

Web Programming Fundamentals

February 2019

Today's schedule

- Syllabus
- Course Info
- Browsers! The Internet!
- A little bit about HTML and CSS
- Homework 0 assigned

Check out the course website for all this and more:

<https://webdevcg.herokuapp.com/>

Sponsors



Kumasi Hive



**Creativity Group
UENR**



CSITSA

Syllabus

What is WPF?

Web Programming "Fundamentals"

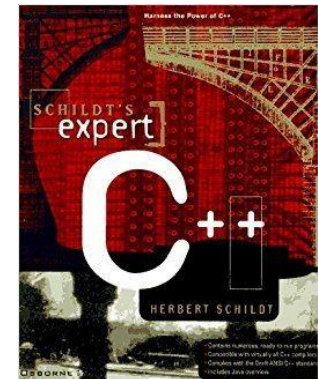
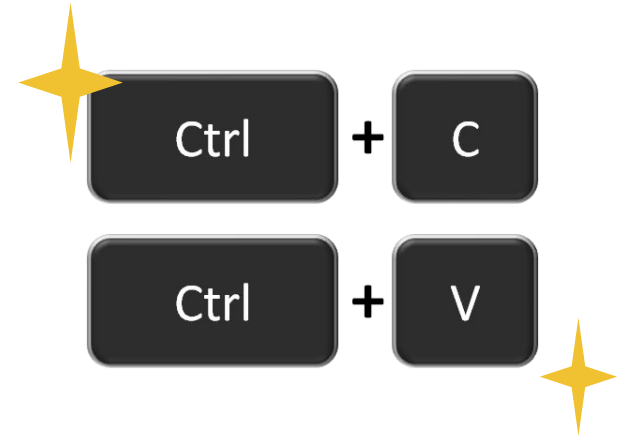
- An introduction to web programming

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;  
}
```

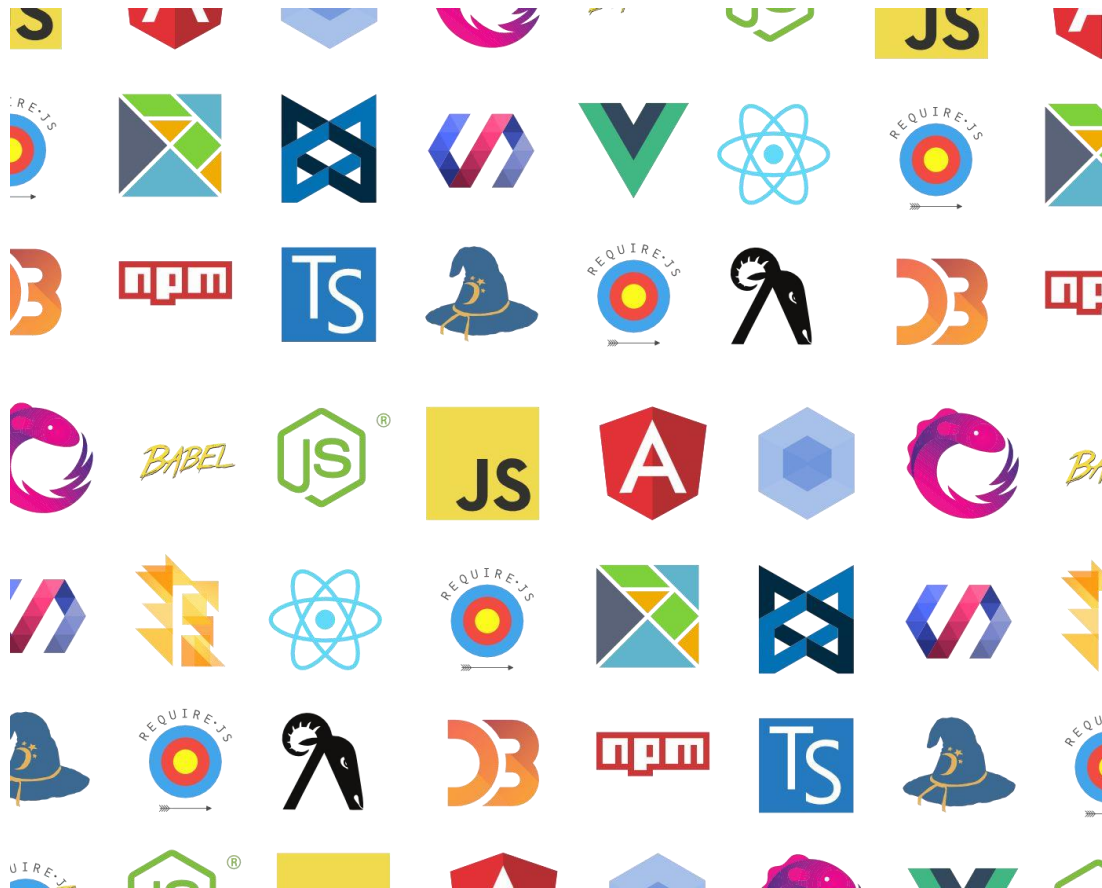


But then when you try to write CSS, literally nothing works:

CSS IS AWESOME

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 this course about?

Opinionated:

- There are many ways to do things on the web: can't learn them all at once!
- 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

Goals

If you never take another web programming class again, you will leave this course 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)

in detail

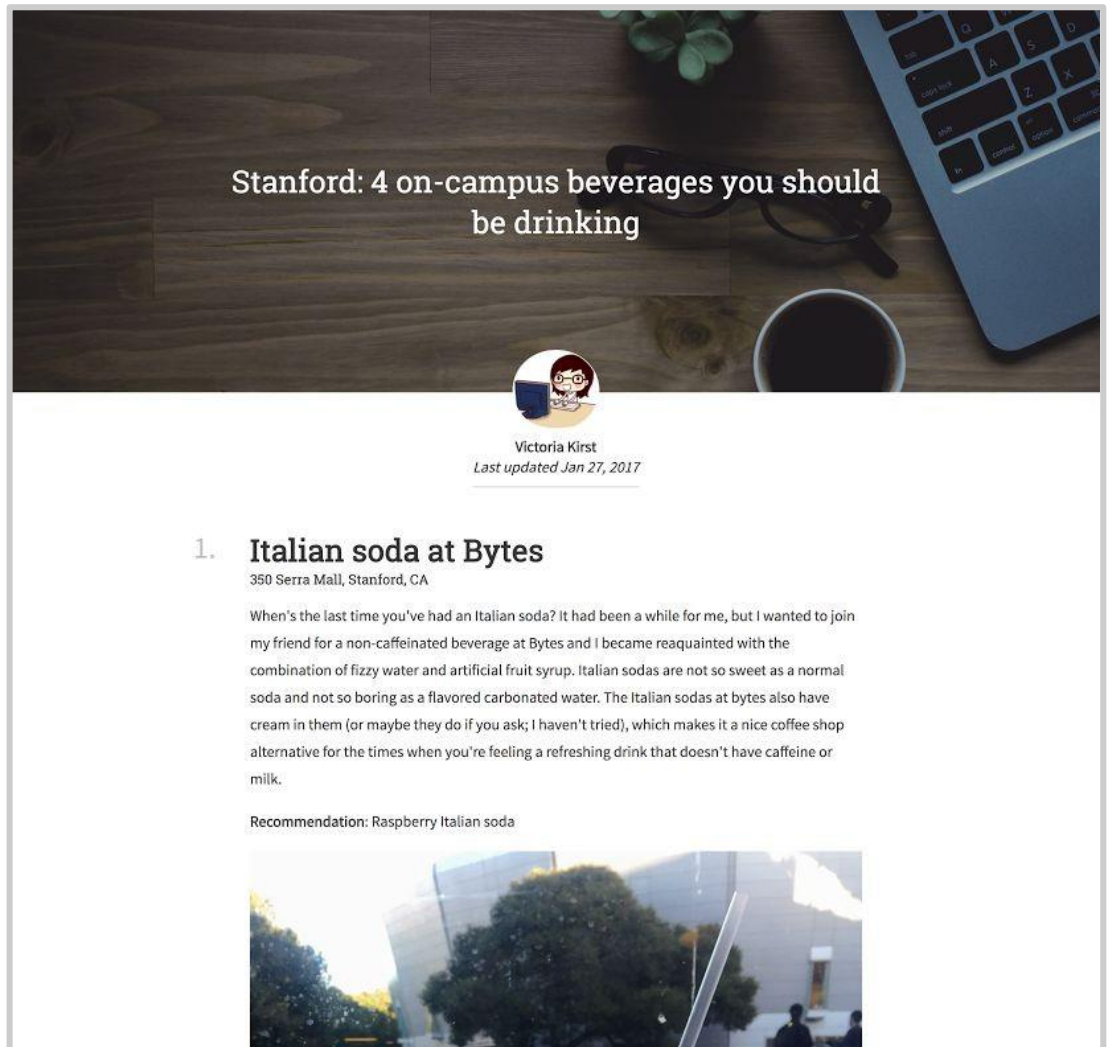
- HTML
- CSS
- JavaScript
- Backend basics
 - Server on NodeJS + Express
 - Database via MongoDB and Mongoose

(Uh...)

CSS, applied

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

(Note: HW1 is not released yet.)



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**

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');  
})());
```

Modern JS / ES6+

JavaScript (~5 weeks)

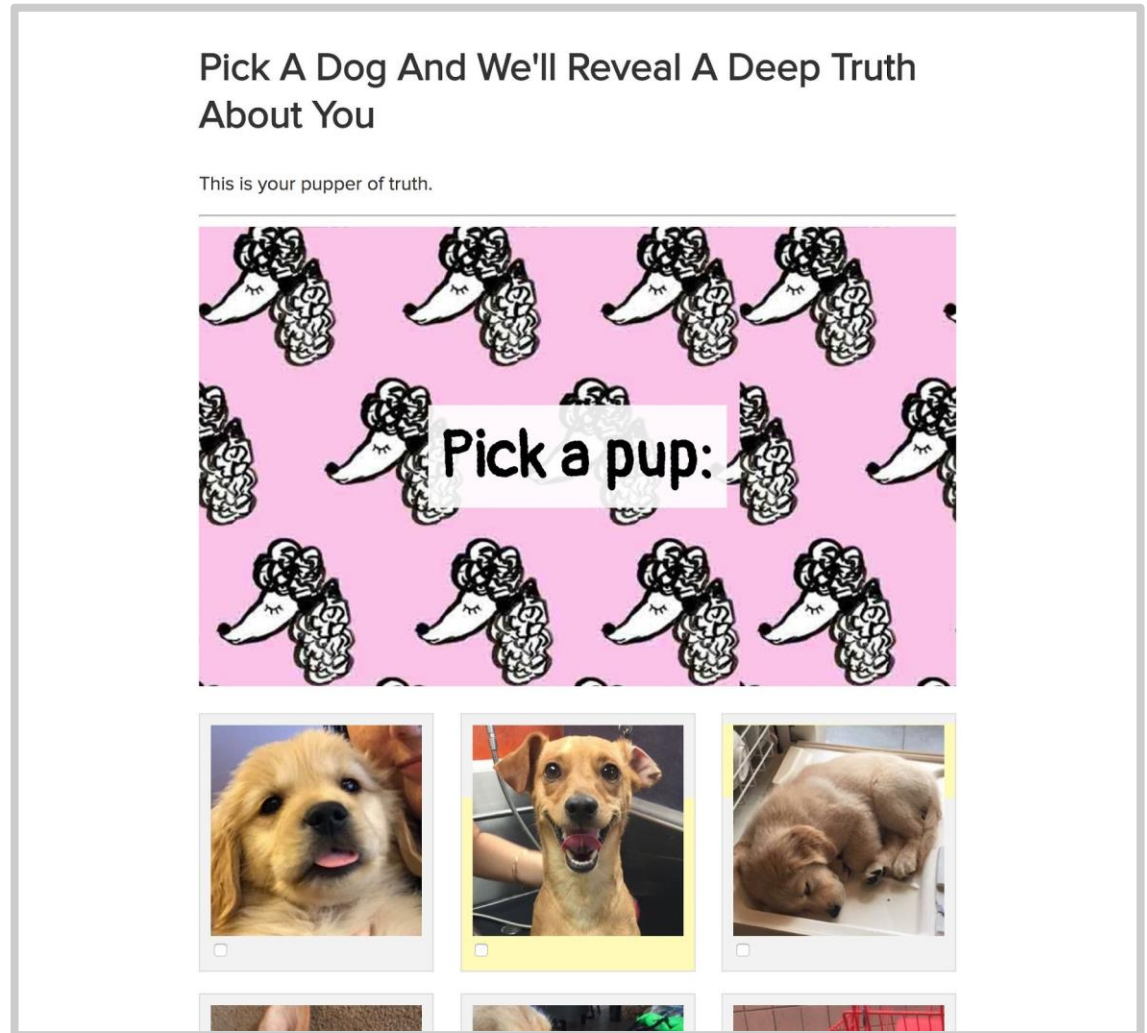
- JavaScript classes
- Relevant functional programming
 - Lambdas
 - Generator functions and async/await
 - "Fat arrow" vs function
 - Closures
- Creating and using Promises
- Understanding the Event Loop
- Modules and encapsulation

NO frontend framework; minimal libraries

No Angular/React/JQuery/etc

JavaScript, applied

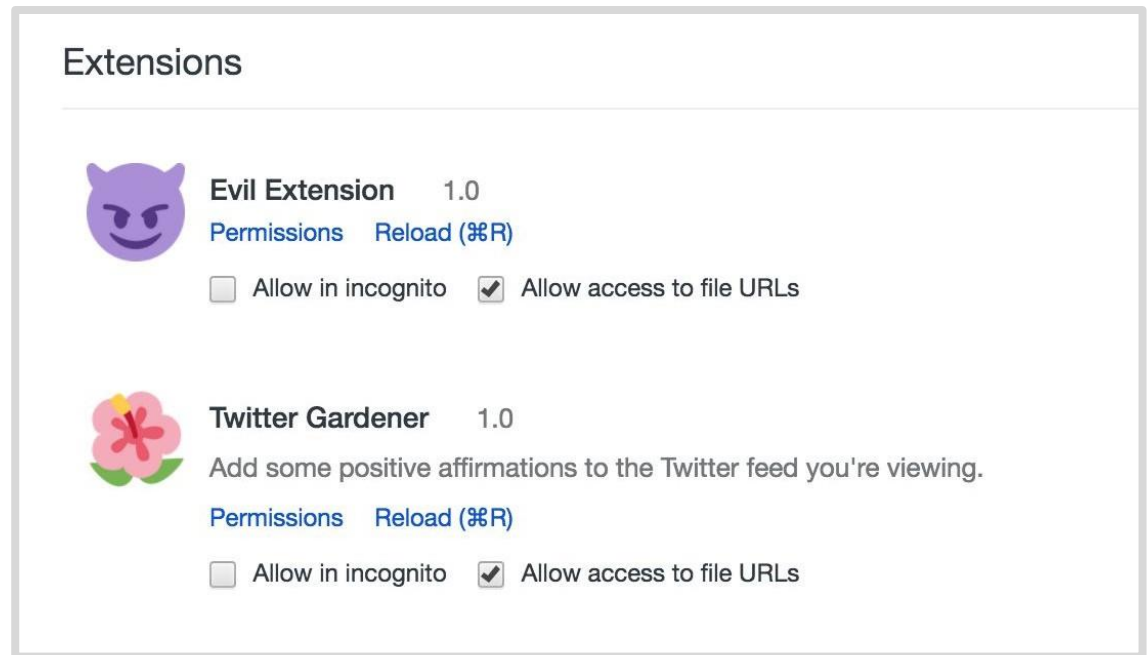
HW2 will ask you to make a webpage that looks like this-ish:



JavaScript, applied

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

(Note: HW2 is also not released yet.)



Baby's first backend

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)

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
- **Individual** assignments; pairing allowed

1 final project

- Open-ended! Details to come.
- ~1 week in scope
- **Individual** project; groups allowed



Course info

Disclaimer

This is the second ever offering of this course, 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.

Browser and Text editor/IDE

- **Text editor:** You can use whatever you want. We recommend [Visual studio code](#).
- **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

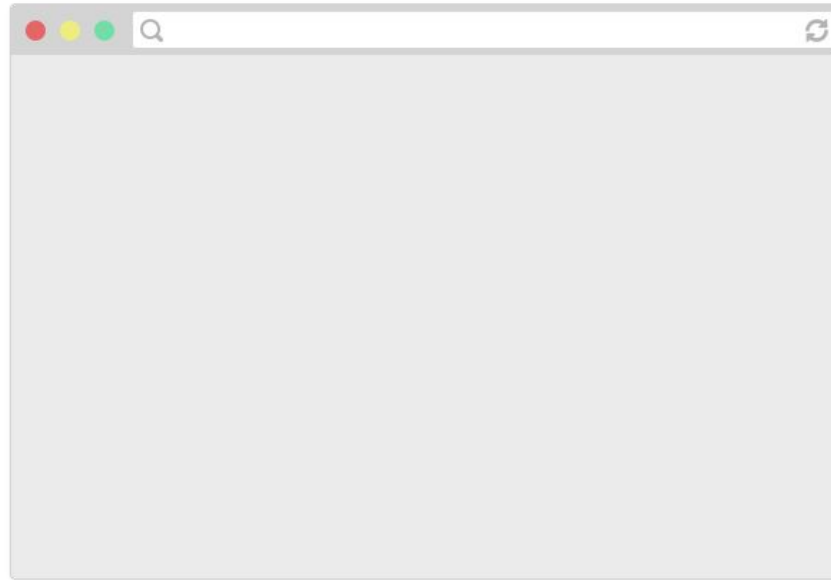
Questions?

Today's schedule

- ~~- Syllabus~~
- ~~- Course Info~~
- Browsers! The Internet!
- A little bit about HTML and CSS

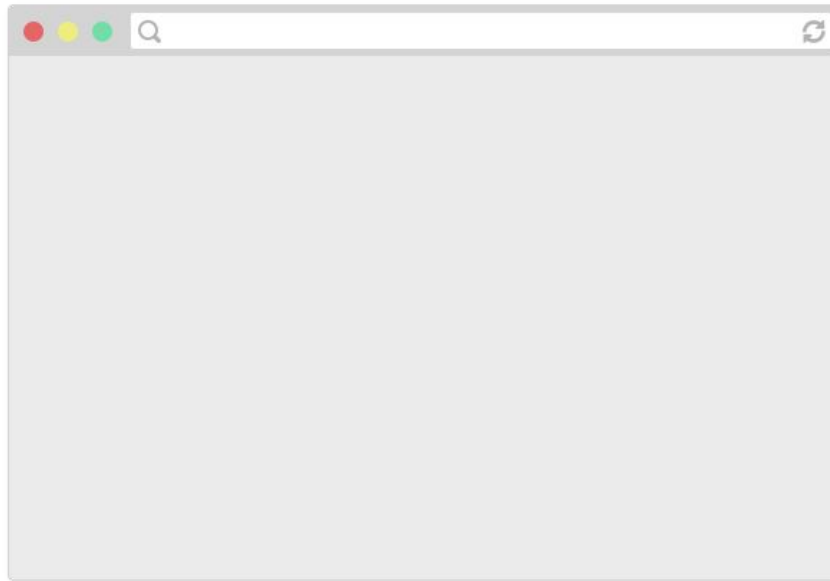
Browsers!
The Internet!
The web!

How do web pages work?



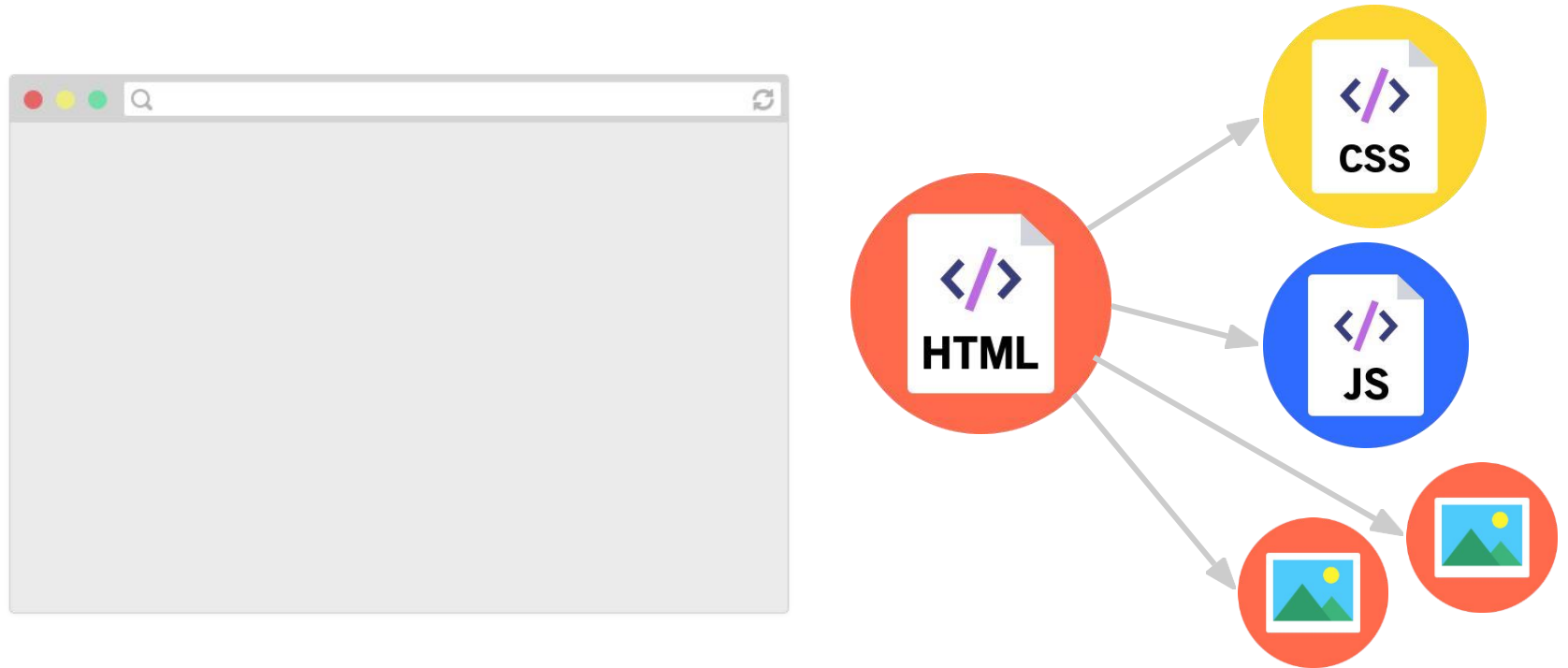
Browsers are applications that can display web pages.
E.g. Chrome, Firefox, Safari, Internet Explorer, Edge, etc.

How do web pages work?



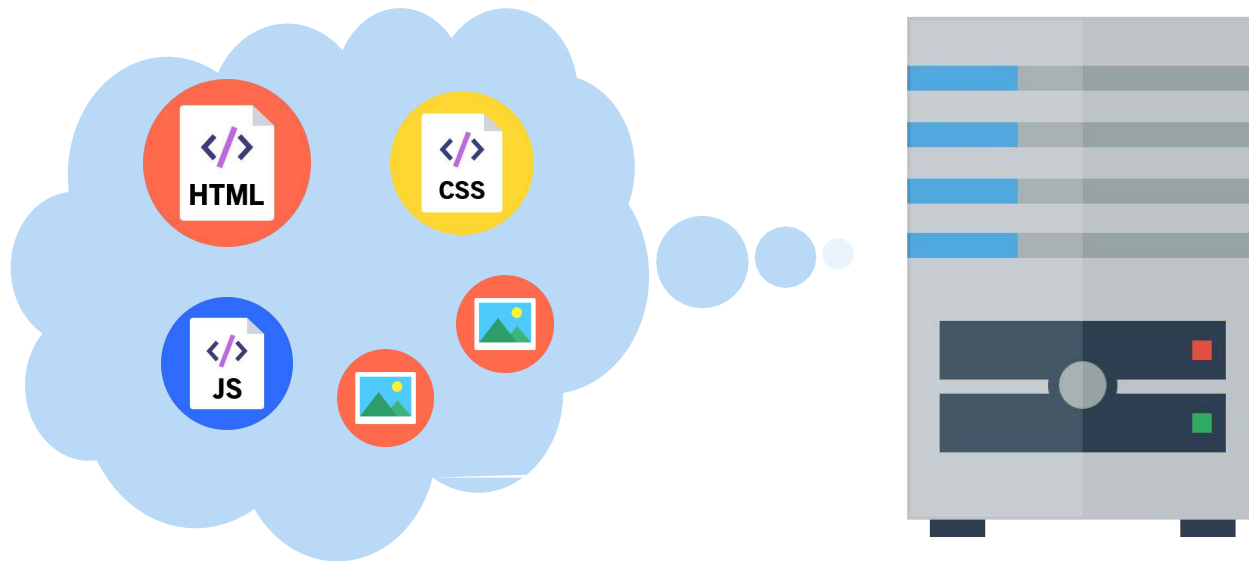
Web pages are written in a markup language called **HTML**, so browsers display a web page by reading and interpreting its HTML.

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?

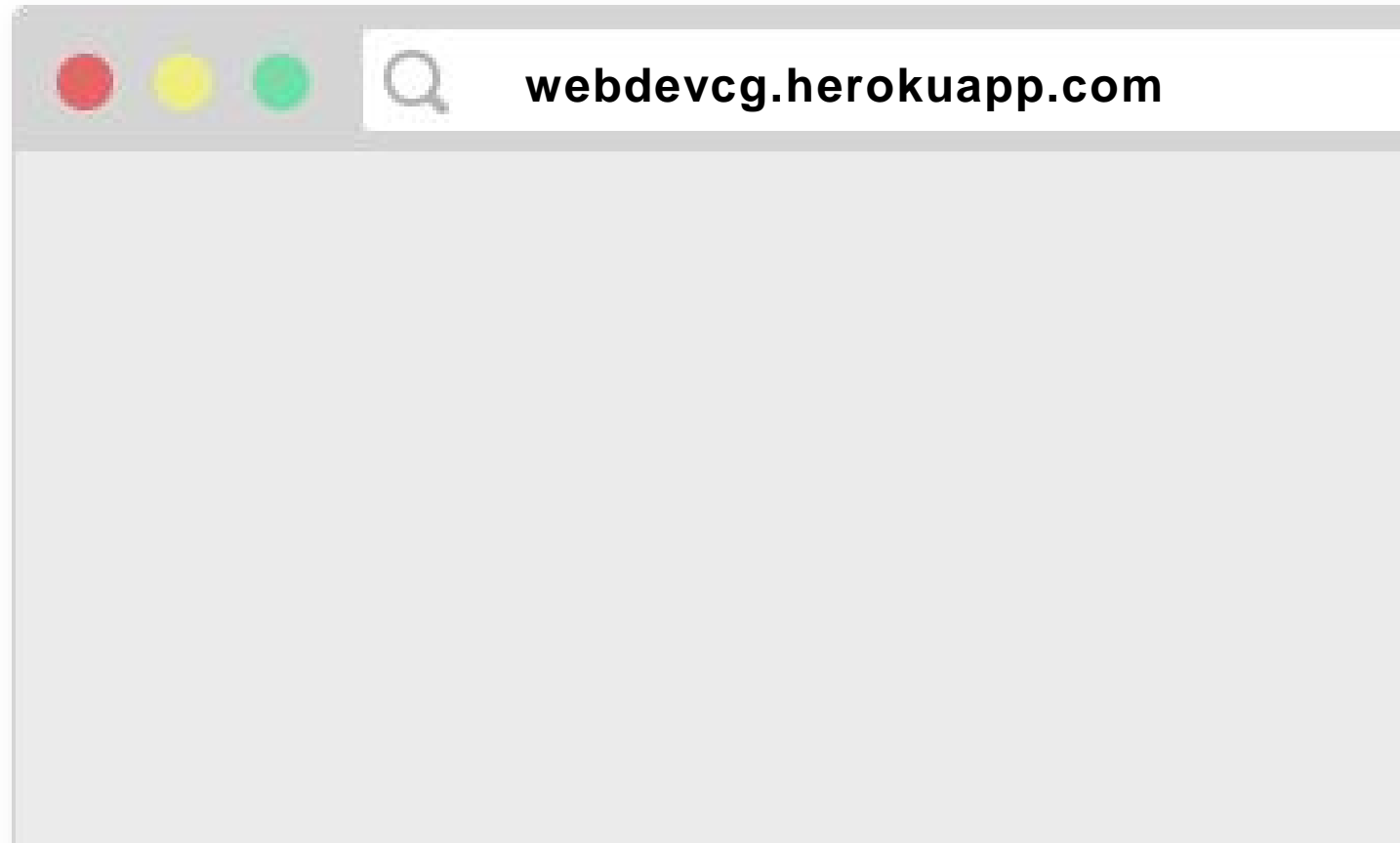


A **web server** is a program running on a computer that delivers web pages in response to requests.

It either stores or generates the web page returned.

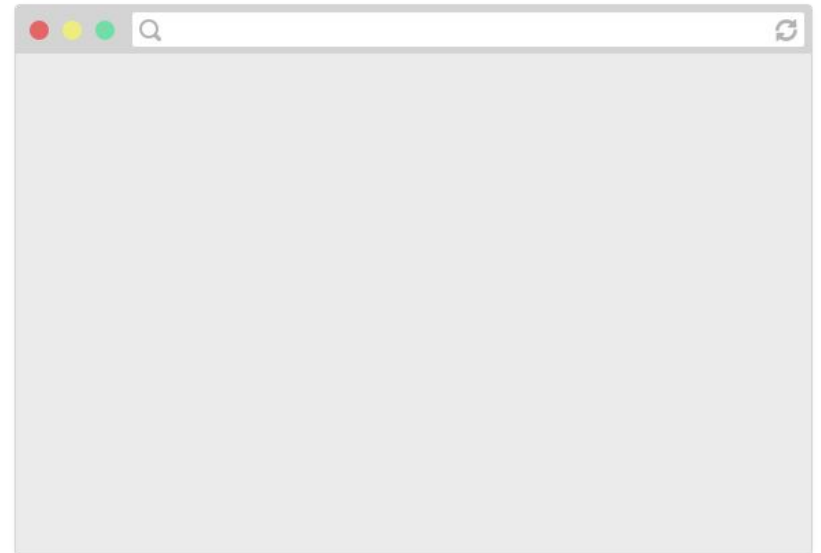
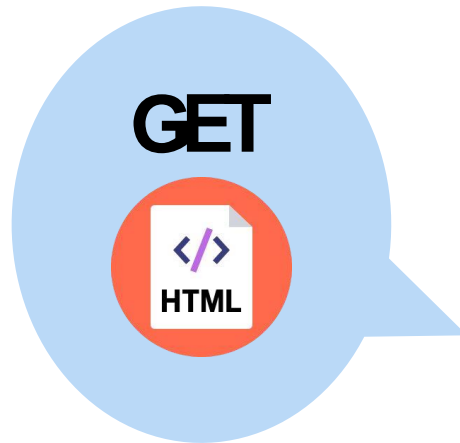
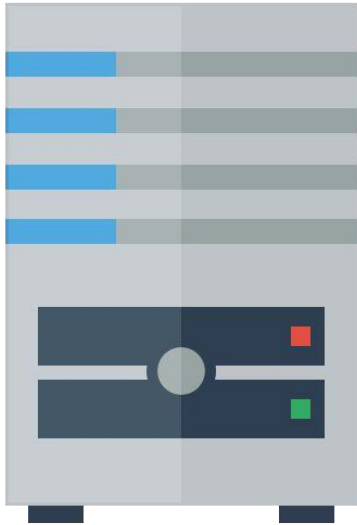
How do web pages work?

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

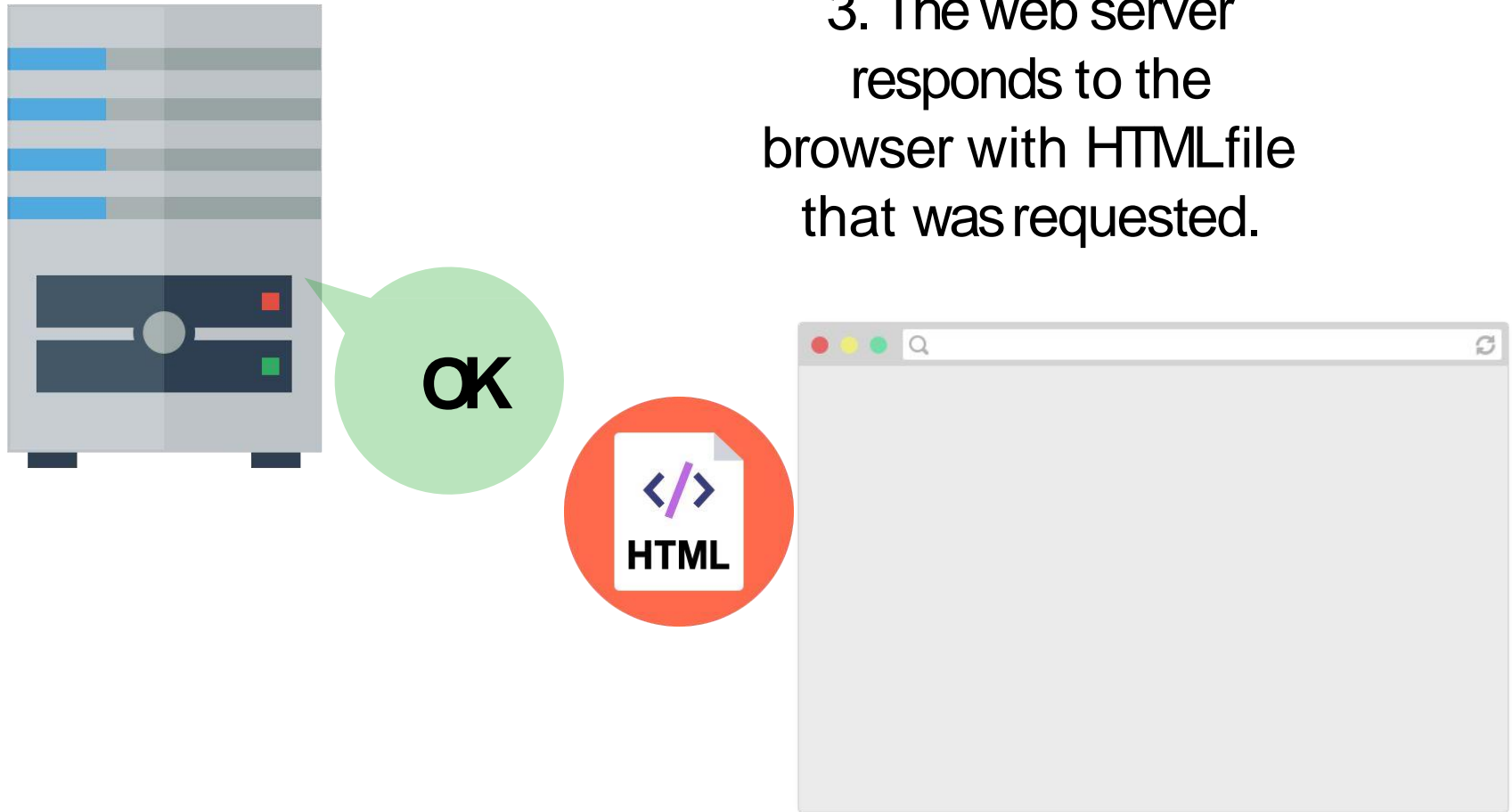


How do web pages work?

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

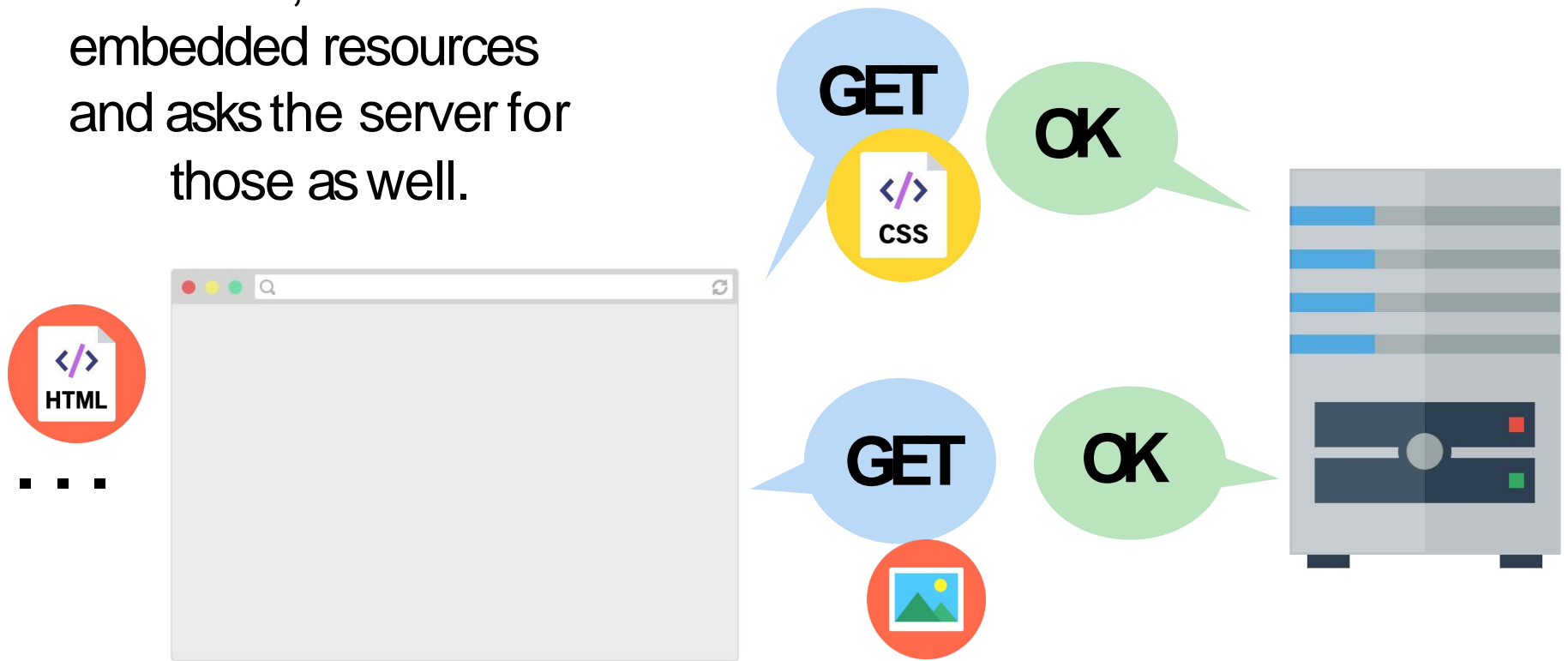


How do web pages work?



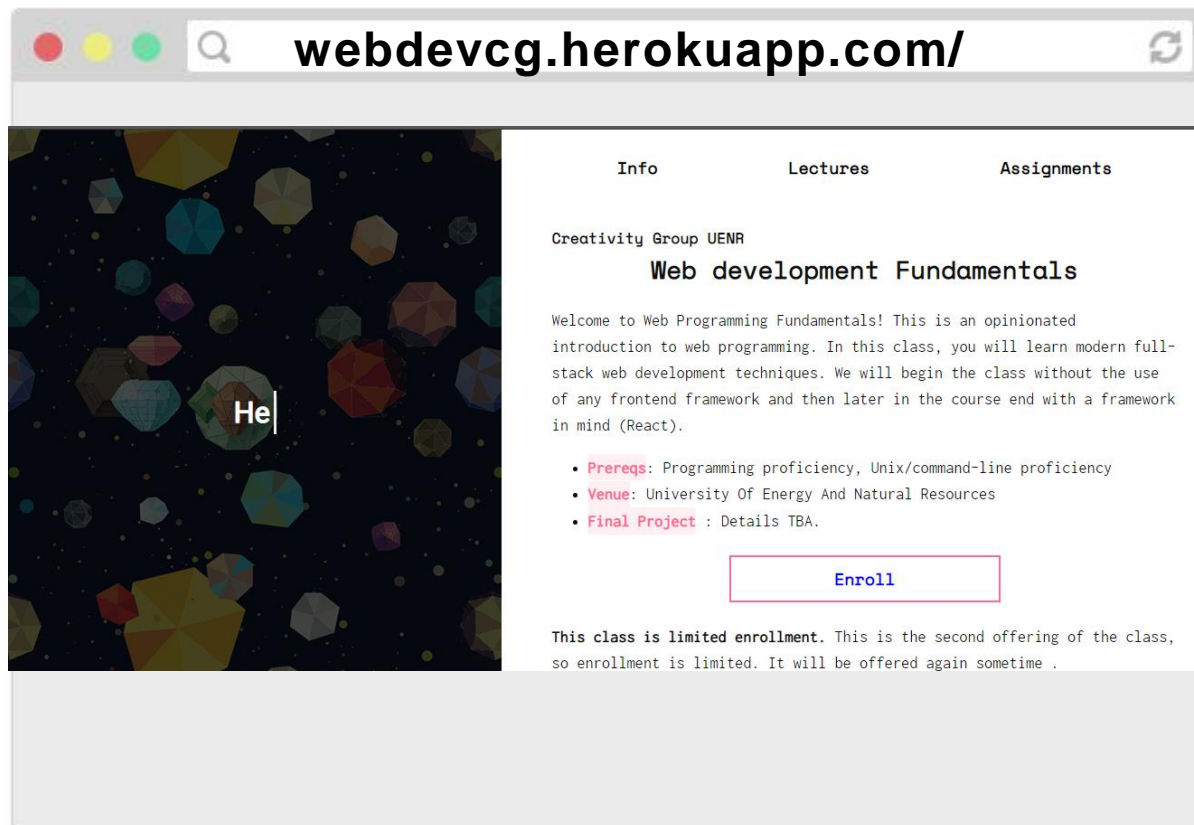
How do web pages work?

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



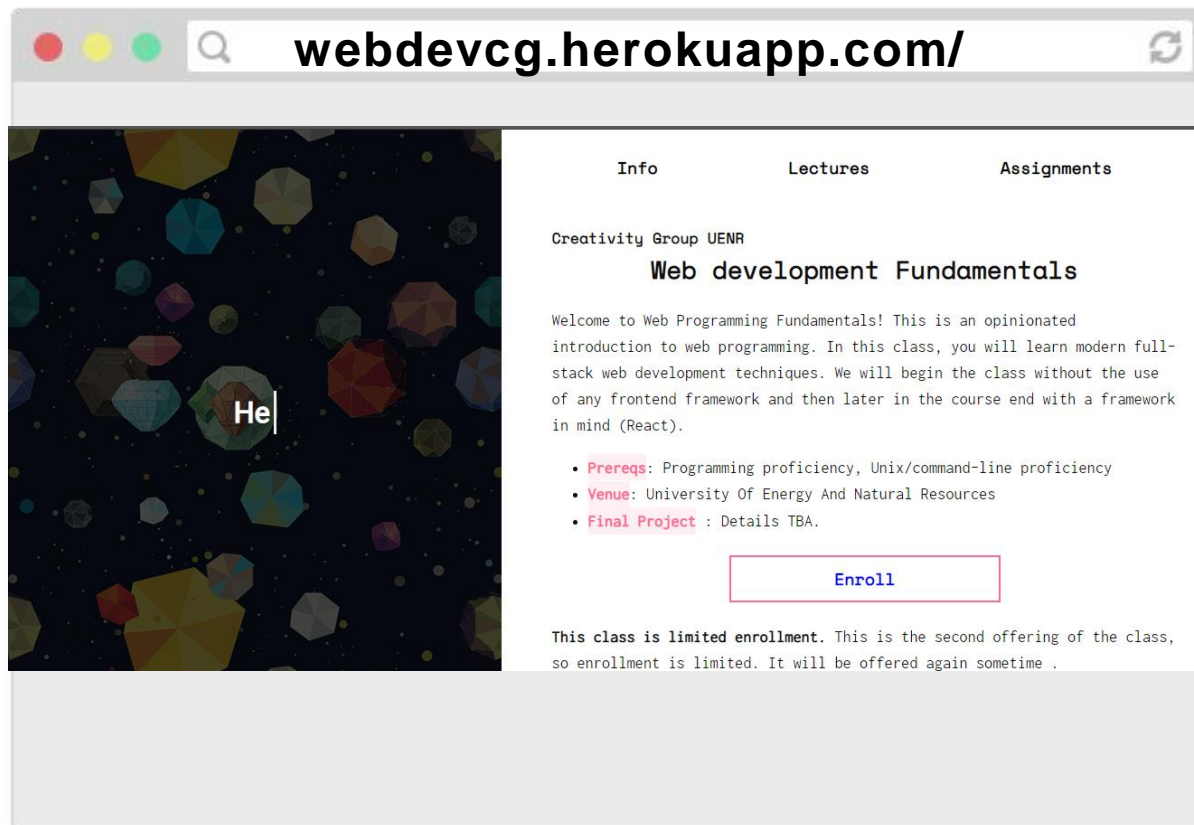
How do web pages work?

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.)



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 awesome!!!

</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)



HTML elements

<p>

HTML is awesome!!!

</p>

- An element usually has start and ending tags (<p> and </p>)
 - **content**: stuff in between start and endtags
- 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**)

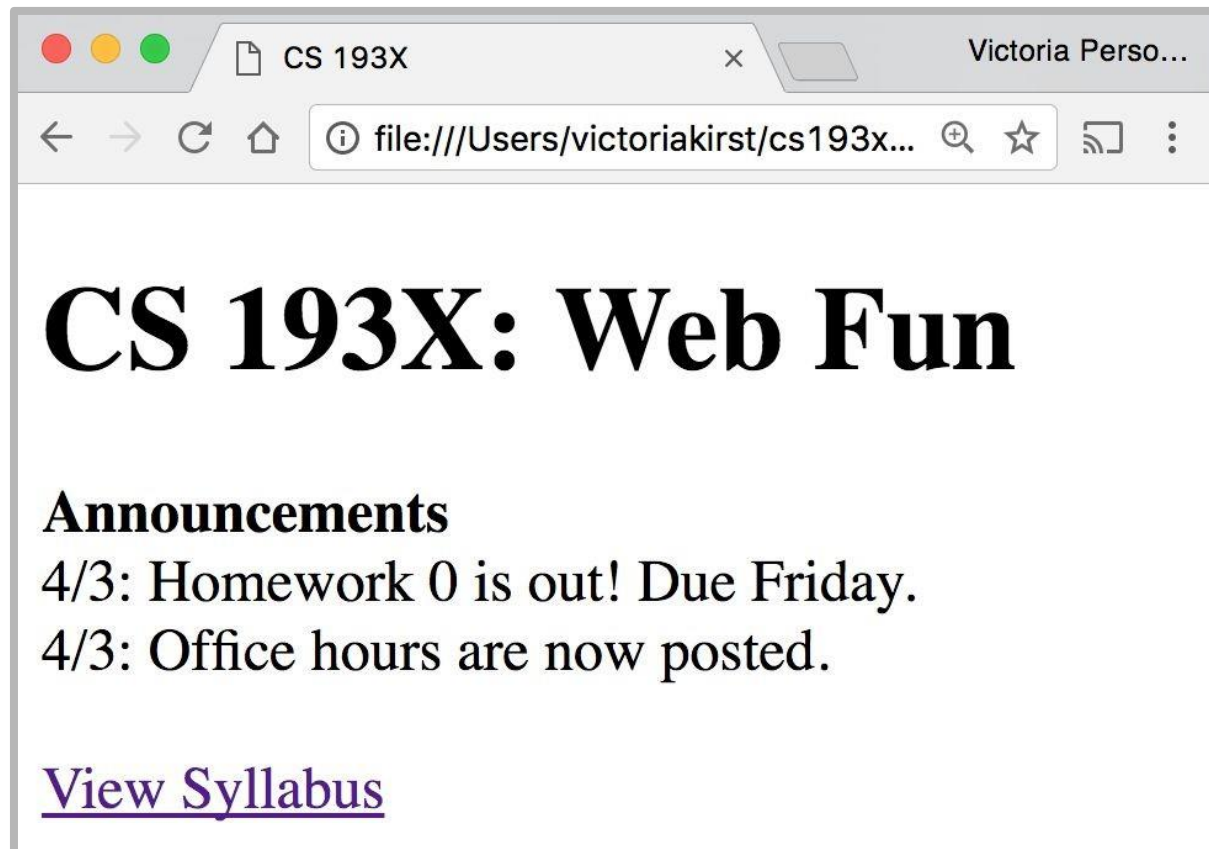
Some HTML elements

(to place within `<body>`)

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

Exercise: Course web page

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



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 **CSSrules**, 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;
```

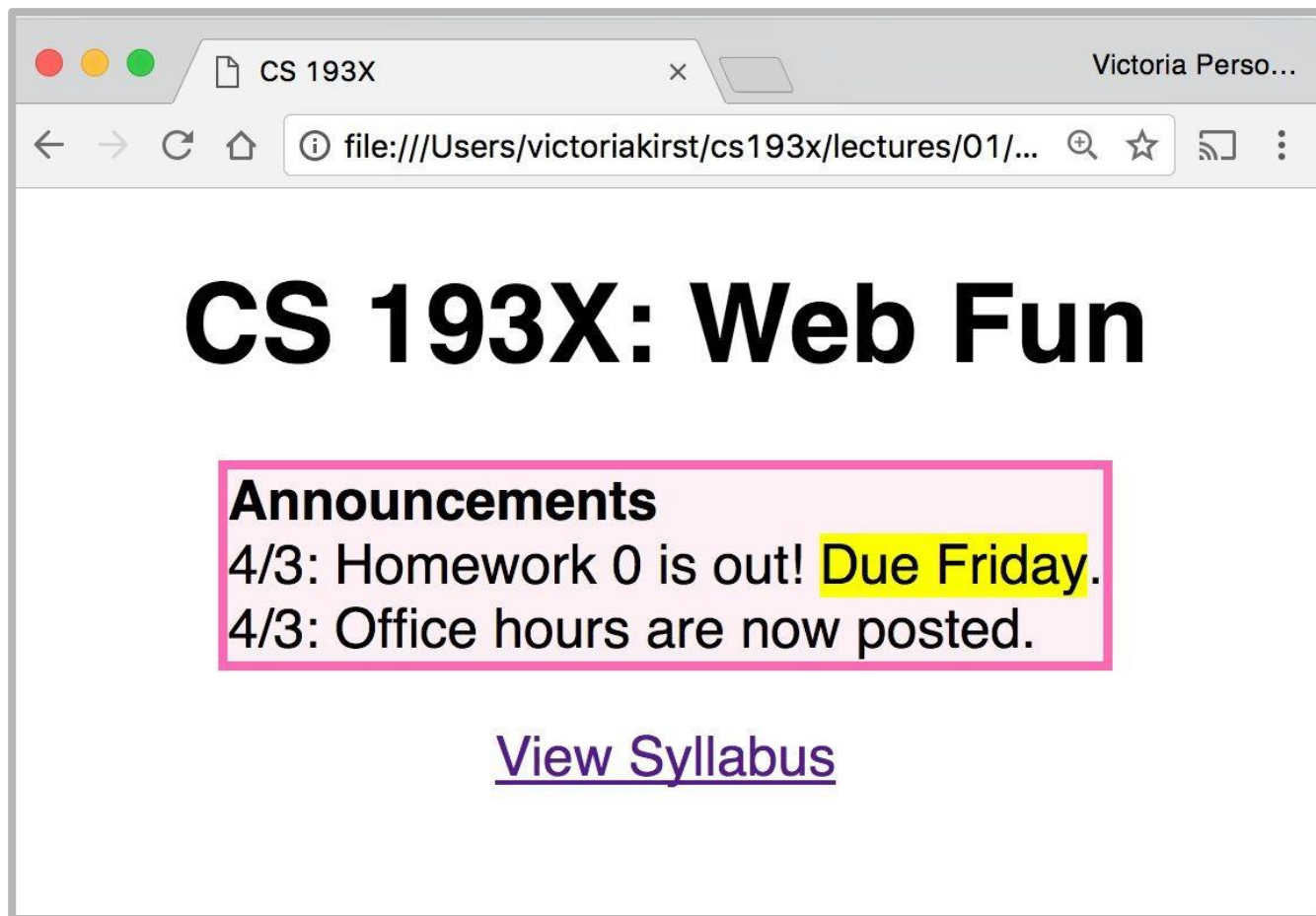
```
color: #0f0;
```

```
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:



Exercise: Course web page

Let's write some CSS to style our page:

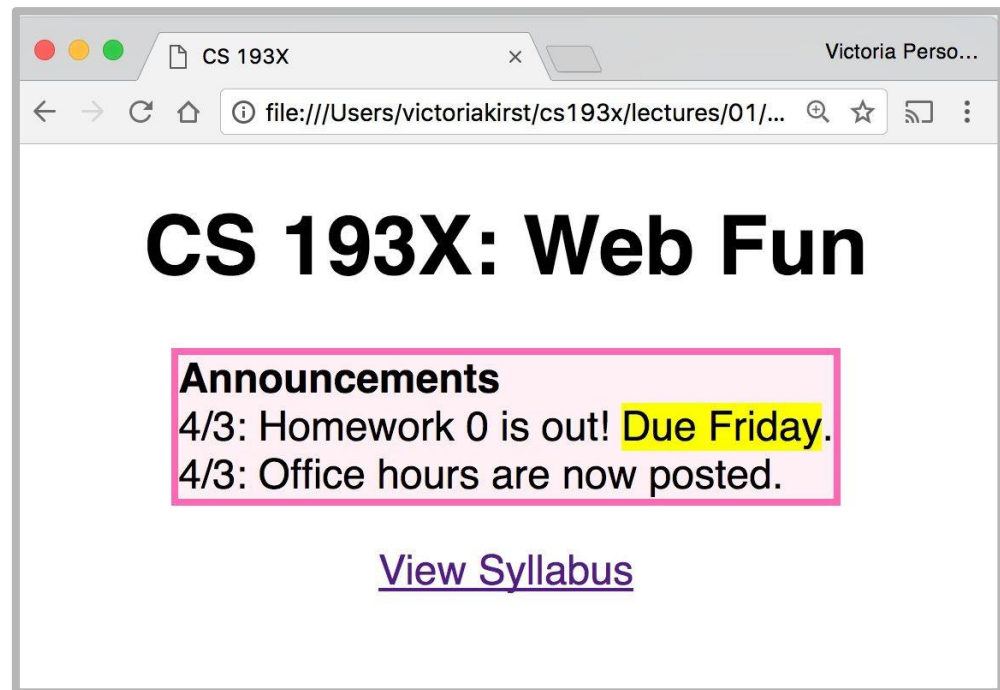
Font face: Helvetica

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: There is one crucial set of rules we haven't learned yet...

block vs inline display

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 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:

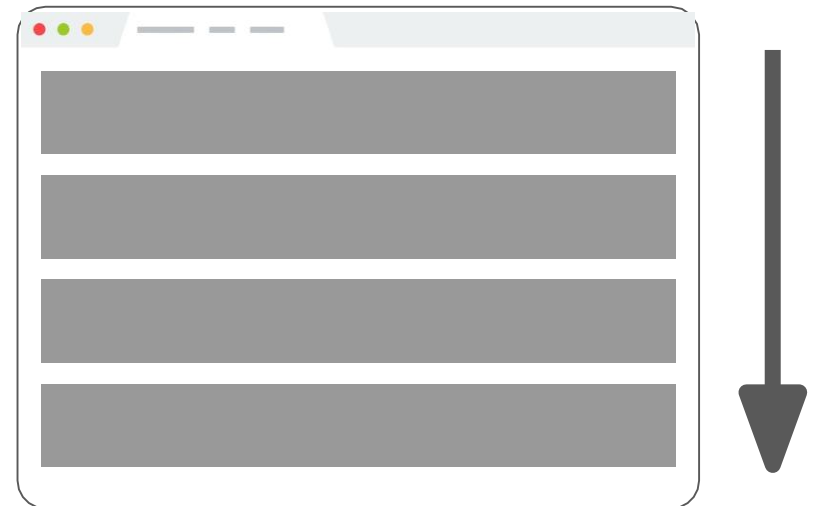
1. **block**: large blocks of content, has height and width
`<p>`, `<h1>`, `<blockquote>`, ``, ``, `<table>`
2. **inline**: small amount of content, no height or width
`<a>`, ``, ``, `
`
 - a. **inline block**: inline content with height and width
``
3. **metadata**: information about the page, usually 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>
```



Block example



Victoria Perso...



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



About vrk

She likes *puppies*

Block-level:

extends the full width of the page

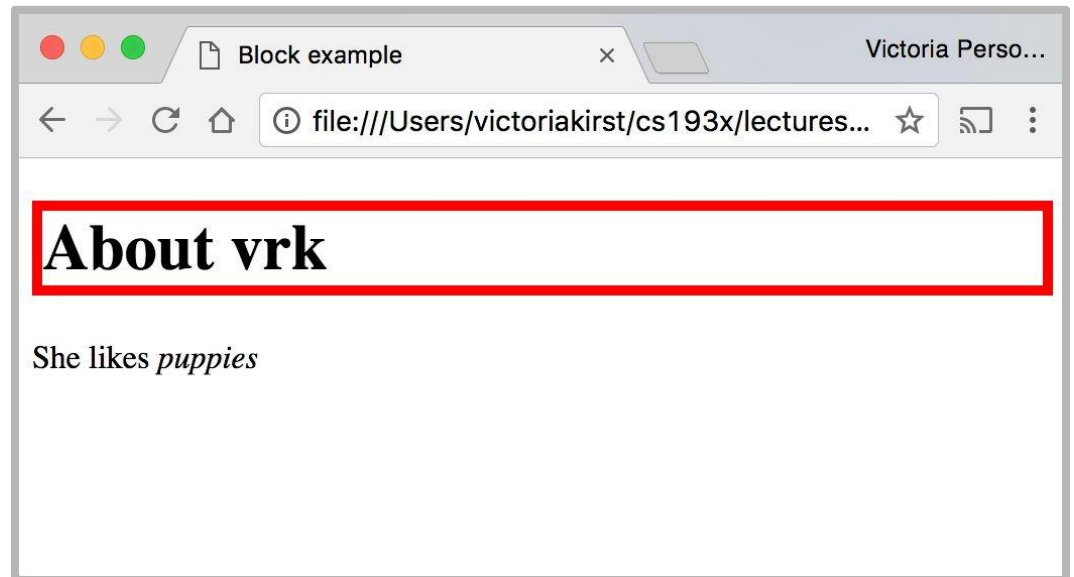
```
h1 {  
  border: 5px solid red;  
}
```

```
<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

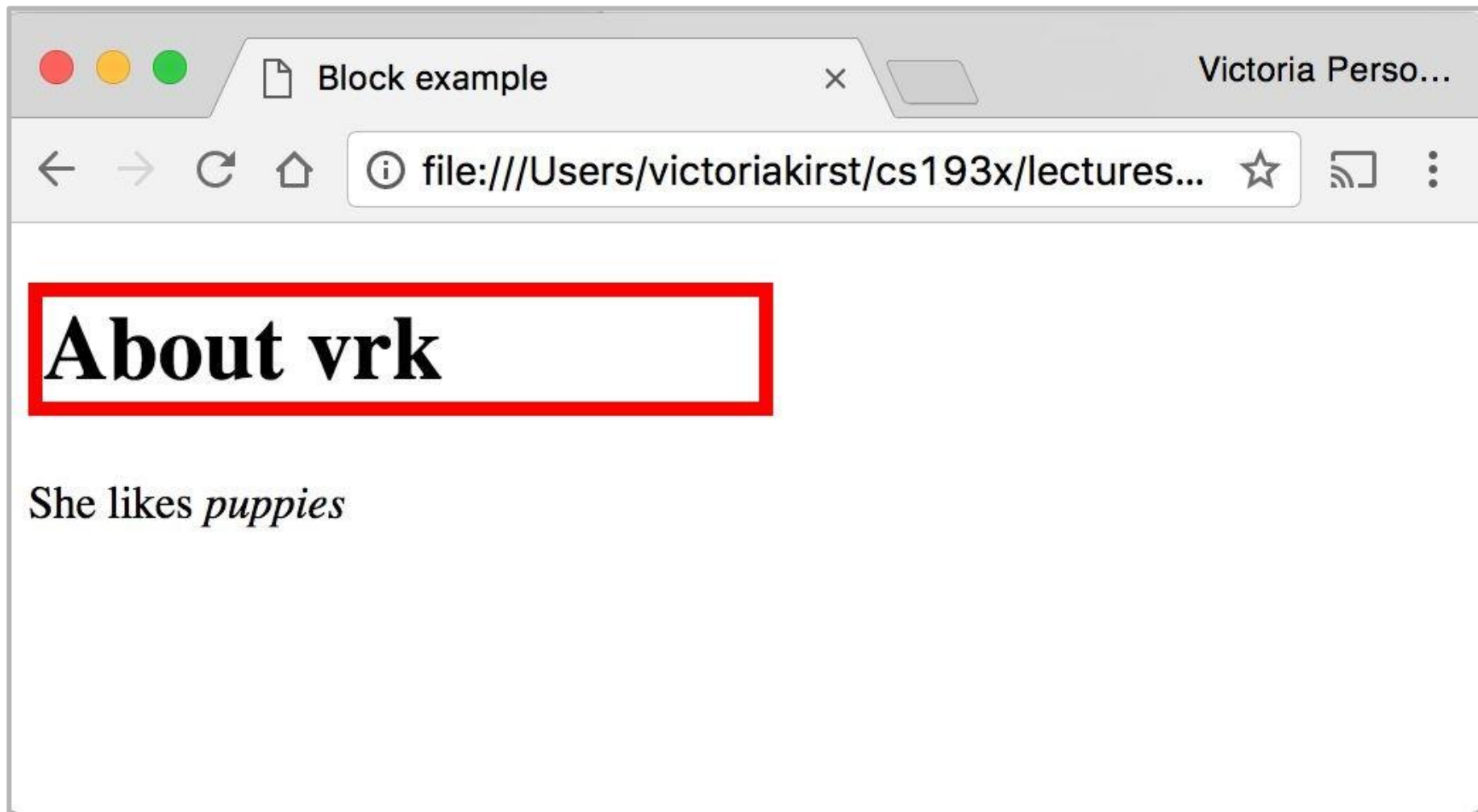
See: [JSBin](#)



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

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





Block-level

width can be modified

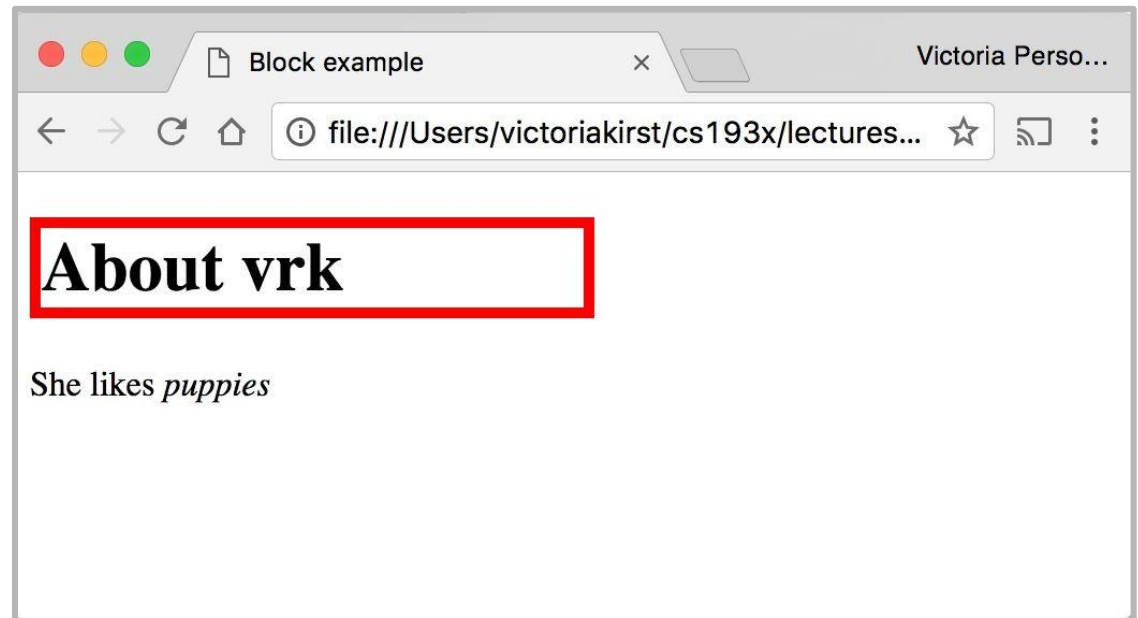
```
h1 {  
  border: 5px solid red;  
  width: 50%;  
  ;  
}
```

```
<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](#)

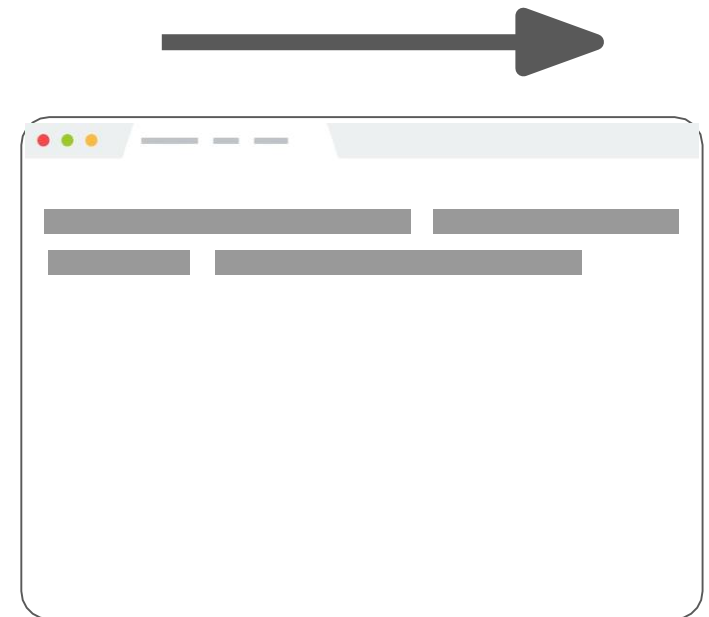


Inline elements

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



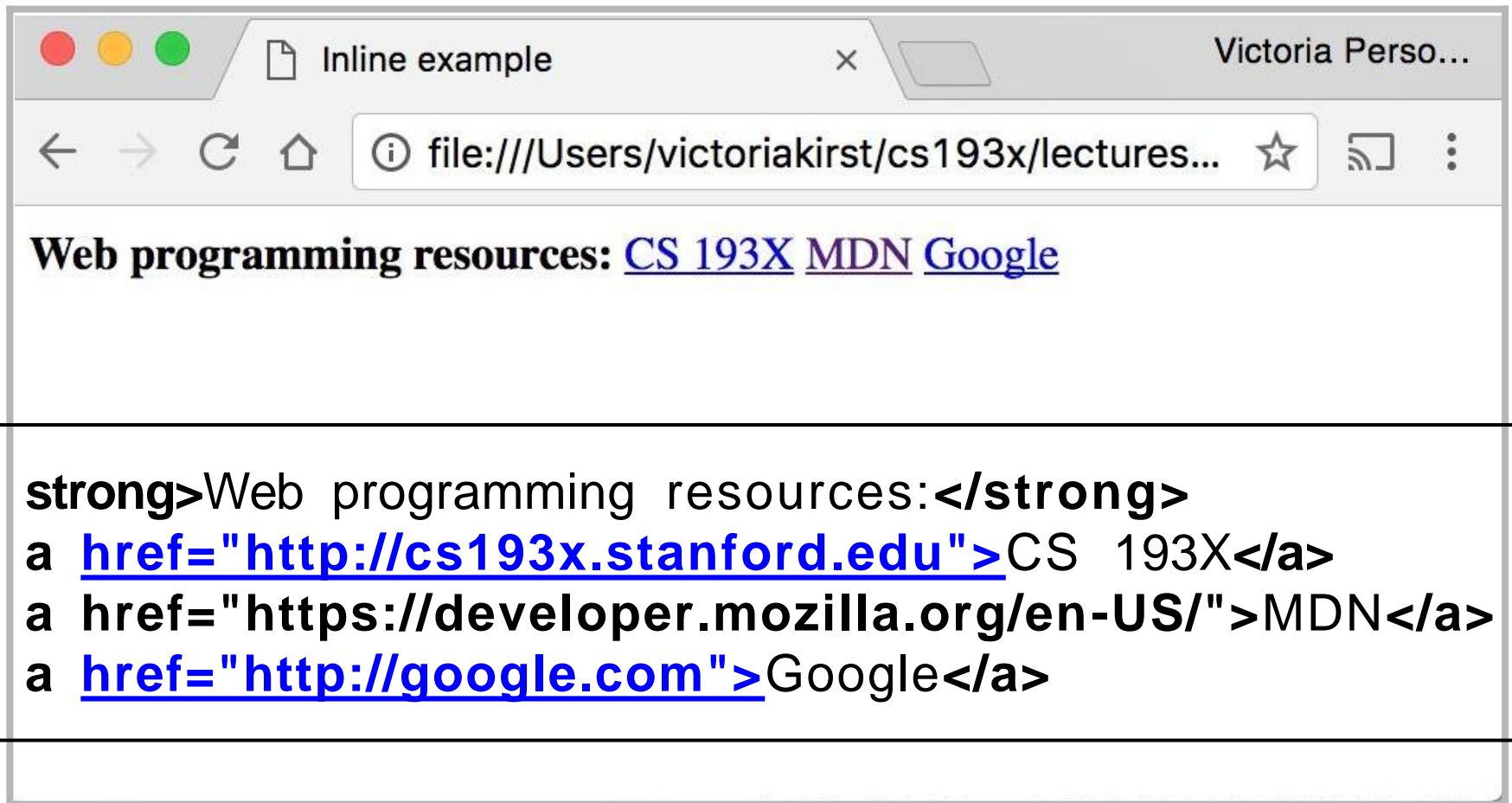
Example: Inline



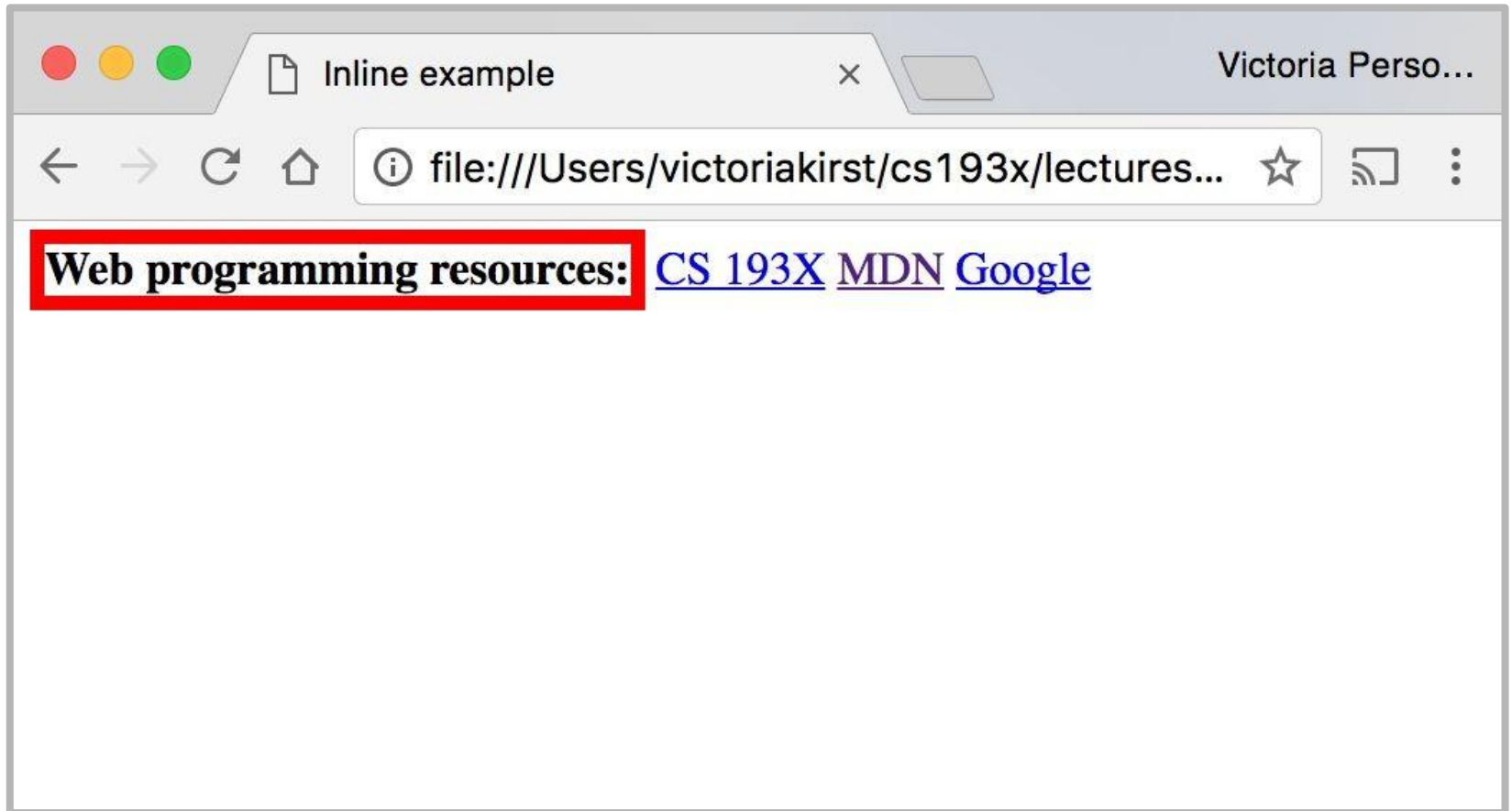
```
<strong>Web programming resources:</strong>  
<a href="http://cs193x.stanford.edu">CS 193X</a>  
<a href="https://developer.mozilla.org/en-US/">MDN</a>  
<a href="http://google.com">Google</a>
```

**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>
```

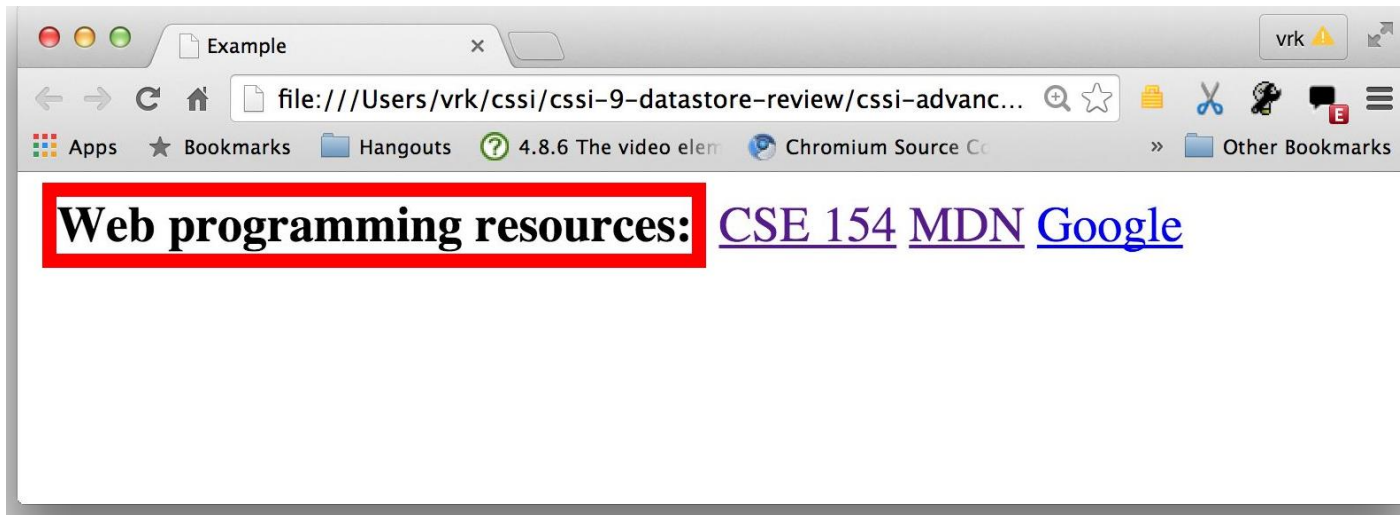


Inline elements ignore width

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">
```

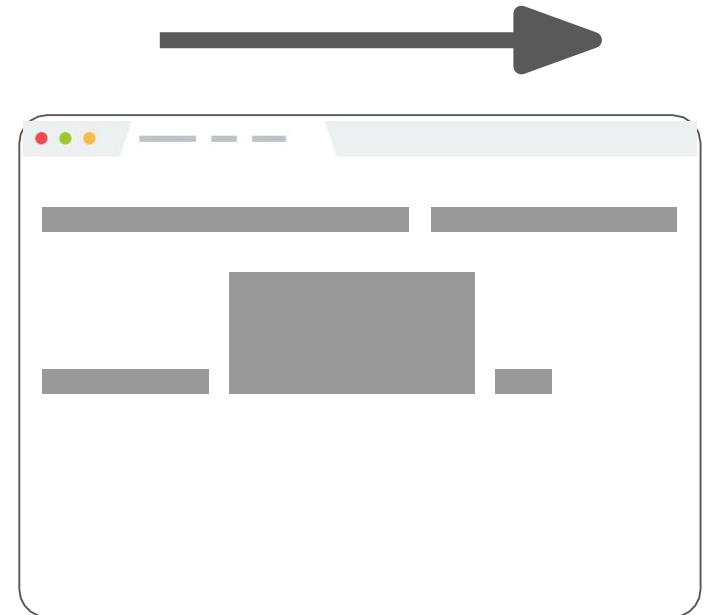


Cannot set **width** on inline element, so it is ignored ([JSBin](#))

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...

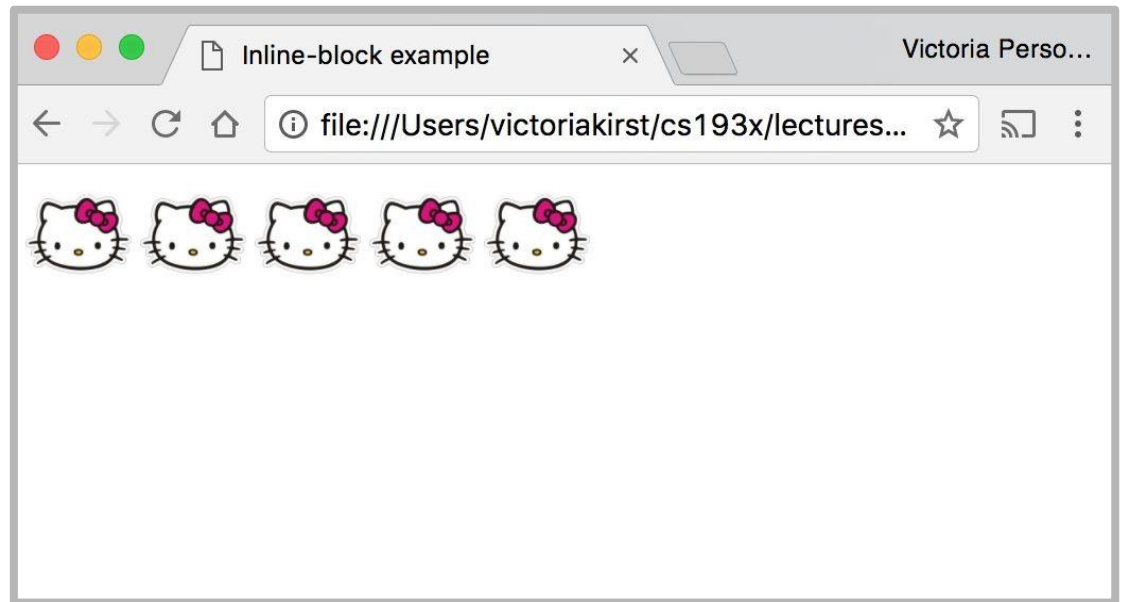


Inline-block

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

Extra slides

1. The course is adopted from vrkirst's lectures on Web Programming class. You can check the original page at <http://cs193x.stanford.edu> for the original course materials.
2. <https://www.youtube.com/watch?v=Dxcc6ycZ73M>
3. <https://www.techopedia.com/definition/2419/internet>
4. <https://www.theguardian.com/technology/2018/oct/22/what-is-the-internet-13-key-questions-answered>
5. <https://medium.freecodecamp.org/whats-the-document-object-model-and-why-you-should-know-how-to-use-it-1a2d0bc5429d>