# Section 1: Front-End Web Development

## Lecture

### Notes

### Questions

### Summary

Standard Template Below

# Section 1: Front-End Web Development

## Lecture 6: How does the Internet Work?

* Think of the internet as a long wire connected to computers along the way
* Some computers have to stay online 24/7 ready to serve data – these are **servers**
* Any computer used to ‘access’ the internet is referred to as a **client**

Searching:

* Browser sends message to your ISP with the URL you want to visit
* ISP sends that message on to a DNS (Domain Name System) server
  + Essentially a super phonebook
* DNS Server will look up that website in their database , and return the exact IP address
* Once found, it sends the IP address back to your browser
* You then send a direct request to that address through your ISP to be delivered through the internet backbone, the server at the given IP address will process that request, then send the data back to you through the internet backbone

## Lecture 7: How do Websites Work?

Browser

* Software that lets you look up the IP address of the website, and render it
* Data sent back generally come back in HTML, CSS, and JS files

HTML

* Responsible for structure of the site (skeleton)

CSS

* Responsible for styling the website (fashion / clothing of a human)

Javascript

* Code that actually lets your website do things or have behaviour (muscles of the body)

# Section 2: Intro to HTML

## Lecture 11: Introduction to HTML

HTML 🡪 HyperText Markup Language

XML 🡪 Extensible Markup language

GML 🡪 Generalized Markup Language

## Lecture 14: The HTML Boilerplate



* HTML tags tell the browser that everything in this tag is HTML code
  + Consists of Head and Body
* Head is the component that holds info about the webpage, and tells the browser how it should handle the page
  + Could have **title** tag which will give the site’s title (look at tab name)
  + **Meta** elements give additional meta data to your html site
    - In this case, it’s that everything is coded using the utf-8 standard
    - 

Move on to Section 6 to skip most of the intro HTML and CSS

# Section 6: Introduction to Bootstrap 3

## Lecture 67: What is Bootstrap

* Bootstrap is a front-end library
* Huge advantage in responsiveness
  + Responding to the viewport
* Also has a bunch of pre-styled elements
* CSS button generator?
* Codeply
* Bootstrap documentation is very friendly

## Lecture 68: Installing Bootstrap

Quick explanation of CDN:

CDN – Content Delivery Network

Rather than having all the information delivered over the internet backbone, CDN provides a bunch of different points where data can be accessed – cuts down on latency

Bootstrap is hosted at tons of different nodes on the planet

* Likely to be stored in the user’s cache, but this will still cut down on latency
* Some of the components actually need bootstrap javascript – might be worthwhile using the started template for bootstrap sites

## Lecture 69: Web Design 101 – Wireframing

Wireframes are low fidelity designs of your website

* Kind of like first pass blueprinting

Mockups are higher fidelity – almost like screenshots from the future

Dribble is a phenomenal resource for looking at other people’s work

Sneakpeekit gives you templates for your own design

Balsamiq is industry standard for wireframing on the computer

## Lecture 74: Bootstrap Grid Layout System

[Grid layout bootstrap documentation](https://getbootstrap.com/docs/4.0/layout/grid/)

* Start with row div, then add your columns
* Your columns (**col**) will automatically distribute themselves evenly across the row
* Total for each row is 12
* Col-md-6 explianed:
  + We should have a 6 unit column on any size medium or greater
  + Anything smaller will take up the full width

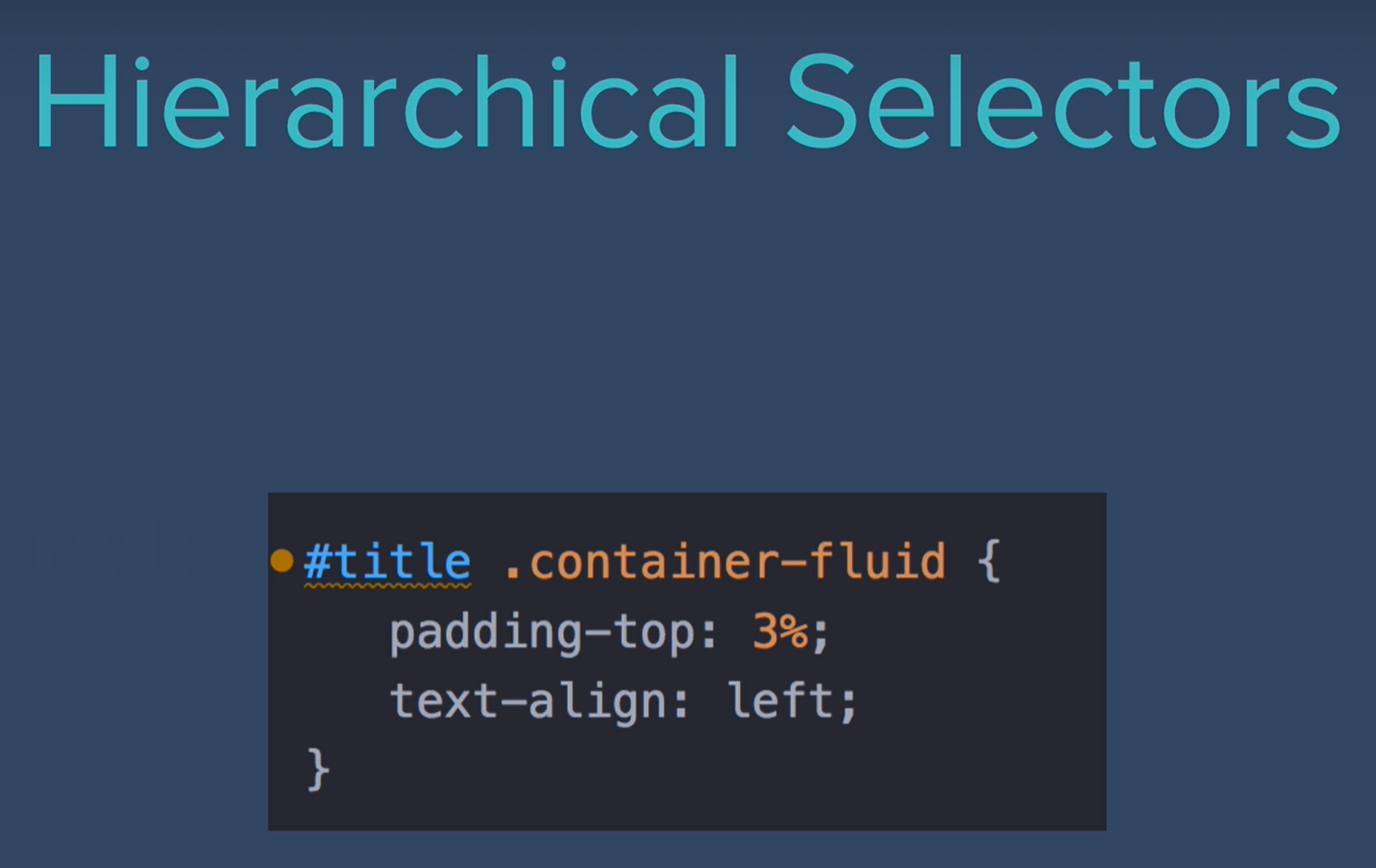
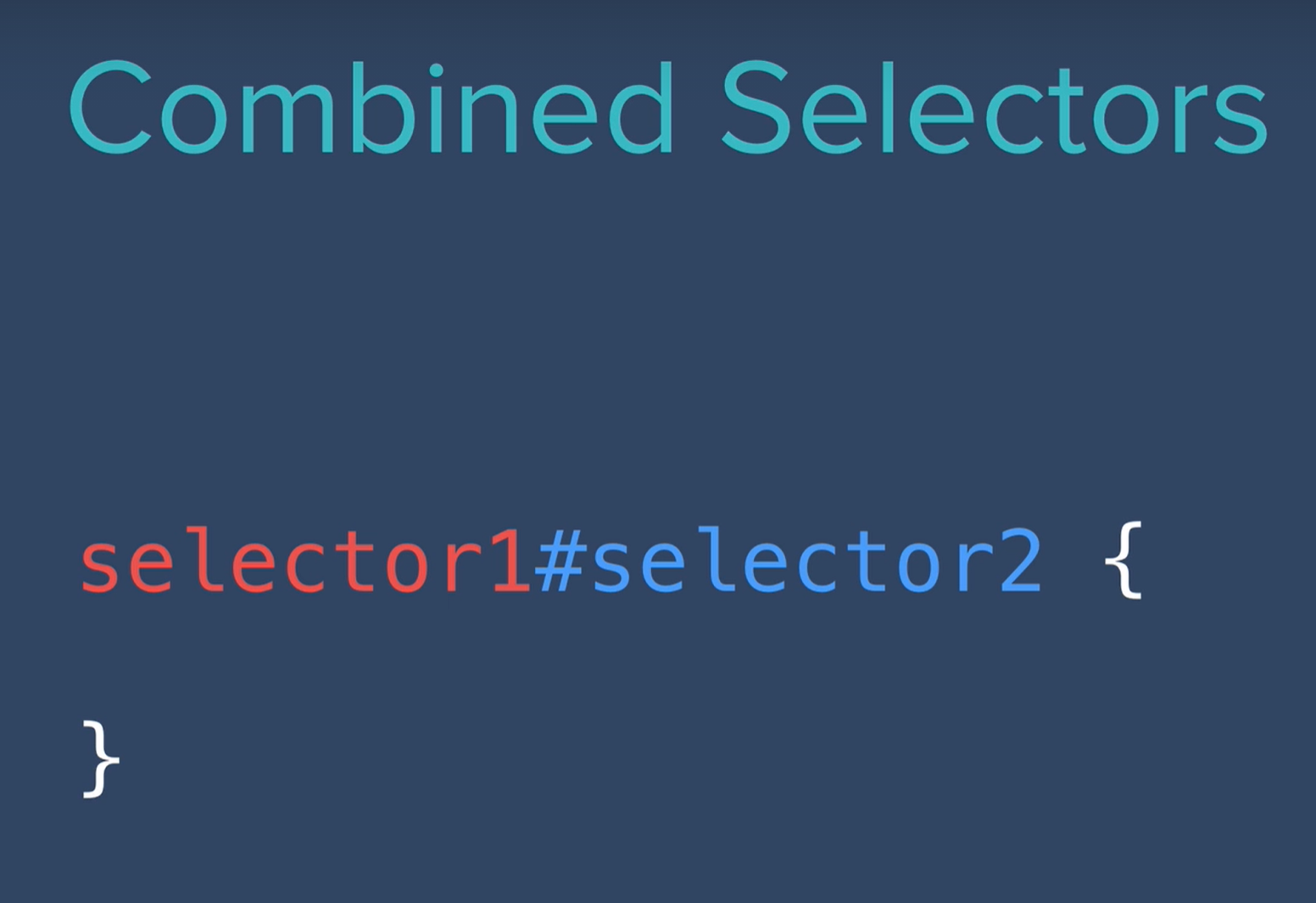
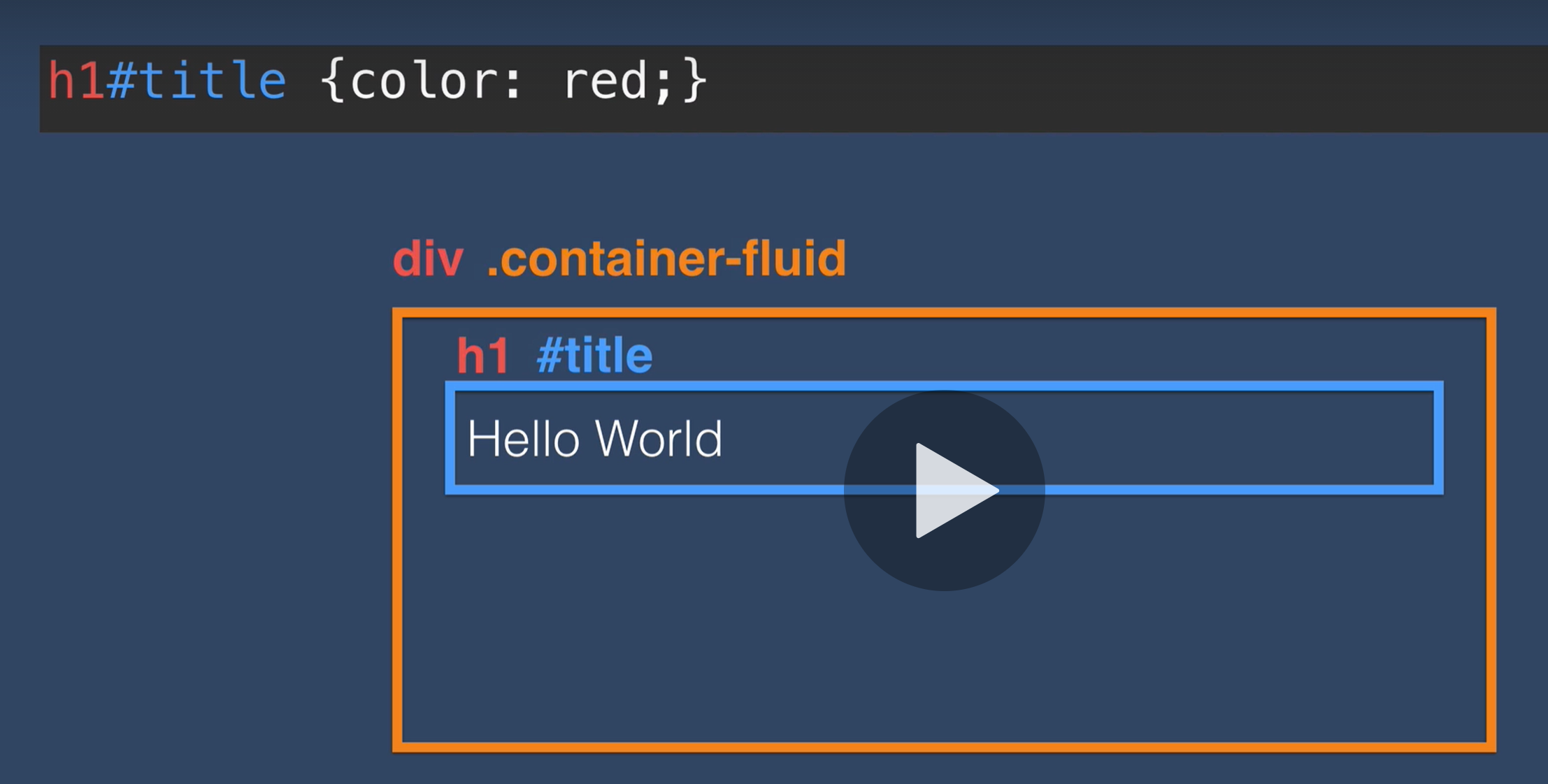
Container-fluid gives a more clean movement through the viewport breakpoints

* Works really well for text

# Section7: Intermediate Bootstrap

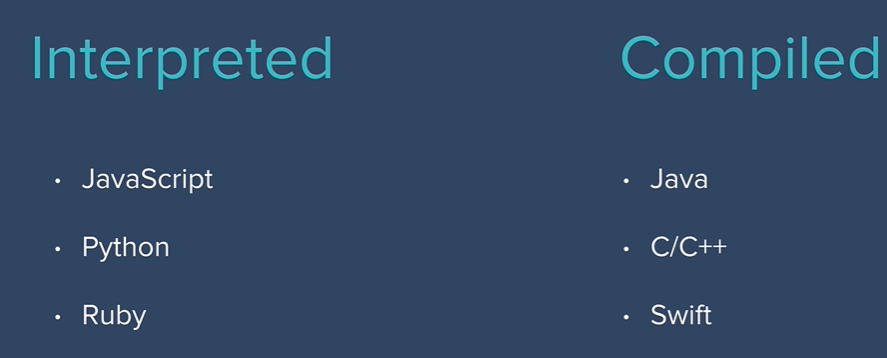
Standard padding of 3% 7% seems to work really well

## Lecture 94: Advanced CSS – Combining Selectors

 \*\*\*\*\* Must occur in the same element 

## Lecture 96: Advanced CSS – Selector Priority

# Section 9: Introduction to Javascript ES6



## Lecture 113: Naming Conventions for Javascript Variables

* camelCase

## Lecture 114- : Javascript Fundamentals

String Concatenation: “a” + “b”

String.length will give you the length

**Tweet Count exercise:**

var userInput = prompt("Enter your tweet ");

console.log(userInput);

var inputCharacterCount = userInput.length;

var remainingCharacters = 140 - inputCharacterCount;

alert(String(inputCharacterCount) + ' Used. ' + String(remainingCharacters) + " Remaining");

**slice(x,y)**

Let’s you slice your strings

String.slice(0,1) 🡪 just going to get the starting slice (start, end)

|

|

|

V

Skipping to section 10

# Section 10: Intermediate Javascript

## Lecture 132: Random Number Generation in Javascript



# Section 11 & 12: The Document Object Model

* Basically catalogs the page into individual objects that we can select and manipulate
* Browser turns all the html into a tree structure

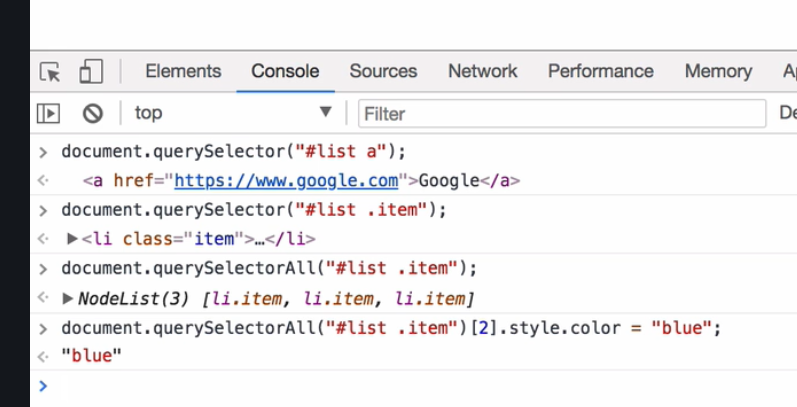




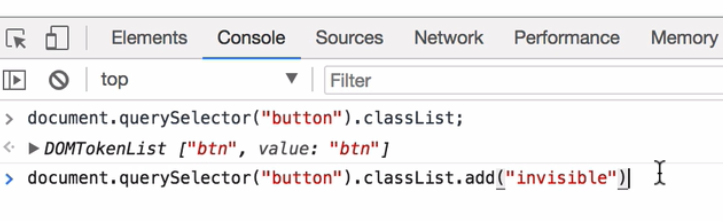
Getter and setters

document.querySelector(" ")

document.getElementsByTagName(“ “ )



* The idea is that all styles should be stored or predefined in your style.css sheet
* But what do you do if you want to change some values on the fly?
* Well, you can use javascript to directly change that object’s structure or attributes
* OR you can add predefined classes to it!

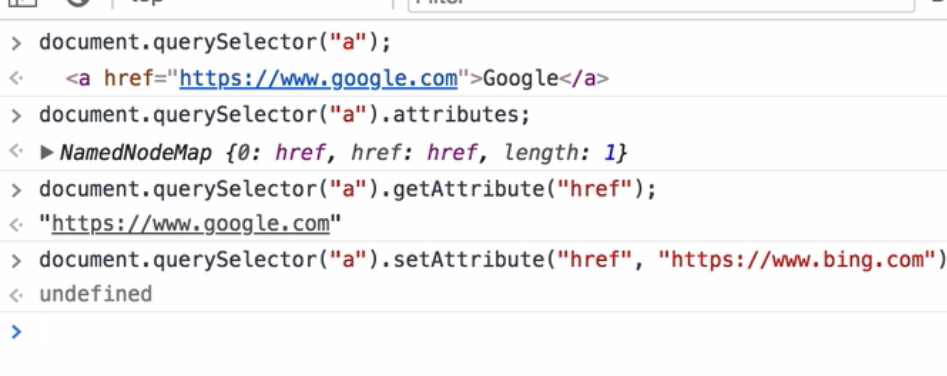


Can also toggle these



Can pass style through querySelector.innerHTML

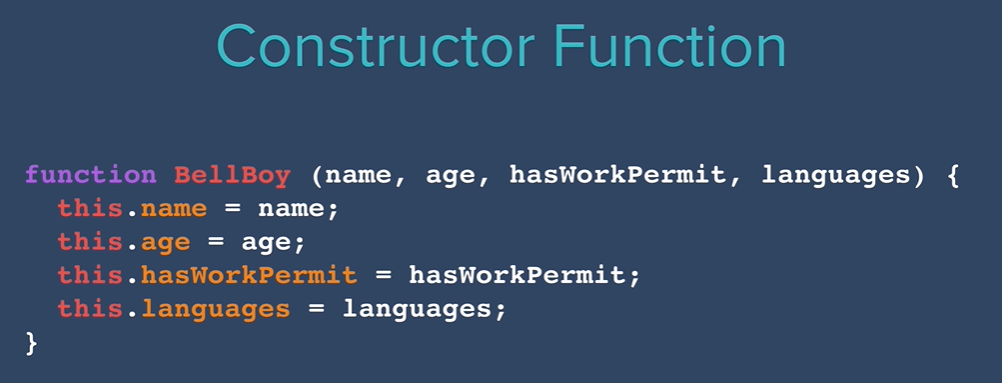


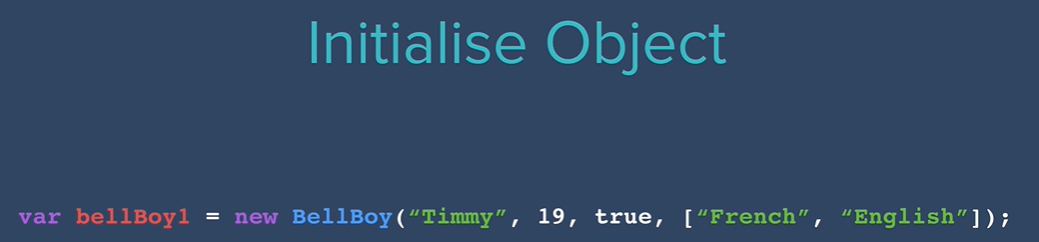


Attributes are anything that go within the tag

* Class
* Id
* Type
* Href

# Section 13: Advanced Javascript and DOM manipulation

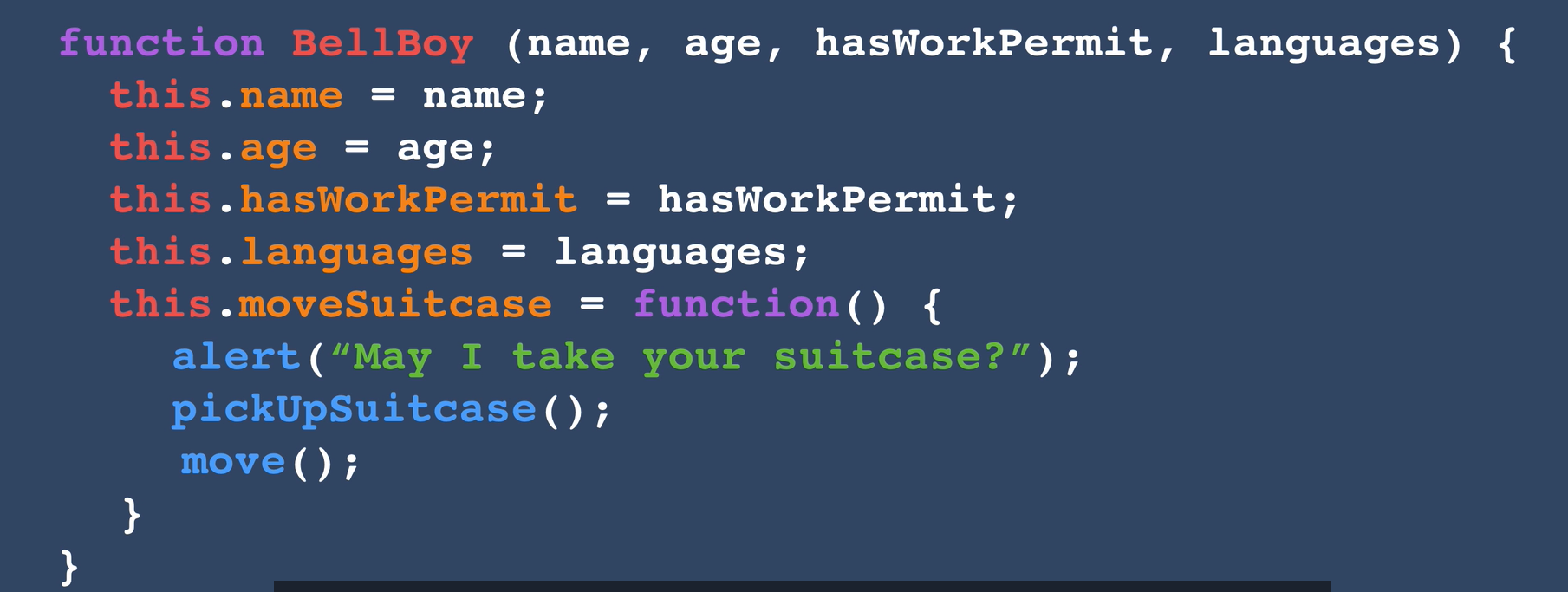


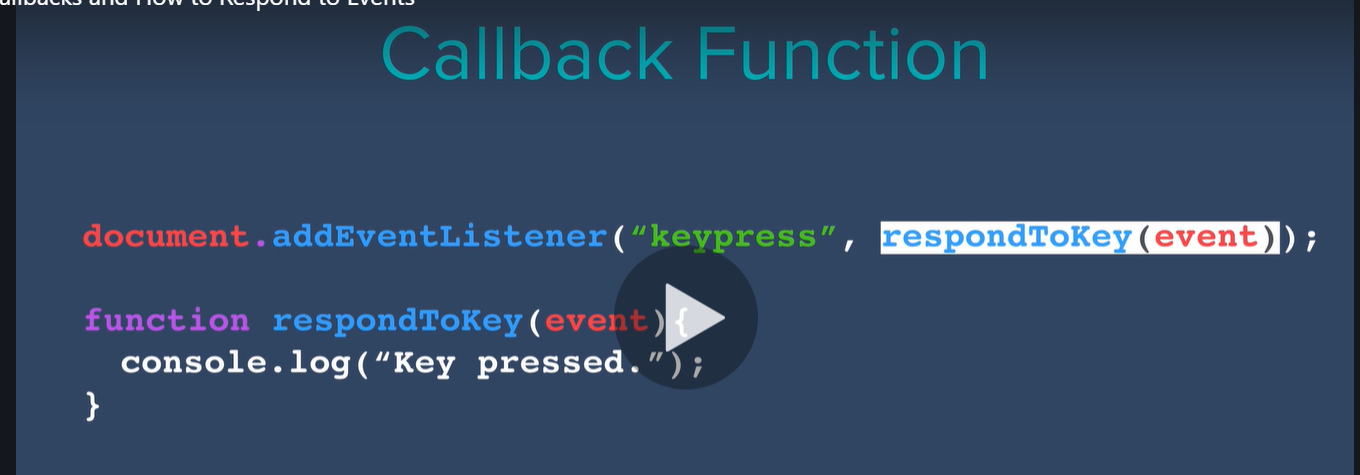


- constructor function / object name should follow FirstSecond uppercase structure

- use the “new” keyword to initialize the new object

Defining Methods





# Section 14 & 15: jQuery

* jQuery is the most popular javascript library



With the .css property, if you fill one argument, you’re getting the value of that property. If you fill in another, you’re now setting it

Set: 

Get: 

Changing the style directly through javascript feels wrong through

* Better to use jQuery to **ADD** to maintain the distinction between files.





Can add and remove classes



Use a space to add multiple classes



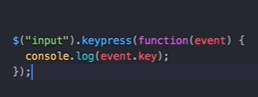
Can check for classes and return boo



$(“selector”).attr(“get”,”set”)



Event Listeners 🡪 No need to for loop with jQuery if you have multiple selections you want to apply this to

Can apply to keydown events as well



The **on** method allows you to select the type of listener via parameter



Can add html elements before and after the queried element

There’s also prepend and append to add it directly into the element you’ve queried



To remove all queried elements

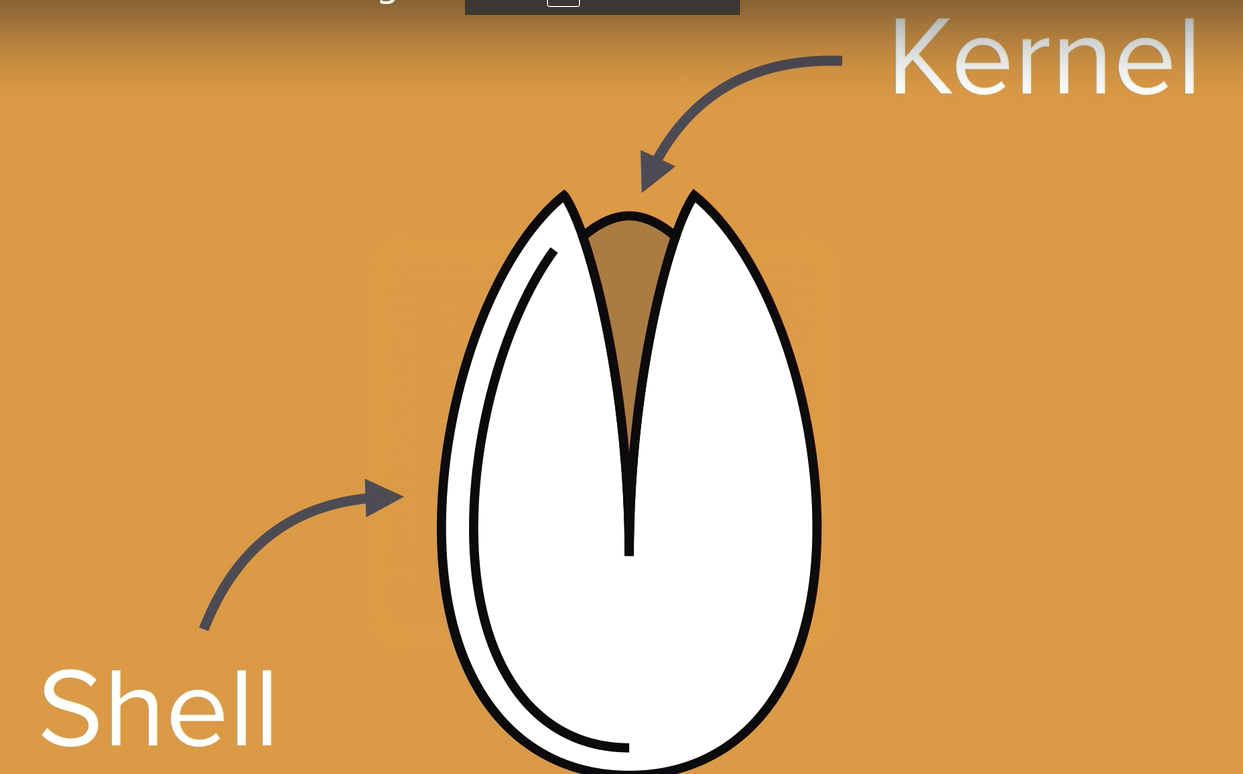
Animations:

$().hide(), .show(), .toggle(), fadeout, fadein, fadetoggle.

* Slideup, slidedown,slidetoggle (actually seems useful)

**DOCUMENTATION: jQuery Efffect Methods**

# Section 16: The Unix Command Line



Kernel is the program that interfaces with the hardware – the core of your OS

The shell refers to the user interface – how the human gets to interact with the hardware

* GUIs and Command line interfaces

Bash = Bourne Again Shell

CLI (command line interpreter) for the unix system

ls 🡪 list all files in current directory

cd 🡪 change directory

~ is your root directory

Create file -> touch filename.extension

Use ‘open’ to open file

* Specify application with open -a Atom filename.txt
* (windows) start <application> <file>

Use **rm** to remove file

Remove all files in current directory with rm \* 🡪 \*as a wildcard

To remove a folder or directory use rm -r directoryName

# Section 17: Backend Web Dev Explained

Server, database, applications, business logic,

Using the example of a restaurant:

* Client-Side: Where everyone’s seated, eating
* Server: Kitchen. Place where all the dishes get served from
* Larder: Where food is stored – the database

# Section 18: Nodejs

* The javascript we’ve written so far is trapped within the browser.
* Can’t access file systems on the computer, listen to requests independent of the browser.
* nodeJS lets us write JS to affect the hardware

REPL:

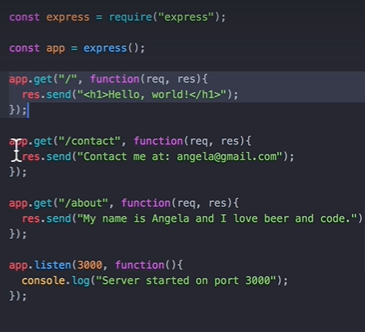
**Read**

**Evaluation**

**Print**

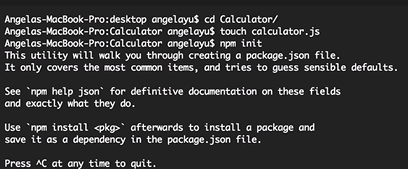
**Loop**

# Section 19: express.js



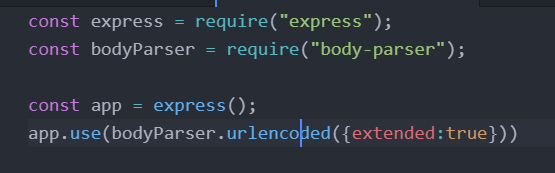
The underlying idea is that once we receive a request for a given route, we respond with response.send

**Setup:**

****

**Follow up with npm install express**

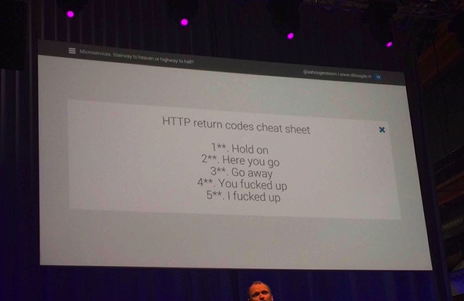
****

****

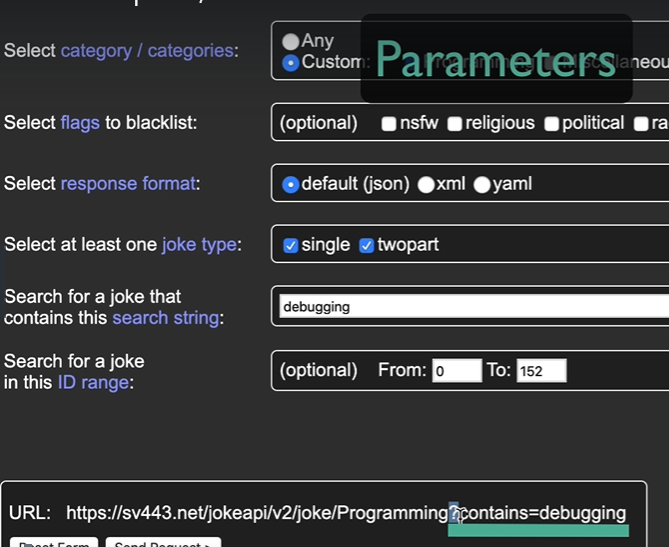
**For html forms**



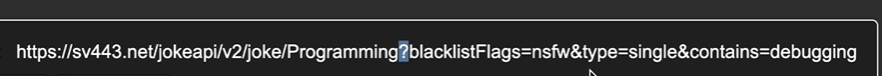
\_\_dirname will consistently return the directory name of the server file



# Section 20: APIs



<https://v2.jokeapi.dev/joke/any>



Every subsequent parameter follows a &

* When you’re writing for your server side, app.get is how you handle people trying to reach a page – not you trying to get something from someone else
* https.get however is using the htttps library instead of the node library to send a get request

Although you can only use one res.send per get request, you can use multiple res.write

Skipping ahead again:………

# Section 26: MongoDB

Commands:

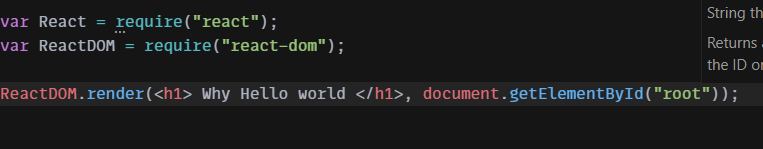
mongod 🡪 spins up mongo server

mongo -> takes you to mongo shell

show dbs 🡪 shows the database names

# Section 33: React.js

* Within the body tags, you will often find a div id = “root”
  + This acts as the standard root for our JS application



React works by creating these JSX files where you have HTML right in the body of a javascript

* The compiler takes the HTML and compiles it down to actual javascript

Inside the react module is **babel** – a javascript compiler that takes next generation javascript and converts it down to browser readable javascript

NOTE: The render method only accepts one element, but you can nest them inside a single div to get around this

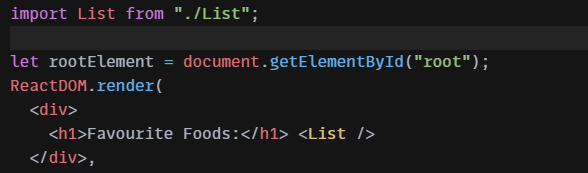
Expressions vs statements in javascript

In your script src in the html, include type as text/JSX

JSX Components should be separated into their own files

At the bottom of your component file, you should use **export default <componentname>**

Then, import <Component name> from “./



A lot of the react bits won’t have html inside their index.js

There will usually be an App.jsx file

React apps will generally have a lot of components

So you should have a component file within the src folder

Local setup time

PROPS:

* Passing properties is like passing parameters into one of these react functions
* Call on the props using props.name (for example)
* I don’t yet see the advantage of using this structure over more strict parameters
  + More continuity with html-like structure?

You Can simplify using the ternary operator:

Condition **?** Do if True **:** DO if False

isCloudy === true ? bringUmbrella() : bringSunscreen()

If your alternative to your condition being true is null, you can just use the && operator

You can essentially structure it as **condition** && expression

* This is because not meeting the condition will evaluate to null

Meeting the condition will render the other segment

States

Hooks need to be used within functional components