CS50 - SECTION 5

10/16/18

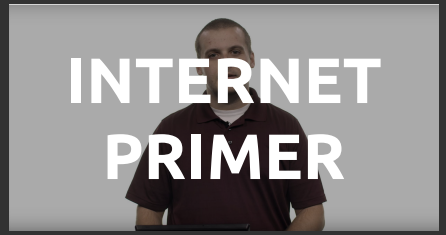
POST-MORTEM ON PSET 4

QUICK REMINDERS

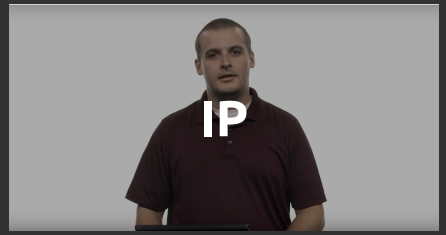
1. Remember to check for a `NULL` return when using `malloc()`, `fopen()`, etc.
2. Be careful how you implement your functions and try to encapsulate as much output as possible
   a. `hash()` should perform the `tolower()` and `%` operations for you
3. Extract common code from if-else branches
   a. Only use the if-else statement for non-common, distinct code (remember DRY)

CONCEPTS DEEP-DIVE

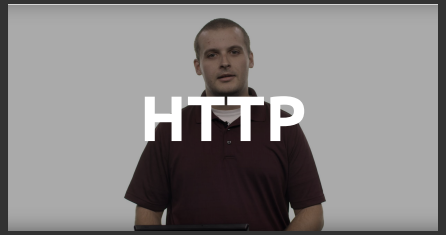
“SHORTS” FOR THE WEEK



https://www.youtube.com/watch?v=04GztBlVo_s

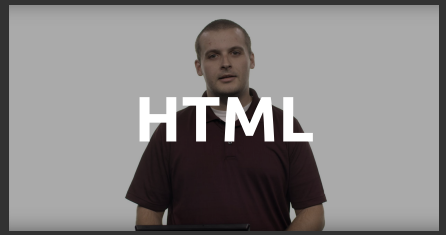


https://www.youtube.com/watch?v=A1g9SokDJSU



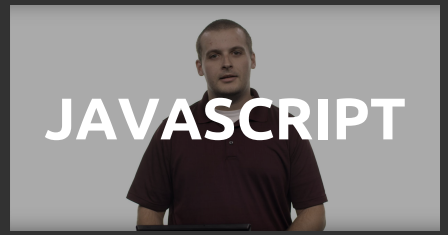
https://www.youtube.com/watch?v=4axL8Gfw2nI

https://www.youtube.com/watch?v=GP7uvI_6uas



https://www.youtube.com/watch?v=YK78KhMf7bs

https://www.youtube.com/watch?v=Ub3FKU21ubk



https://www.youtube.com/watch?v=Z93IaNfavZw

TCP/IP MODEL

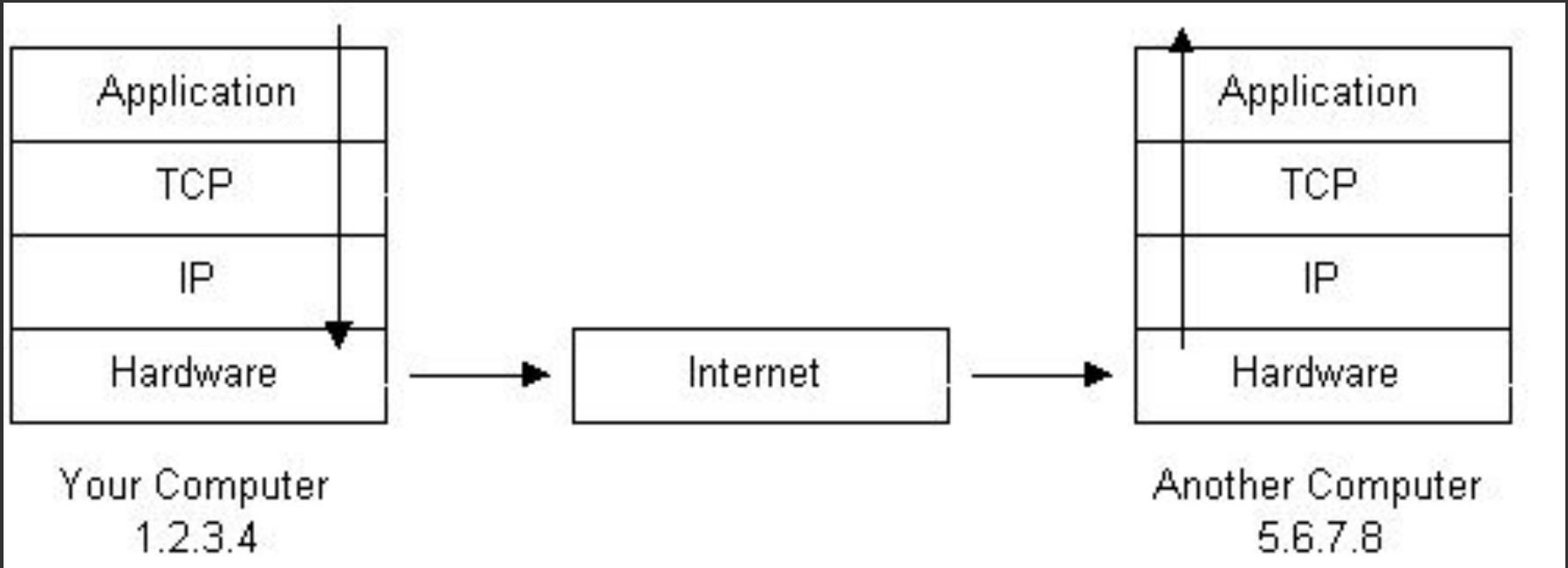
- The **TCP/IP Model** is simply a set of protocols that govern how communication happens over the internet
- Communication happens over a set of *layers* and information is broken up into *packets*

STANDARD LAYERS OF THE PROTOCOL STACK

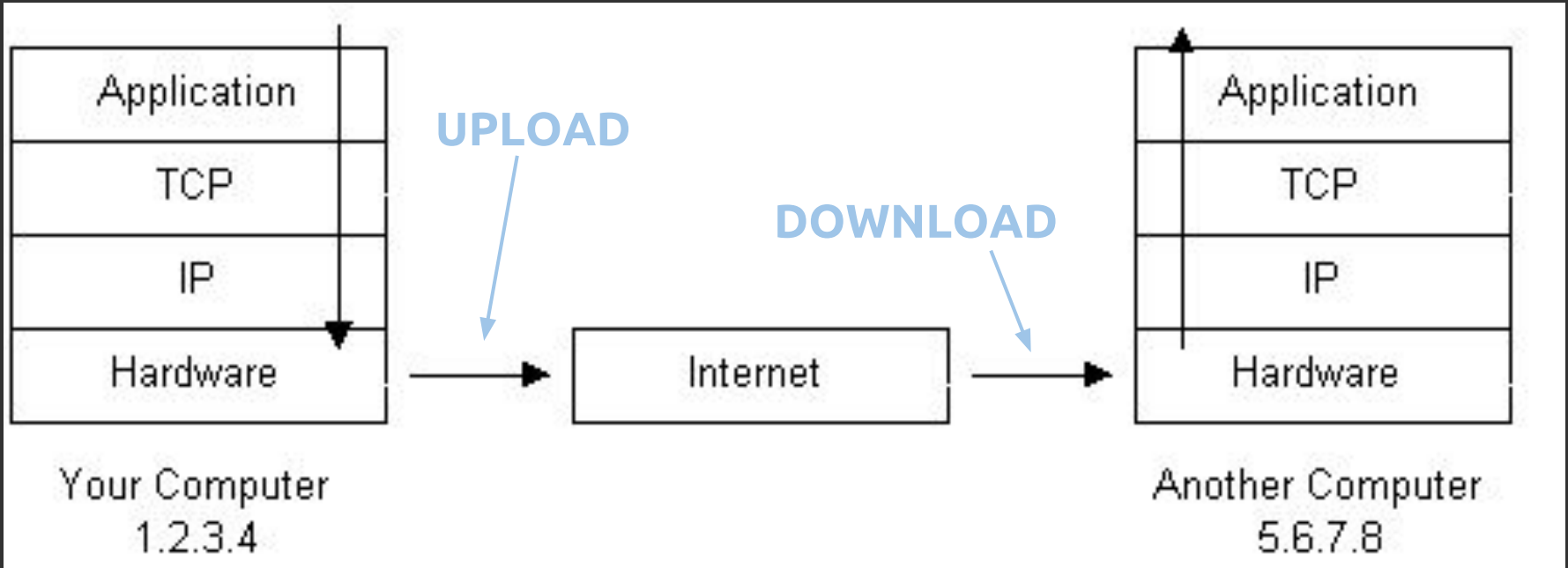
PROTOCOL LAYER	DESCRIPTION
APPLICATION	Protocols specific to applications such as WWW, e-mail, FTP, etc.
TCP	TCP directs packets to a specific application on a computer using a port number.
IP	IP directs packets to a specific computer using an IP address.
HARDWARE	Converts binary packet data to network signals and back. (E.g. ethernet network card, modem for phone lines, etc.)

Source: https://web.stanford.edu/class/msande91si/www-spr04/readings/week1/InternetWhitepaper.htm

AN EXAMPLE OF UPLOAD/DOWNLOAD



AN EXAMPLE OF UPLOAD/DOWNLOAD



HYPERTEXT TRANSFER PROTOCOL (HTTP)

- **HTTP** defines the the way different messages are transferred over the images (“what’s inside the envelope”)

GET / HTTP/1.1
Host: www.google.com

HYPERTEXT TRANSFER PROTOCOL (HTTP)

- **HTTP** defines the the way different messages are transferred over the images (“what’s inside the envelope”)

REQUEST METHOD

```
GET / HTTP/1.1
Host: www.google.com
```

HYPERTEXT TRANSFER PROTOCOL (HTTP)

- **HTTP** defines the the way different messages are transferred over the images (“what’s inside the envelope”)

REQUEST METHOD

GET / HTTP/1.1
Host: www.google.com

HTTP VERSION

HYPERTEXT TRANSFER PROTOCOL (HTTP)

- **HTTP** defines the the way different messages are transferred over the images (“what’s inside the envelope”)

REQUEST METHOD

GET / HTTP/1.1

Host: www.google.com

HTTP VERSION

REQUEST PAGE

HYPERTEXT TRANSFER PROTOCOL (HTTP)

- **HTTP** defines the the way different messages are transferred over the images (“what’s inside the envelope”)

```
GET / HTTP/1.1
Host: www.google.com
```

REQUEST METHOD

HTTP VERSION

REQUEST PAGE

Will be resolved to an IP address using DNS

HYPERTEXT TRANSFER PROTOCOL (HTTP)

- HTTP requests return a **status code** to indicate how the request was processed

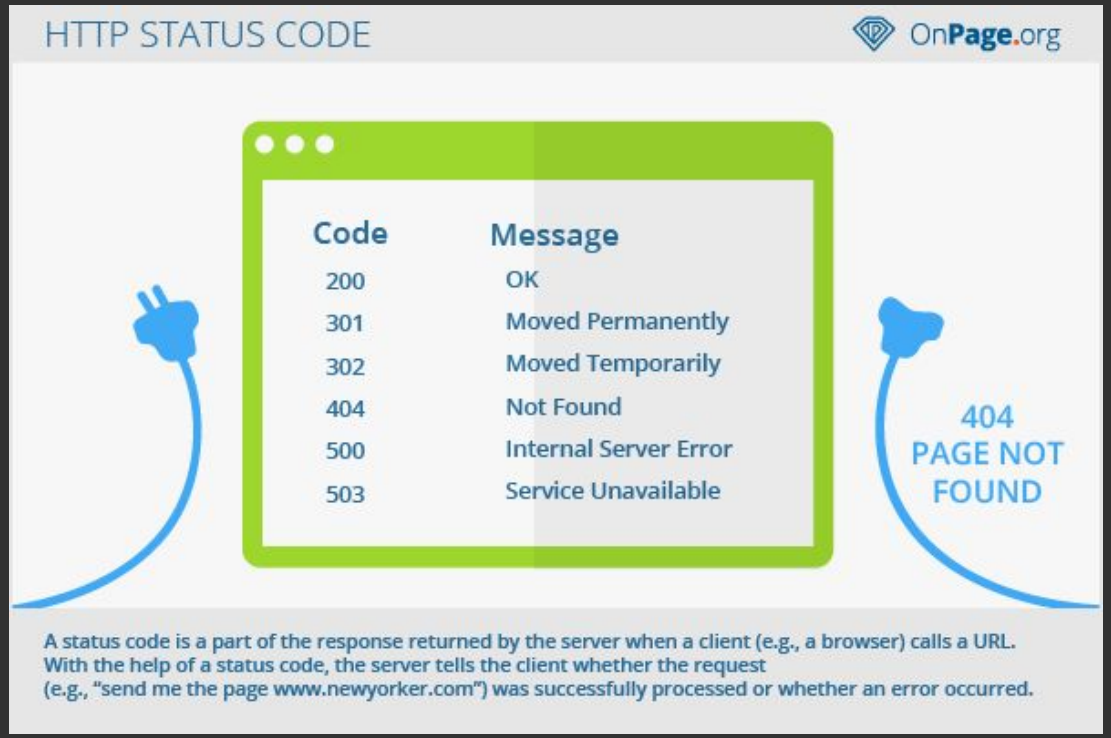
GET / HTTP/1.1
Host: www.google.com

200: OK

HTTP REQUEST METHODS

SAFE METHODS NO ACTION ON SERVER	GET	HTTP/1.1 MUST IMPLEMENT THIS METHOD
	HEAD	INSPECT RESOURCE HEADERS
MESSAGE WITH BODY SEND DATA TO SERVER	PUT	DEPOSIT DATA ON SERVER – INVERSE OF GET
	POST	SEND INPUT DATA FOR PROCESSING
	PATCH	PARTIALLY MODIFY A RESOURCE
	TRACE	ECHO BACK RECEIVED MESSAGE
	OPTIONS	SERVER CAPABILITIES
	DELETE	DELETE A RESOURCE – NOT GUARANTEED

HTTP STATUS CODES



And many more...

APPLICATION PROGRAMMING INTERFACES (APIs)

- An API is a standardized way for two web applications to talk to one another
- APIs are all about *transferring data* from one place to another in a standard way
  - <u>Example:</u> Twitter has an API to allow you to request tweets from it, post tweets, etc.

APPLICATION PROGRAMMING INTERFACES (APIs)

Why do APIs matter?

APPLICATION PROGRAMMING INTERFACES (APIs)

Why do APIs matter?

1. **We need them to communicate with a web application in a standardized way.**
    a. **Twitter wouldn't know how to handle our requests if they weren't structured in a consistent way**
2. **They greatly expand the opportunities for interoperability on the web**
    a. **You can now write web applications pull in data from other sources and post your own data elsewhere!**

RESTful APIs

- **Representational state transfer (REST)** is web-lingo for a standardized way of creating web resources
- REST APIs are APIs that comply with all the HTTP concepts you just learned—so you can GET, POST, PUT, DELETE, etc. to them

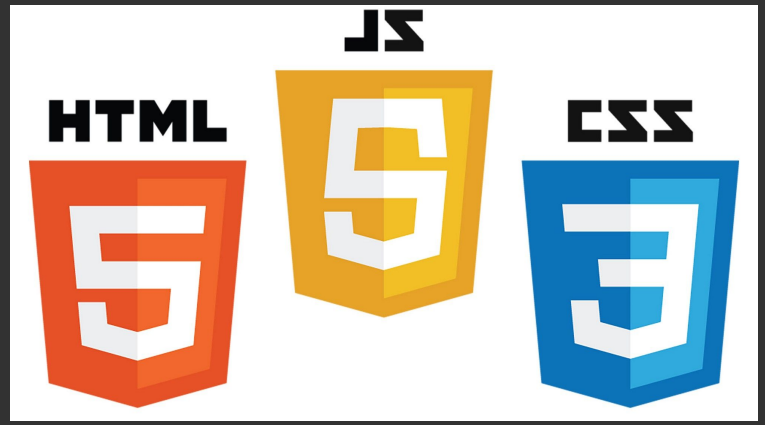
GET	/movies	Get list of movies
GET	/movies/:id	Find a movie by its ID
POST	/movies	Create a new movie
PUT	/movies	Update an existing movie
DELETE	/movies	Delete an existing movie

HTML, CSS, & JAVASCRIPT

- **HyperText Markup Language (HTML)** is a standard way for documents to be written on the web
  - It's standardized so that every browser knows how to interpret web pages when loading them
- **Cascading Style Sheets (CSS)** is another standard that allows you control the formatting of your web pages
- **JavaScript (JS)** is a web-oriented programming language allowing you to add interactive functionality to your web application

HTML, CSS, & JAVASCRIPT

- Each of these plays an important role on the web:
  - HTML = Document
  - CSS = Styling
  - JS = Bridge between document, styling, and interactive functionality



QUICK REVIEW OF HTML

- HTML gives us the tools to create the “content” of our document
- HTML is a standardized markup language that every browser can read
  - N.b. there are different specifications of HTML
  - The most recent one is HTML5, but it’s not even fully rolled out (e.g. Internet Explorer often can’t view HTML5-specific elements)

QUICK REVIEW OF HTML

```
<p style="color: red;">hello</p>
```

TAG

ATTRIBUTE

ELEMENT VALUE

QUICK REVIEW OF HTML

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>hello, world</title>
    </head>
    <body>
        <p style="color: red;">hello</p>
    </body>
</html>
```

DOCTYPE

BEGINNING OF HTML DOC

HEAD

BODY

ELEMENT

QUICK REVIEW OF CSS

- Creating HTML documents raises the natural question: “How do I format these?”
- CSS offers us a solution to this problem by allowing us to create **style sheets** with which we can insert different styles to format our HTML
- The first ‘C’ in CSS stands for *cascading* - This is because you can include multiple style sheets in your document and CSS will figure out the “precedence” of your styles for you when they conflict

QUICK REVIEW OF CSS

```
<p style="color: red;">hello</p>
```

How might we refactor the above code?

QUICK REVIEW OF CSS

HTML CODE	CSS CODE	

`<p>hello</p>`	`p`
	`{`
	`  color: red;`
	`}`
		Targeting the type of tag

`<p id="elem">hello</p>`	`#elem`
	`{`
	`  color: red;`
	`}`
		Targeting a specific ID

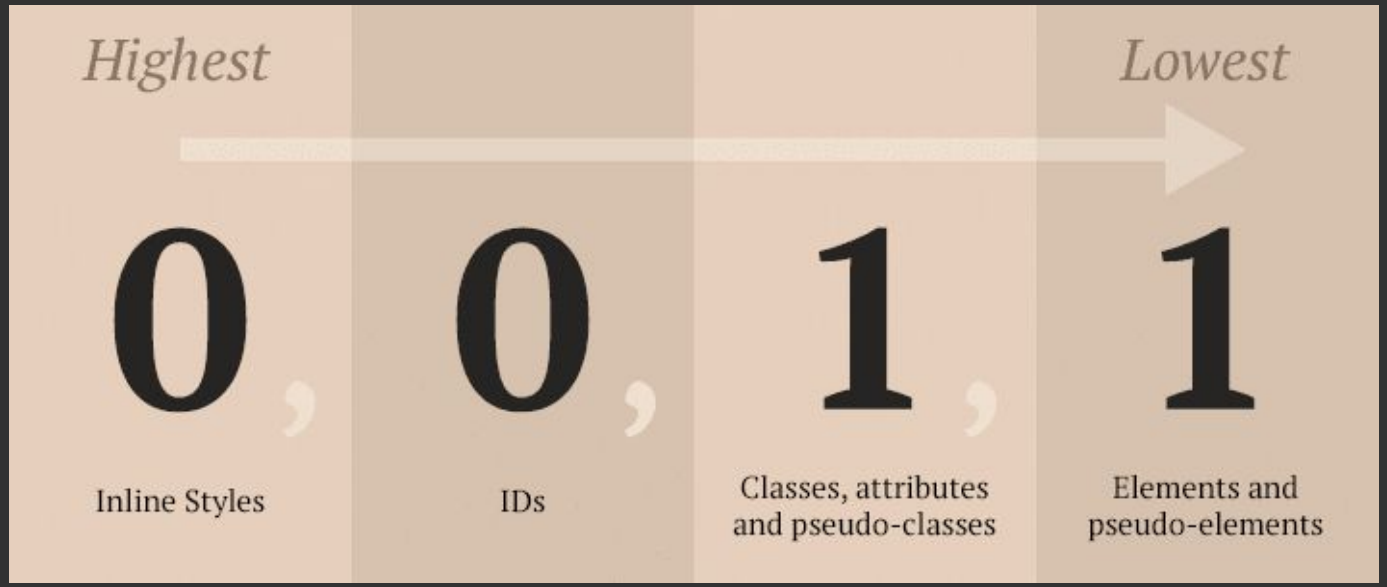
`<p class="red">hello</p>`	`.red`
	`{`
	`  color: red;`
	`}`
		Targeting an entire class

QUICK REVIEW OF CSS

- There are different “best practices” for when we use a specific approach over another
  - Tag Selector - You want a particular element to always look a certain way
  - ID Selector - You want a single element to look a specific way (you generally avoid giving different elements the same ID)
  - Class Selector - You want different elements (e.g. `<a>`, `<p>`, etc) to all look a certain way

QUICK REVIEW OF CSS

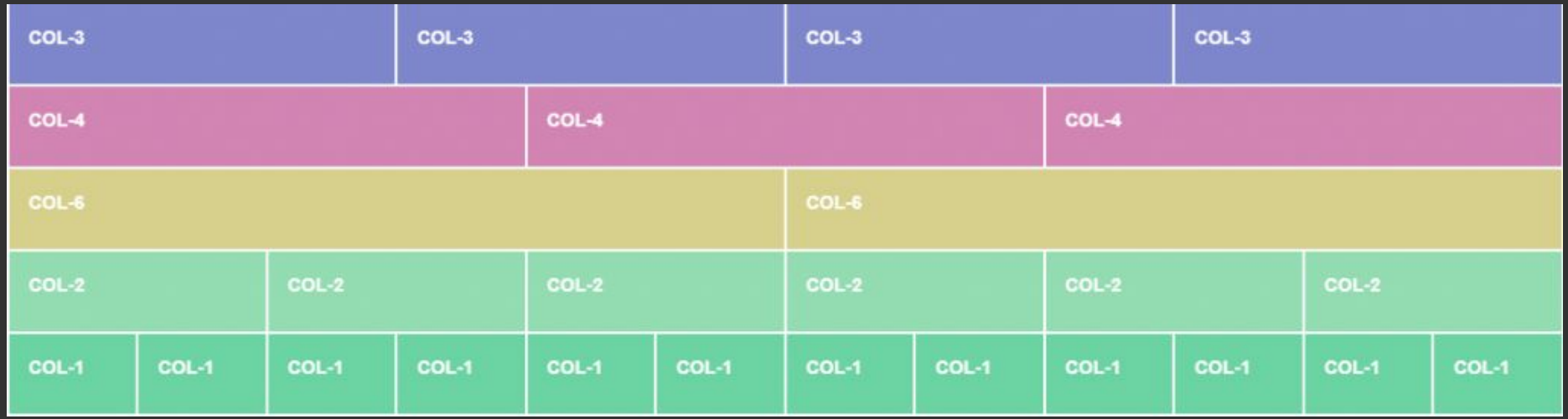
- Note that CSS will handle conflicts for you in terms of "precedence"



BOOTSTRAP

- **Bootstrap** is a front-end framework developed by Twitter
- It's *massively* popular and you've probably seen instances of it all over the web
- It was originally designed to ensure internal consistency among applications at Twitter, but most web developers rely upon it to design **responsive web applications** using its grid

BOOTSTRAP


COL-3
COL-3
COL-3
COL-3
COL-4
COL-4
COL-4
COL-6
COL-6
COL-2
COL-2
COL-2
COL-2
COL-2
COL-2
COL-1
COL-1
COL-1
COL-1
COL-1
COL-1
COL-1
COL-1
COL-1
COL-1
COL-1
COL-1

HOW DO I EVEN GET STARTED USING ANY OF THIS?

Include Bootstrap into your code:

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
crossorigin="anonymous">
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"
integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"
integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
crossorigin="anonymous"></script>
```

Include any custom CSS you want as well:

```
<link rel="stylesheet" href="custom.css">
```

BROWSER DEVELOPER TOOLS

- Most browsers include some sort of developer tools to aid in web development
  - By far, Google Chrome and Firefox offer the most robust and full-featured ones

BROWSER DEVELOPER TOOLS

- Most browsers include some sort of developer tools to aid in web development
  - By far, Google Chrome and Firefox offer the most robust and full-featured ones

DEMO TIME

INTERACTIVE ACTIVITY

Use developer tools to visit a website and modify its HTML and CSS in some way. Be ready to share what your remix!

JAVASCRIPT

- JavaScript allows us to add interactivity to our web applications, create dynamic interfaces, and manipulate HTML/CSS in complicated ways
- JavaScript is a **weakly typed** language—Unlike C, it doesn't ask for your to specify a variable type
- It typically runs **client-side**, but server-side JS (`node.js`) has exploded in popularity in recent years

JAVASCRIPT - VARIABLE ASSIGNMENT

```
// a simple variable
let age = 19;

// an array
let array = [1, 2, 3, 4, 5];

// string
let str = "Happy birthday, Maria!";

// an object
let teacher = {name: "David", course: 50};
```

JAVASCRIPT - CONDITIONS AND LOOPS

```
// while loop
while (true)
{
   // do something
}

// for loop
for (initialization; condition; update)
{
   // do something
}

// if condition
if (true)
{
   // do something
}
```

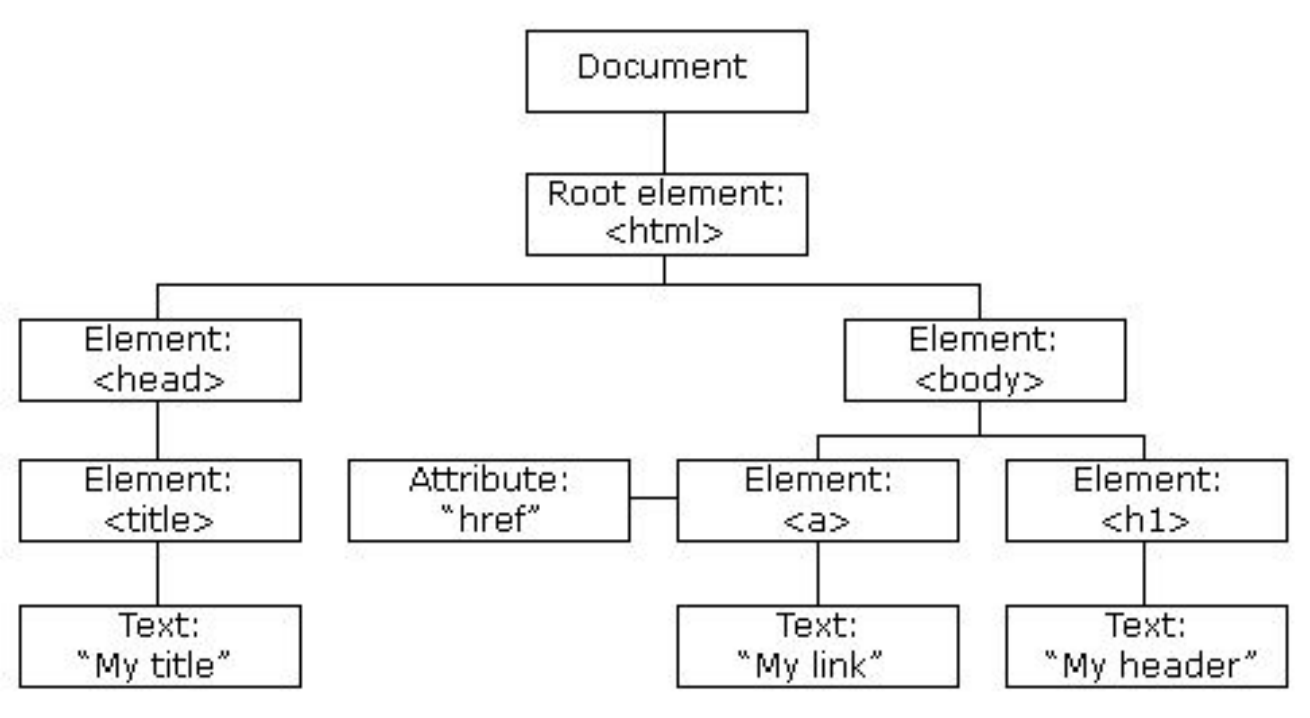
JAVASCRIPT

- On its own, JS is just like C: We can write programs in it, perform calculations, etc.
- Where JS becomes powerful is in how it interacts with the web ecosystem
  - We can use JS to *actively* modify HTML/CSS through the **Document Object Model (DOM)**
  - JS also includes a powerful **event listener** toolset to *respond* to changes in the DOM

DOCUMENT OBJECT MODEL

- Every HTML document is **parsed** by a browser into a hierarchical DOM and then rendered to the user
- We can access the DOM using JavaScript and begin to modify it dynamically as we wish

DOCUMENT OBJECT MODEL



MODIFYING THE DOM WITH JAVASCRIPT

`let elem = document.querySelector("p");`	**Retrieves a single node from JavaScript using a query selector**
`let elems = document.querySelectorAll("p");`	**Retrieves *all* applicable nodes from JavaScript using a query selector**

MODIFYING THE DOM WITH JAVASCRIPT

- Once we have selected a node using JS, we can begin to access its attributes using dot notation and modify them:

```
let elem = document.querySelector("p");
elem.innerHTML = "New paragraph contents.";
```

MODIFYING THE DOM WITH JAVASCRIPT

- We might also want to traverse over multiple elements we've selected:

```
let elems = document.querySelectorAll("p");

elems.forEach(function(item) {
  item.innerHtml = "New HTML";
});
```

MODIFYING THE DOM WITH JAVASCRIPT

- We might also want to traverse over multiple elements we've selected:

Returns a collection of nodes, similar to an array

```
let elems = document.querySelectorAll("p");


elems.forEach(function(item) {
  item.innerHtml = "New HTML";
});
```

MODIFYING THE DOM WITH JAVASCRIPT

- We might also want to traverse over multiple elements we've selected:

Returns a collection of nodes, similar to an array

```
let elems = document.querySelectorAll("p");

elems.forEach(function(item) {
  item.innerHtml = "New HTML";
});
```

Iterates over each element in the collection, accessed via `item` for each iteration of the loop

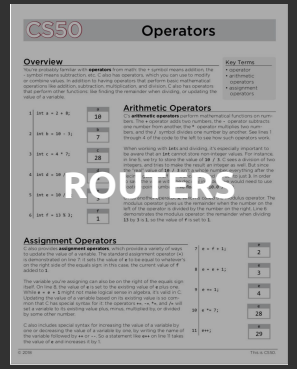
COMMON DOM PROPERTIES

PROPERTY	DESCRIPTION
`innerHTML`	Holds the HTML inside a set of HTML tags.
`nodeName`	The name of an HTML element or element’s attribute.
`id`	The "id" attribute of an HTML element
`parentNode`	A reference to the node one level up in the DOM.
`childNodes`	An array of references to the nodes one level down in the DOM.
`attributes`	An array of attributes of an HTML element.
`style`	An object encapsulating the CSS/HTML styling of an element.

INTERACTIVE ACTIVITY

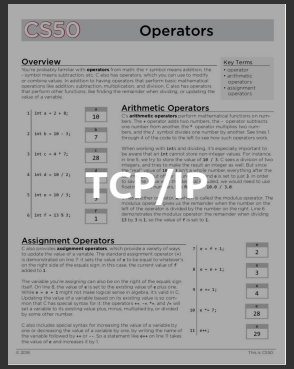
Use developer tools to inject a JavaScript script of your own creation into a web page. It should utilize a `forEach()` loop and perform some sort of DOM manipulation systematically of your choice.

REFERENCE SHEETS



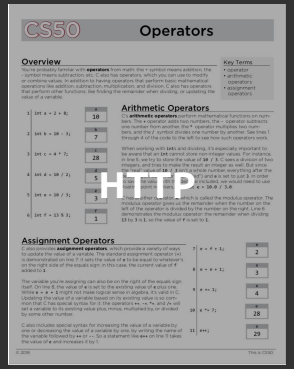
ROUTERS

https://www.dropbox.com/sh/5y662ey1hc4sde4/AABgqPxFuk1JF7qVowbyLCNza/Routers.pdf?dl=0



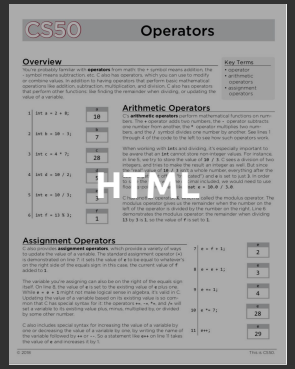
TCP/IP

https://www.dropbox.com/sh/5y662ey1hc4sde4/AACjgHN3NtSKk4ShsRDFd_Sja?dl=0&m=&preview=TCP+and+IP.pdf



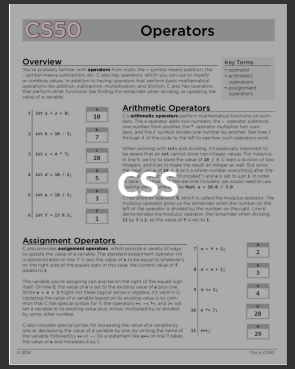
HTTP

https://www.dropbox.com/sh/5y662ey1hc4sde4/AACjgHN3NtSKk4ShsRDFd_Sja?dl=0&m=&preview=HTTP.pdf



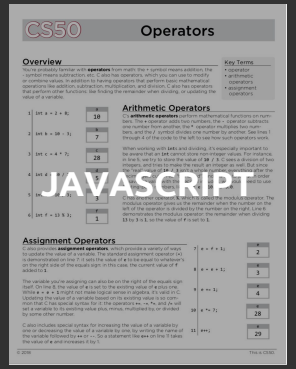
HTML

https://www.dropbox.com/sh/5y662ey1hc4sde4/AACjgHN3NtSKk4ShsRDFd_Sja?dl=0&m=&preview=HTML.pdf



CSS

https://www.dropbox.com/sh/5y662ey1hc4sde4/AACjgHN3NtSKk4ShsRDFd_Sja?dl=0&m=&preview=CSS.pdf



JAVASCRIPT

https://www.dropbox.com/sh/5y662ey1hc4sde4/AACjgHN3NtSKk4ShsRDFd_Sja?dl=0&m=&preview=JavaScript.pdf

LOOKING AHEAD

LOOKING AHEAD

Problem Set	Language	Released	Due
Problem Set 0	Scratch	Fri 9/7	Tue 9/11, 11:59pm
Problem Set 1	C	Fri 9/14	Thu 9/20, 11:59pm
Problem Set 2	C	Fri 9/21	Thu 9/27, 11:59pm
Problem Set 3	C	Fri 9/28	Thu 10/4, 11:59pm
Problem Set 4	C	Fri 10/5	Thu 10/11, 11:59pm
Problem Set 5	HTML, CSS	Fri 10/12	Tue 10/16, 11:59pm
Problem Set 6	Python	Fri 10/19	Thu 10/25, 11:59pm
Problem Set 7	Python, HTML, CSS	Fri 10/26	Thu 11/1, 11:59pm
Problem Set 8	SQL, Python, HTML, CSS	Fri 11/2	Thu 11/8, 11:59pm

LOOKING AHEAD

Problem Set	Language	Released	Due
Problem Set 0	Scratch	Fri 9/7	Tue 9/11, 11:59pm
Problem Set 1	C	Fri 9/14	Thu 9/20, 11:59pm
Problem Set 2	C	Fri 9/21	Thu 9/27, 11:59pm
Problem Set 3	C	Fri 9/28	Thu 10/4, 11:59pm
Problem Set 4	C	Fri 10/5	Thu 10/11, 11:59pm
Problem Set 5	HTML, CSS	Fri 10/12	Tue 10/16, 11:59pm
Problem Set 6	Python	Fri 10/19	Thu 10/25, 11:59pm
Problem Set 7	Python, HTML, CSS	Fri 10/26	Thu 11/1, 11:59pm
Problem Set 8	SQL, Python, HTML, CSS	Fri 11/2	Thu 11/8, 11:59pm

YOU ARE HERE

LOOKING AHEAD

Problem Set	Language	Released	Due
Problem Set 0	Scratch	Fri 9/7	Tue 9/11, 11:59pm
Problem Set 1	C	Fri 9/14	Thu 9/20, 11:59pm
Problem Set 2	C	Fri 9/21	Thu 9/27, 11:59pm
Problem Set 3	C	Fri 9/28	Thu 10/4, 11:59pm
Problem Set 4	C	Fri 10/5	Thu 10/11, 11:59pm
Problem Set 5	HTML, CSS	Fri 10/12	Tue 10/16, 11:59pm
Problem Set 6	Python	Fri 10/19	Thu 10/25, 11:59pm
Problem Set 7	Python, HTML, CSS	Fri 10/26	Thu 11/1, 11:59pm
Problem Set 8	SQL, Python, HTML, CSS	Fri 11/2	Thu 11/8, 11:59pm

MORE WEB-ORIENTED THINGS TO COME...

AND PYTHON TOO

REMINDER: GOALS OF THE COURSE

1. Learn to think computationally and algorithmically.
2. Improve your problem-solving skills—this will help you in every other course and in real-life too!
3. Gain practical programming skills in a plethora of languages (C, Python, JavaScript, etc.)
4. Familiarize yourself with best practices for software design.
5. Build a community with your peers that will stay with you after this course.

LOOKING AHEAD

- ★ Now is the time to think back to the goals you set for yourself at the beginning of the course
- ★ Ask yourself:
  - What did you want out of this class when you began? Have those things changed?
  - How are you performing in meeting the goals you set for yourself? How might you adjust as needed?
  - What do you like about the class? What do you find frustrating? What did you want to learn that you haven't yet?

LOOKING AHEAD

- ★ Since there is not problem set due later this week, spend the extra time you have to sit down and think intentionally about these reflection questions
- ★ *I'm always a resource for you—Reach out if you want to co-strategize, get advice, or just talk*