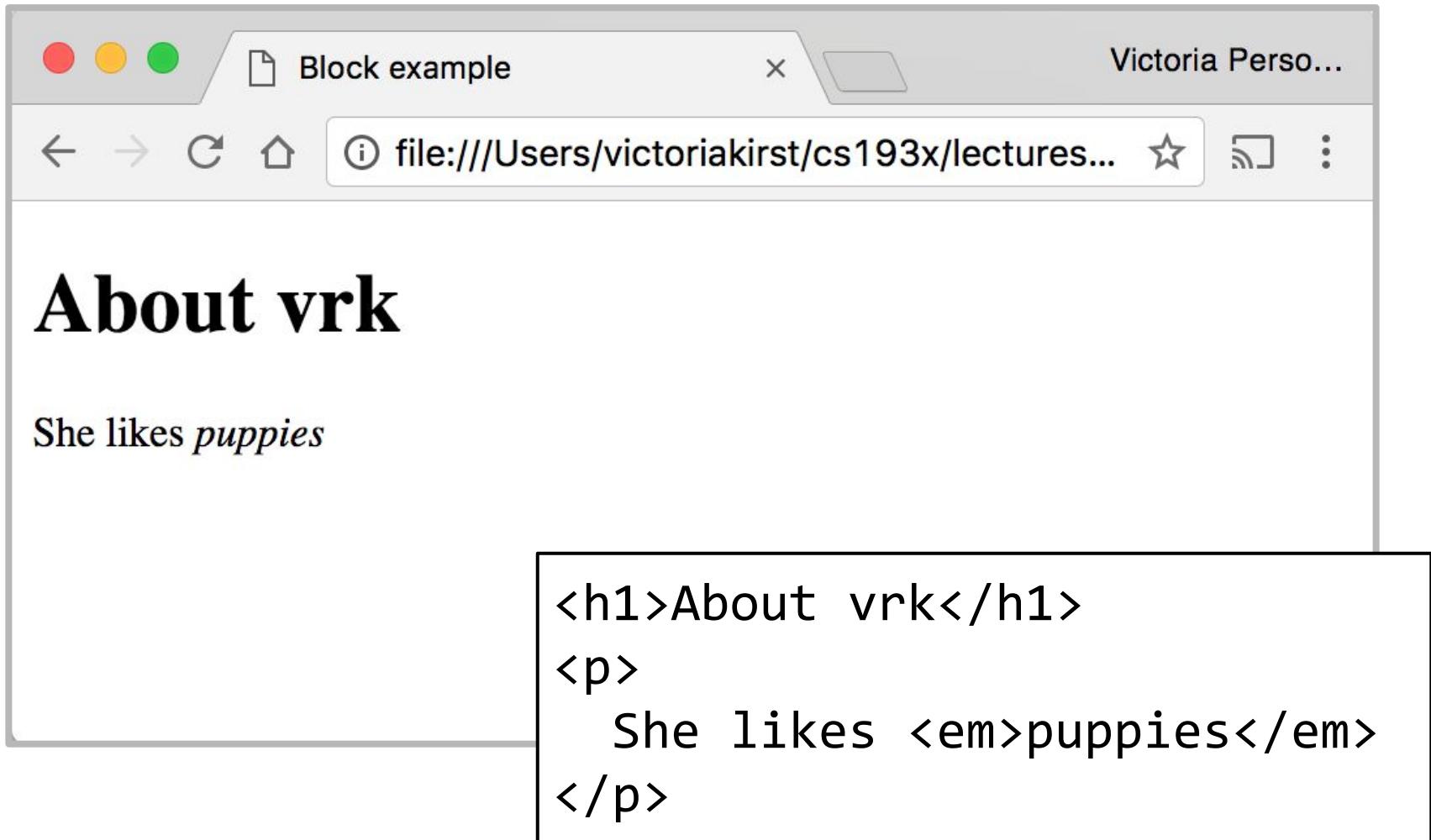
Examples:

`<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

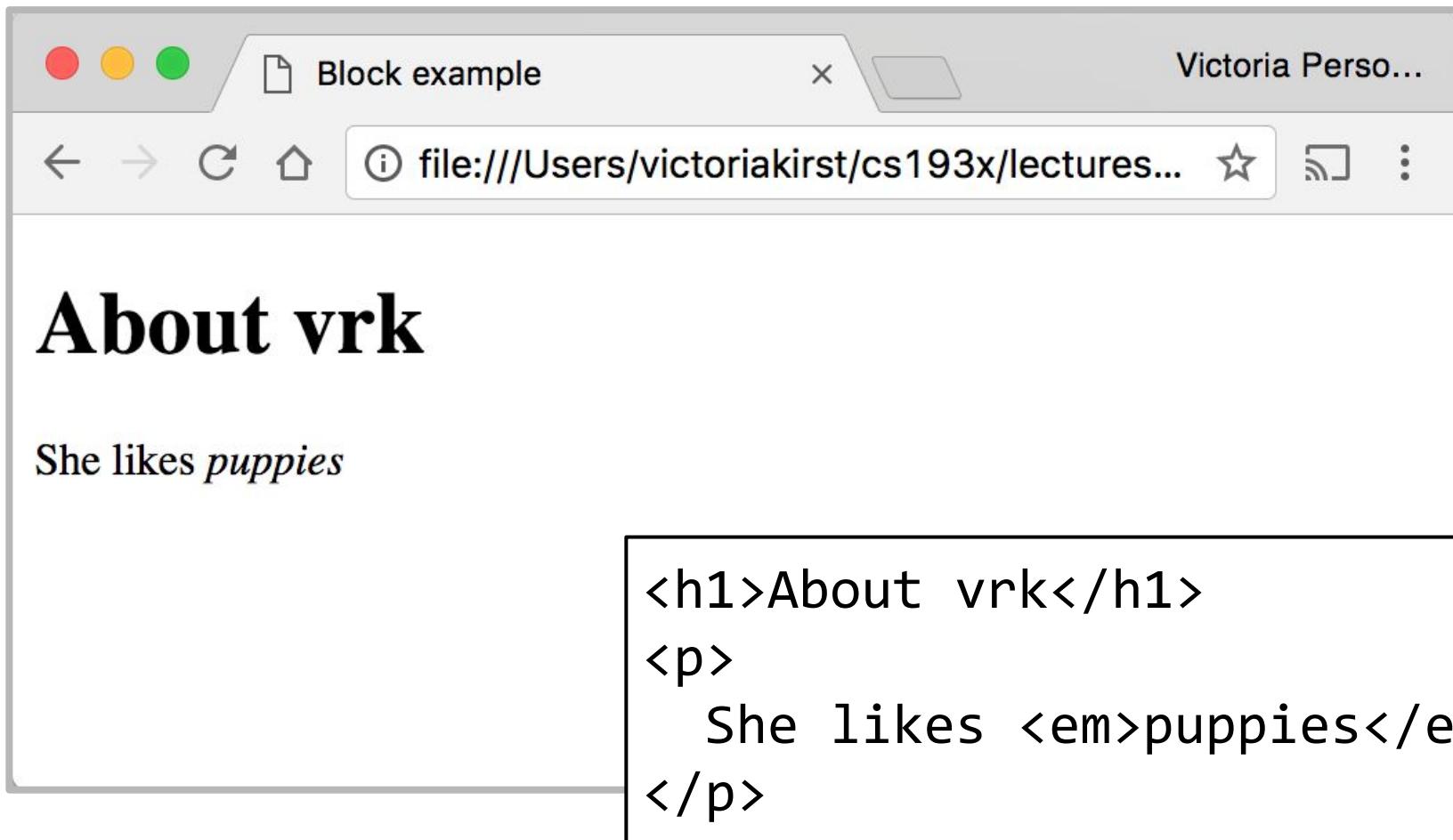


A screenshot of a web browser window titled "Block example". The address bar shows the URL "file:///Users/victoriakirst/cs193x/lectures...". The main content area displays the text "About vrk" followed by "She likes *puppies*". A callout box with a black border highlights the first paragraph, containing the HTML code: <h1>About vrk</h1><p>She likes puppies</p>".

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

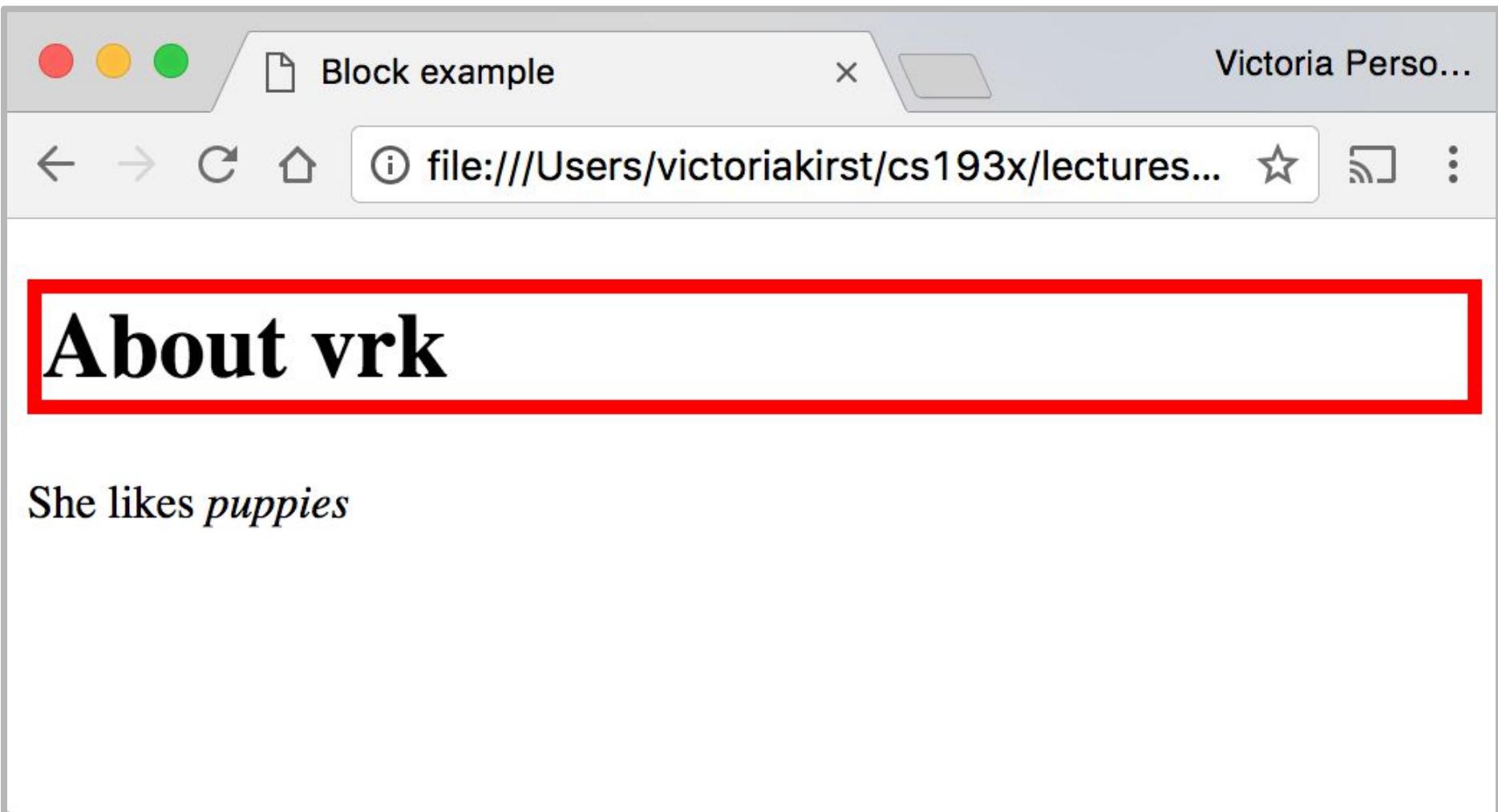
Q: What does this look like in the browser?

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



The screenshot shows a web browser window with the title "Block example". The address bar indicates the page is at "file:///Users/victoriakirst/cs193x/lectures...". The main content area displays the text "About vrk" in a large, bold, black font. Below it, the text "She likes *puppies*" is displayed in a smaller, regular black font. A red border is visible around the "About vrk" heading, demonstrating the effect of the CSS rule shown in the code block above.

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



Block-level: extends the full width of the page

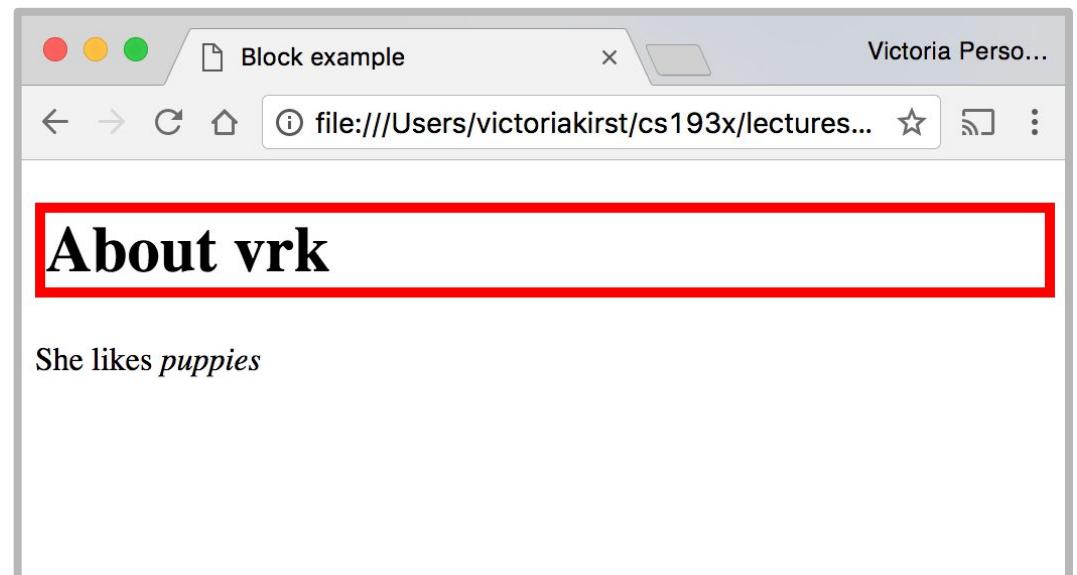
```
h1 {  
    border: 5px solid red;  
}
```

```
<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

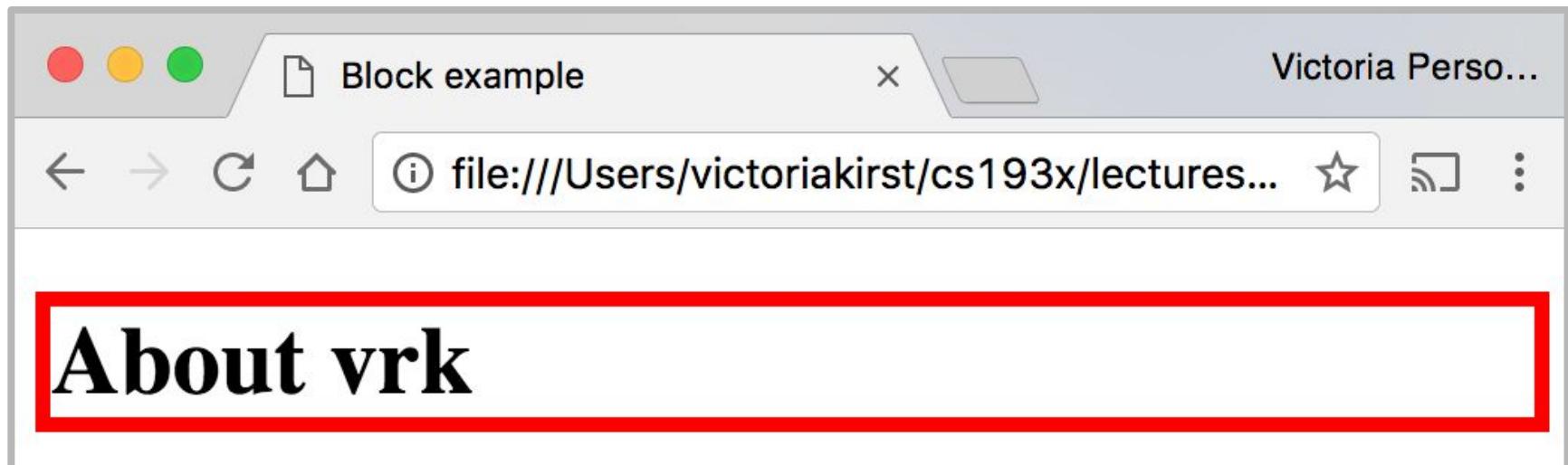
Note how block-level
elements (**h1**, p) flow
top to bottom

See: [JSBin](#)



Q: What does this look like in the browser?

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



She likes *puppies*

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



Block example



Victoria Perso...

← → C ⌂ ⌄ file:///Users/victoriakirst/cs193x/lectures... ☆ ⌁ ⌂ ⌂

About vrk

She likes *puppies*

Block-level

width can be modified

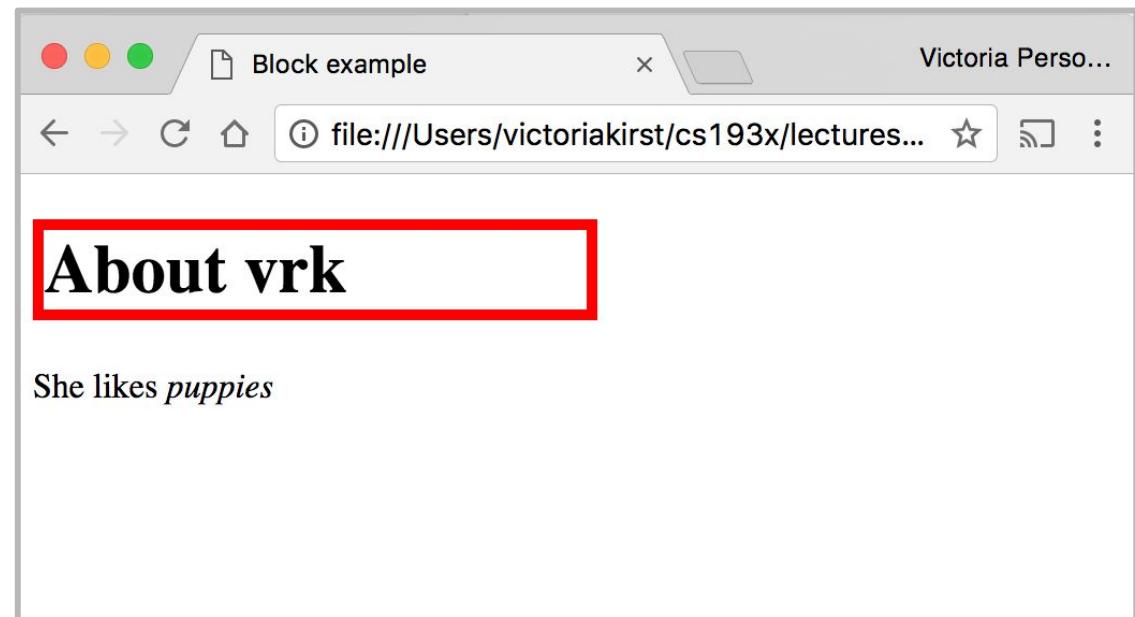
```
h1 {  
    border: 5px solid red;  
    width: 50%;  
}
```

```
<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: [JSBin](#)

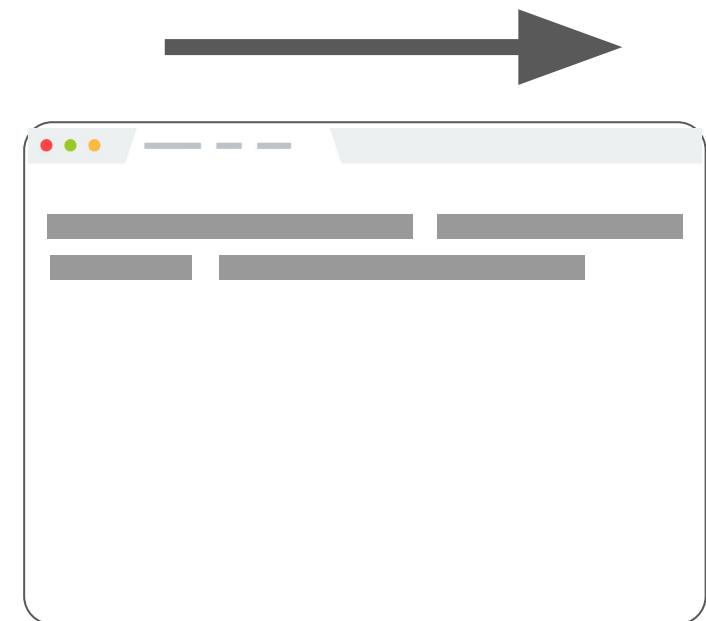


Inline elements

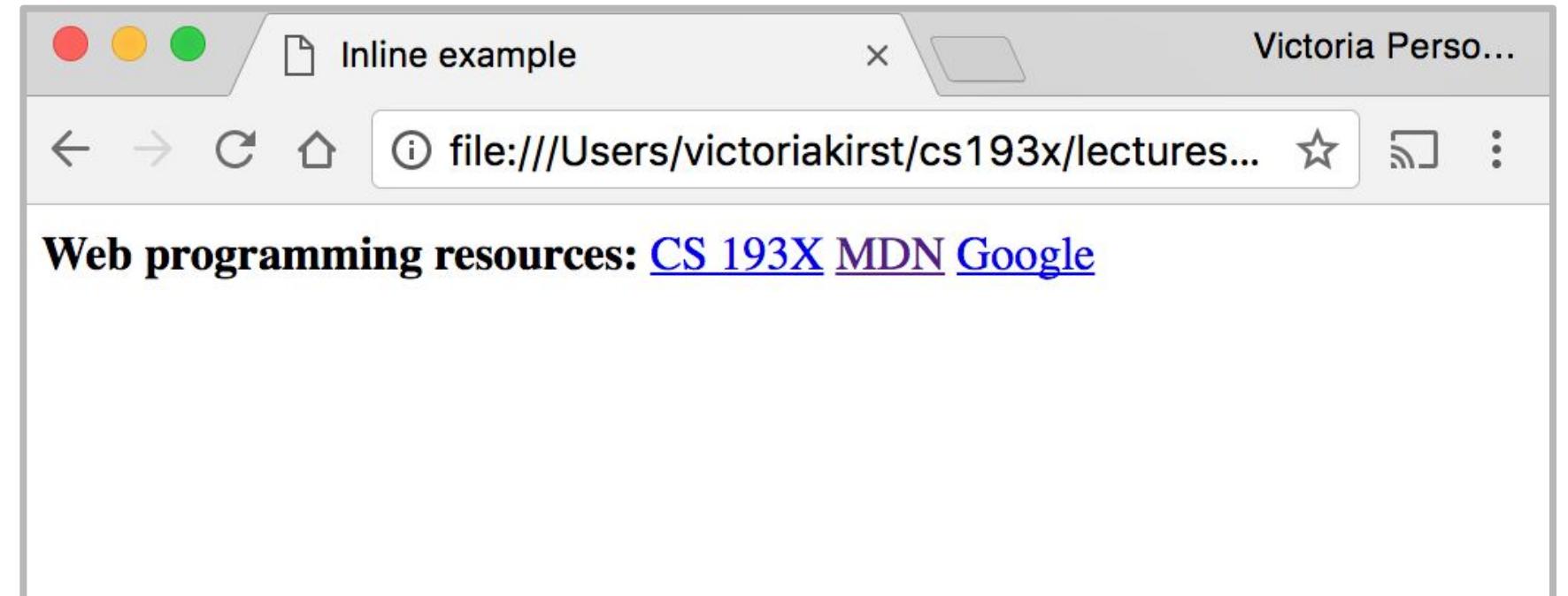
Examples:

<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)
 - o Must position **its containing block element** instead



Example: Inline



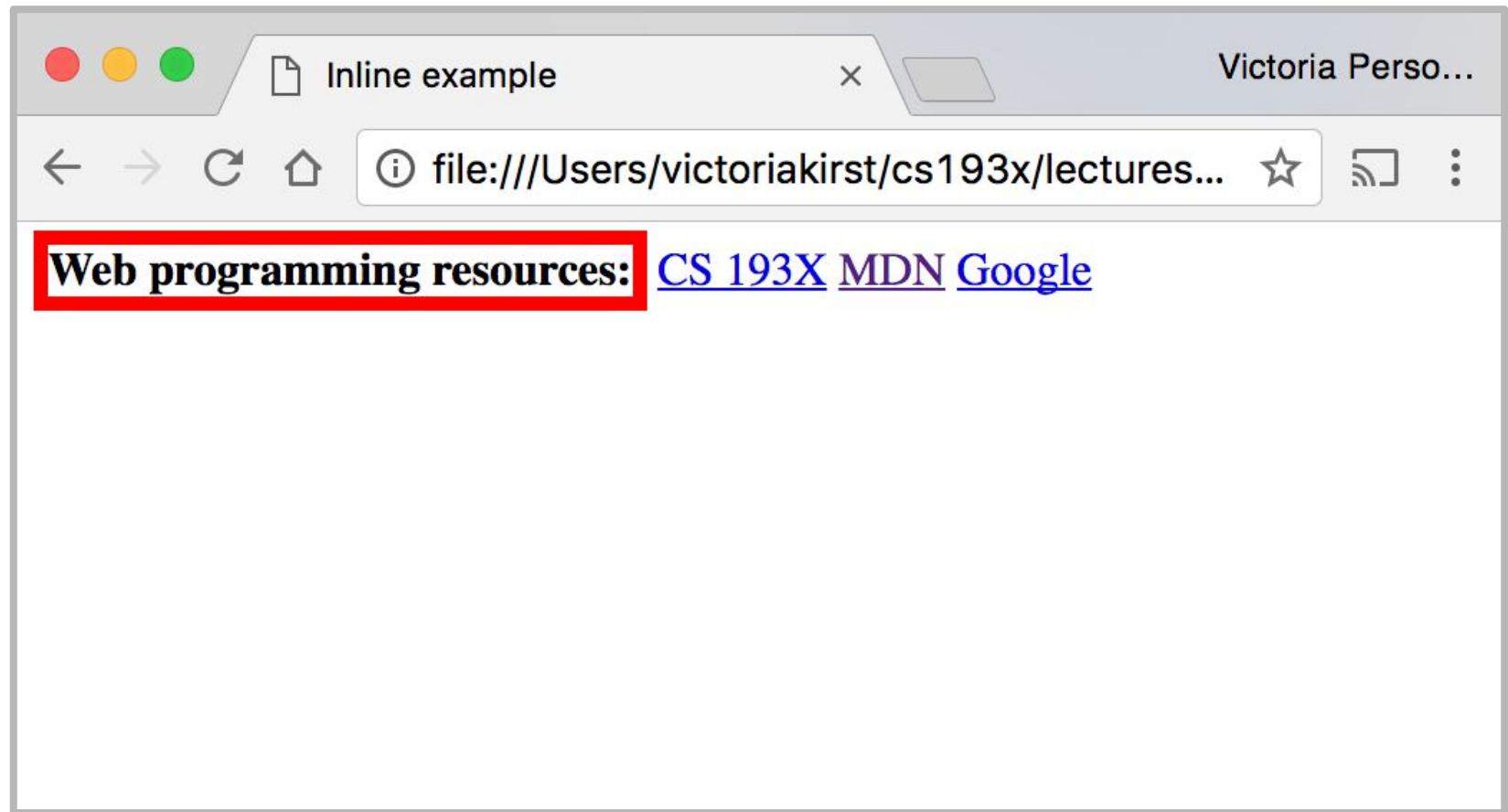
```
<strong>Web programming resources:</strong>
<a href="http://cs193x.stanford.edu">CS 193X</a>
<a href="https://developer.mozilla.org/en-US/">MDN</a>
<a href="http://google.com">Google</a>
```

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>
```

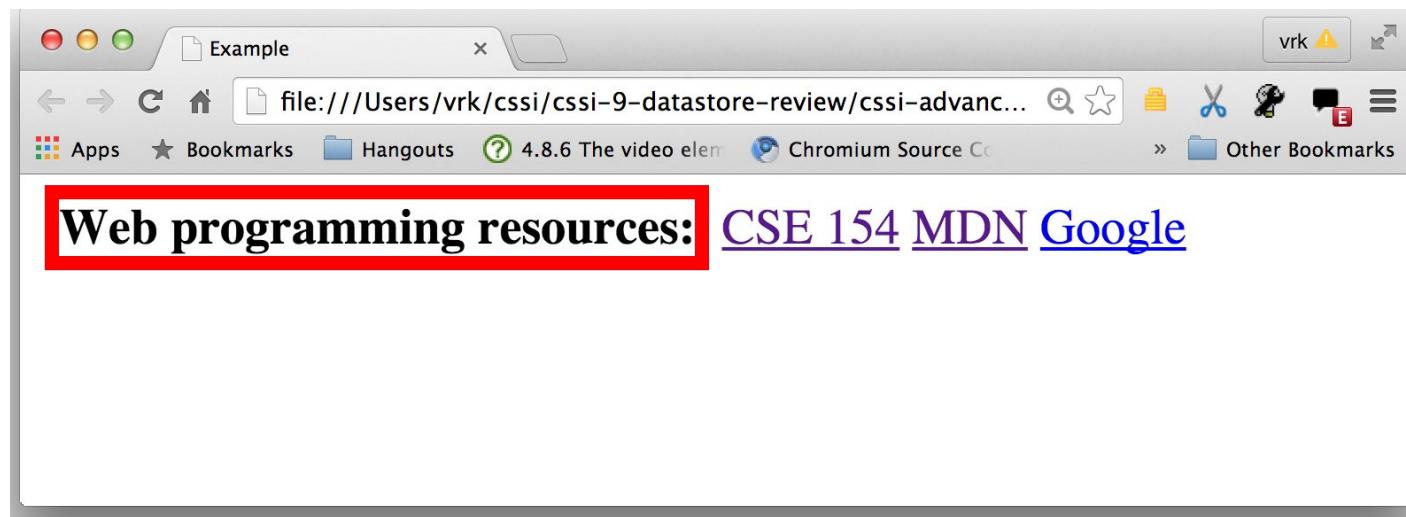


Inline elements ignore width

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">
```

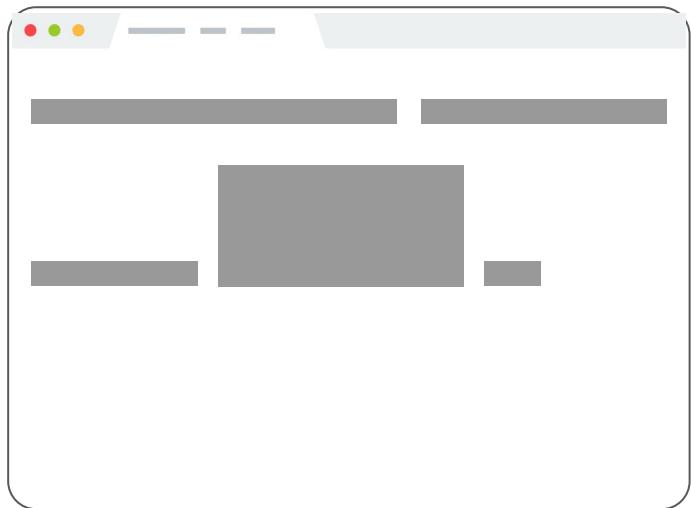


Cannot set **width** on inline element, so it is ignored ([JSBin](#))

inline-block

Examples: , any element with
display: inline-block;

- Take up only as much width 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> =





Inline-block example



Victoria Perso...



file:///Users/victoriakirst/cs193x/lectures...



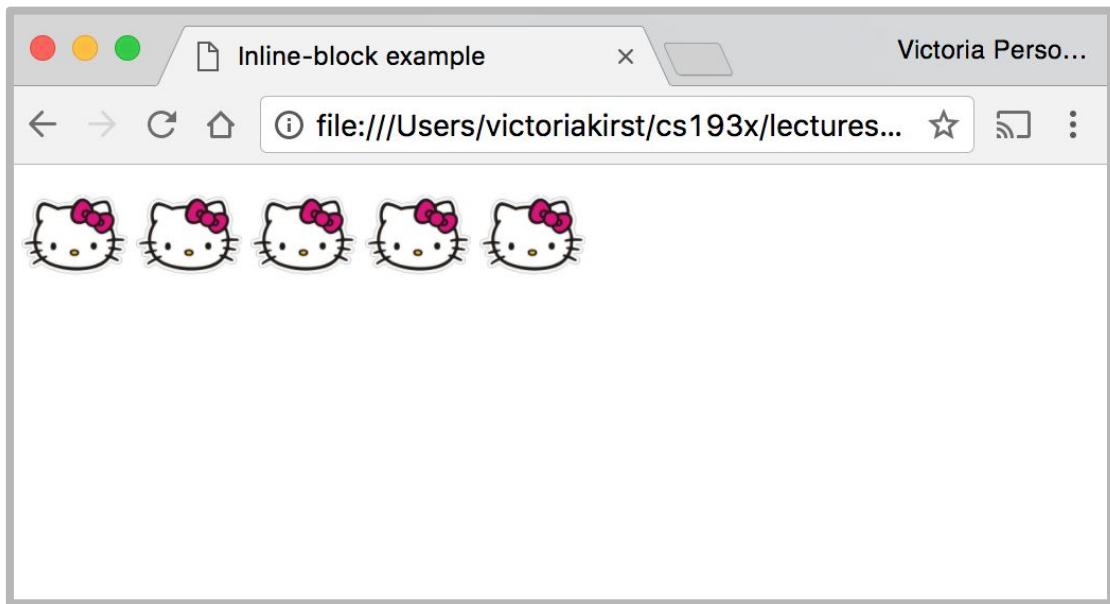
Inline-block

Has width and height; flows left to right

Can set **width** on inline-block element, so image width is set to 50px. ([JSBin](#))

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**
``

Questions?

Moral of the story:

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