Getting your portfolio on track

A Website Tutorial

IEEE

UAlbany Student Branch

Overview

1) An Introduction to Web Servers and Web Deployment Options
2) Working with and getting Domain Names
3) How React.js Works
4) Making a static page in React

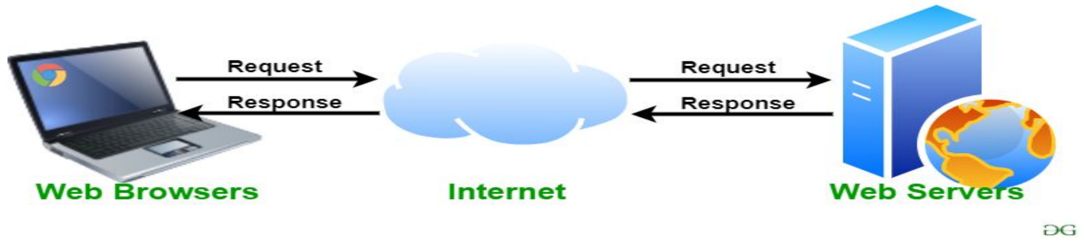
An Introduction to Web Servers and Web Deployment Options

What is a Web Server?

A **Web Server** is a computer that is setup to accept incoming HTTP requests and respond by sending back webpages, normally in the form of HTML Documents. The computers themselves are also normally outfitted with other technologies such as databases or server side applications.

In order for the computer to do this, it needs **Web Server Software**, This is a program that will run in the background managing incoming HTTP requests and sending back the appropriate responses.

In order for a web server to work smoothly, It should ideally always be on and have computational resources available, It is for this reason most web servers are hosted on dedicated hardware using server operating systems to ensure the machine is on and has maximum available resources 24/7.



Real Web Server Software

To run a webserver, your computer Web Server software, and this can widely depend on what backend (server side utilities) you’re planning on using for your website.

Most real web servers run at data centers will use high performance load balancing web server software such as Apache HTTP or NGINX written in languages like C or C++ for optimal performance.

APACHE
HTTP SERVER PROJECT

NGINX

Frameworks

Many developers will develop dynamic applications using language dependent **Web Frameworks** like Django (Python), Spring (Java), Angular + Node (JS), React + Node (JS), ASP.NET (C# & CLI languages). These frameworks allow for users to design active server pages that respond to dynamic content in databases etc, using software design patterns such as **MVC** (More on that later).

Deployment Options For Us Developers

Cloud Based Server	● Lets you upload framework applications without worrying about web server software ● Application resources scale with application popularity ● Costs Money, price scales with traffic ● Very High bandwidth scales with demand
Datacenter Server	● Gives greater control over web server software and server applications ● Costs Money, fixed monthly payments for server use ● Very High bandwidth
Home Server	● Full control over server hardware OS, web server software and server applications ● Costs hardware payment and electricity, Can use any old PC or even a 30$ Ras PI ● Limited by your home bandwidth, Not Practical for thousands of concurrent page viewers or hosting video streaming servers ● Need to portforward so your home network knows where to route web traffic
Deploying on your Local Machine	● Really bad idea unless you leave your PC on 24/7, ● Limited by home bandwidth ● You can deploy using built in, easy to use, web server software and test websites

The Good Stuff (Secret Options We Have available)

Ualbany provides anyone with a free datacenter web server with 100MB storage at http://www.albany.edu/~<Your Net ID> that you can upload to using an sftp client, with credentials Address: perspub.albany.edu, User: <Your Net ID>, Password: <Your MyUalbany password>, Port: 22

Unfortunately, It only works for static pages and has nothing in the way of backend support.

However you can still use it with frontend libraries like react as long as they don’t need dynamic support. Unfortunately this rules out things like MVC.

Demo: Uploading to UAlbany

So What Will We do?

For this workshop we will use the least favorable option to deploy our web app. We will deploy React MVC code on our local machine using the simple web server software that comes packaged in with Node.JS.

By default running any node program that takes advantage of web technology will automatically start a server on their local machine that can be viewed at the domain “localhost”

Working with Domain Names

How Domains Work

A **Domain** is just a references to an IP Address, the same way pointers are references to memory locations in C.

Each **TLD (Top Level Domain)** .com .net .us .uk, has Name Servers also known as **DNSs (Domain Name Servers)**. These servers are specially designed to take requests in the form of domain names and return the true IP Address of the site. They are like phonebooks of IP addresses.

All fully developed operating systems with networking capabilities will have a list of the IP Addresses of the **Root DNSs** stored locally on the computer somewhere with the OS's networking code.

Note DNS also stands for "**Domain Name System**", the protocol used to communicate with DNS servers

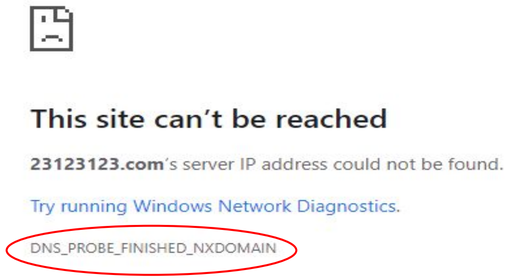
How Domains Work Contd.

When you request a webpage from a site, your computer will split up the domain and query DNS servers right to left.

First the TLD sent to the locally stored address of a Root DNS, which will then return the IP address of a DNS for that TLD.

The computer will then send the name of the site to the returned DNS appropriate to that TLD, which will return the real IP of the website if no other subdomains are being used.

This entire process of obtaining the true IP address of a site is the the **DNS Lookup / Probe** and you can see it when you search up a random string on google



This site can't be reached

23123123.com's server IP address could not be found.

Try running Windows Network Diagnostics.

DNS_PROBE_FINISHED_NXDOMAIN

How Domains Work Visual

Let’s Analyze the steps of accessing albany.edu/index.html

Client’s Computer

Time

User Requests page “albany.edu/index.html”, A Resolver Splits Domain into “.edu” and “albany”, and requests the IP of the “.edu” DNSs from a Root DNS

Root DNS

Looks up an IP of .edu nameservers in its TLD database and returns the IP of an Educase DNS

The Resolver sends “albany” to the provided Educase DNS requesting the true IP of the site

Educase .edu DNS

Looks up and Returns the true IP of albany.edu from its database

Receives the True IP of albany.edu and sends an HTTP request for index.html

Albany.edu Server

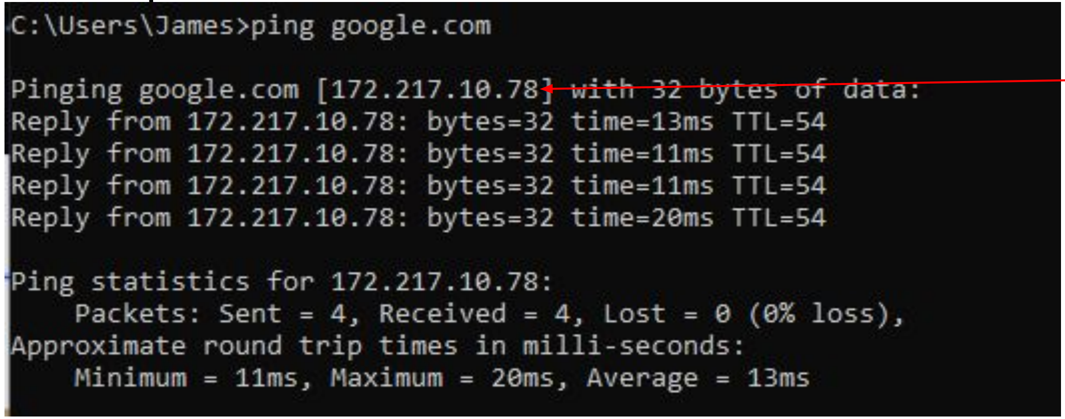
Receives request for index.html and returns it if it exists, otherwise returns a 404 page

Receives HTML code for index.html which can be loaded and displayed by the browser.

Try It Yourself

Use the ping command to get the true IP of a server From a DNS Lookup. (Works on both Windows and Mac)

When you run “ping” your computer will perform a DNS lookup for the true IP of the domain and then send the server data to check if it is up and responding to requests.



```
C:\Users\James>ping google.com

Pinging google.com [172.217.10.78] with 32 bytes of data:
Reply from 172.217.10.78: bytes=32 time=13ms TTL=54
Reply from 172.217.10.78: bytes=32 time=11ms TTL=54
Reply from 172.217.10.78: bytes=32 time=11ms TTL=54
Reply from 172.217.10.78: bytes=32 time=20ms TTL=54

Ping statistics for 172.217.10.78:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 11ms, Maximum = 20ms, Average = 13ms
```

True IP of google.com from a DNS lookup, (Yours may be different since google uses many complex load balancing algorithms and millions of servers to deal with its load of traffic)

The Capitalism of Domains

The Root Domain Name Servers which are the top level phonebook only allow domains decided by ICANN to be added, this is why I can't have a .ieee domain

These TLDs in turn all have their own DNS's which are managed by corporations who got contracts and governments. In order to get a domain into the "phone book" you must purchase it from the TLD provider who will bill you yearly or monthly to be in the book.

However, the most popular TLD providers like "Verisign" who own ".com"s and ".net"s will not sell to you directly, you need to buy domains through "resellers" who act as middle men and reap huge profits by charging additional monthly fees.

Despite this domains can still be relatively cheap, using Name.com (one of verisign's resellers) you can get a .com domain for just $12 a year

Not all is lost, Free Domains and helping the Poor

Thankfully, resellers like FREENOM will give away TLDs like .tk, .ml, .ga, .cf, and .gq addresses for free. The caveat is that if your site does not get 25 visitors every 90 days, they will remove the site and replace it with targeted advertisements to generate revenue for the poor island nations who own these TLDs. It should be noted that if you use the domain for a website you are almost guaranteed to get at 25 visitors in the form of web crawlers. you can also pay 9$ a year to buy these normally to support these countries.

Unfortunately these domains are looked upon negatively due to high use by scammers for emails and popups since they are free.

.tk FREE DOMAINS FOR ALL .ml .ga .cf .gq

Demo: Getting A Free Domain and Redirecting it

How React.js Works

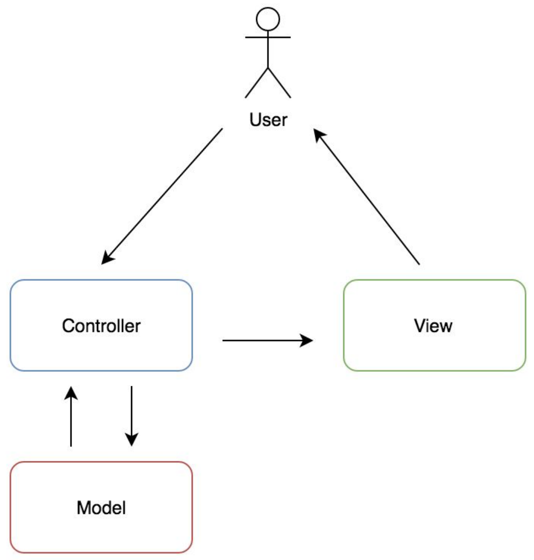
Model View Controller (MVC)

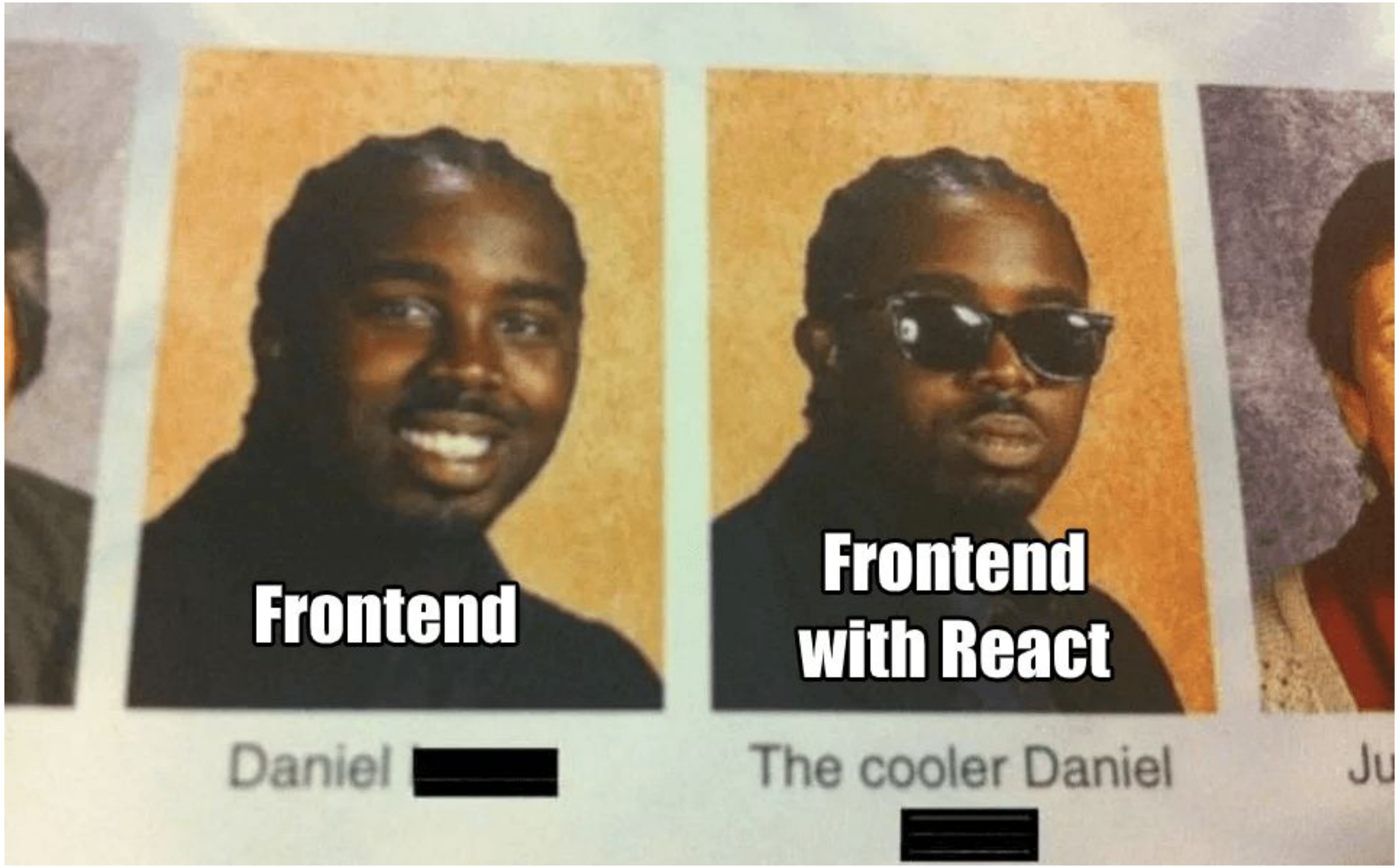
MVC is simply a design pattern

Model: a dynamic data structure

View: how the data structure is presented

Controller: how the user interacts with the data structure



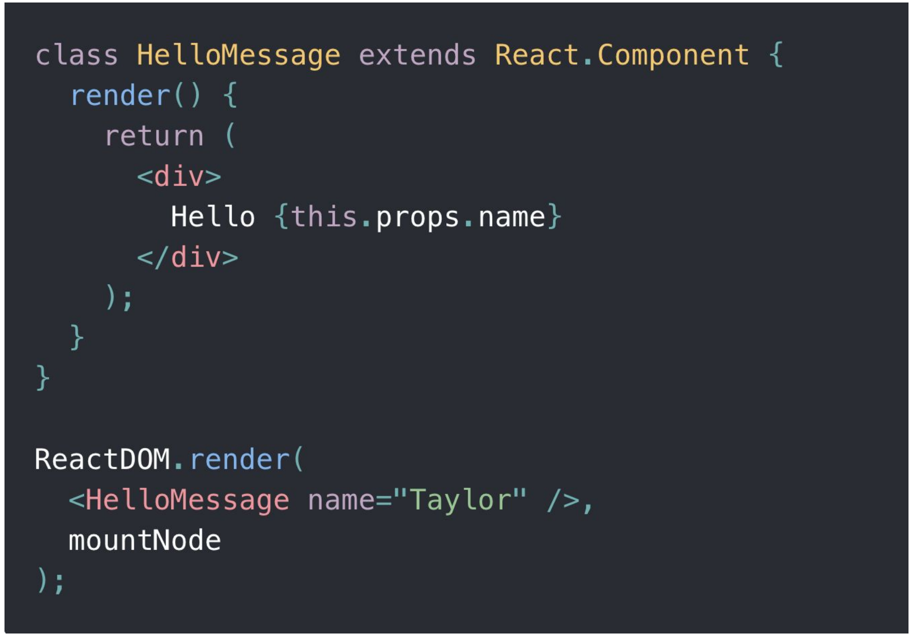

Frontend
Frontend with React
Daniel [redacted]
The cooler Daniel
[redacted]
Ju

React.js is not an MVC framework

- Created and maintained by Facebook, it’s a powerful javascript library for building user interfaces.
- React is component-based, meaning it encourages you to break up your code into modular, reusable pieces. Part of the reason React is so popular is because the codebase is so easy to expand upon and maintain.
- With more traditional frameworks, data that changed would have to be imperatively updated or linked to the DOM, to render the changes in the view.
- React uses a “virtual DOM” where each component is rendered on the ReactDOM, and whenever component data changes only that component is re-rendered.

JSX and Components

JSX is JavaScript that uses HTML syntax



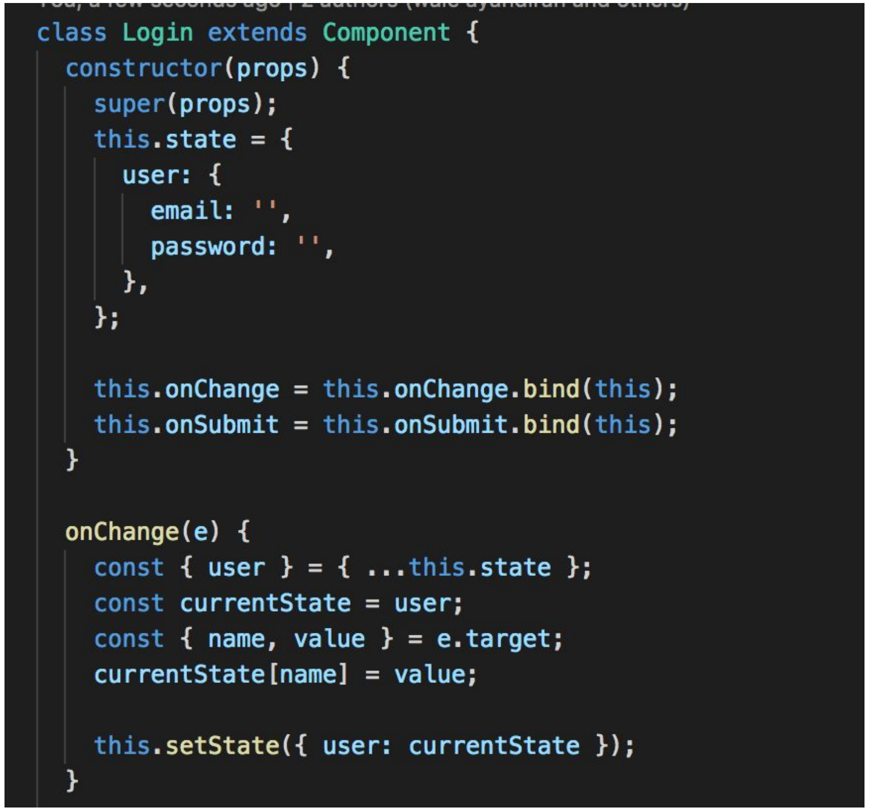
```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hello {this.props.name}
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="Taylor" />,
  mountNode
);
```

Components are similar to classes but they have a render method that returns HTML elements

They have a state object

Arguments passed into a component are called props

Managing data in the state of a component



```
class Login extends Component {
  constructor(props) {
    super(props);
    this.state = {
      user: {
        email: '',
        password: '',
      },
    };

    this.onChange = this.onChange.bind(this);
    this.onSubmit = this.onSubmit.bind(this);
  }

  onChange(e) {
    const { user } = { ...this.state };
    const currentState = user;
    const { name, value } = e.target;
    currentState[name] = value;

    this.setState({ user: currentState });
  }
```

React is able to create components using either Hooks or ES6 Classes. Hooks have a bit of a learning curve.

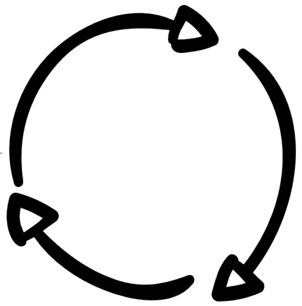
State should never be modified directly, use the setState() method

State updates are asychronous, so if one state depends on the value of another when updating, the value may not render with the correct value (arrow functions can help manage this)

State only flows one way, data can only pass down from a parent component to its children

Component lifecycle

constructor



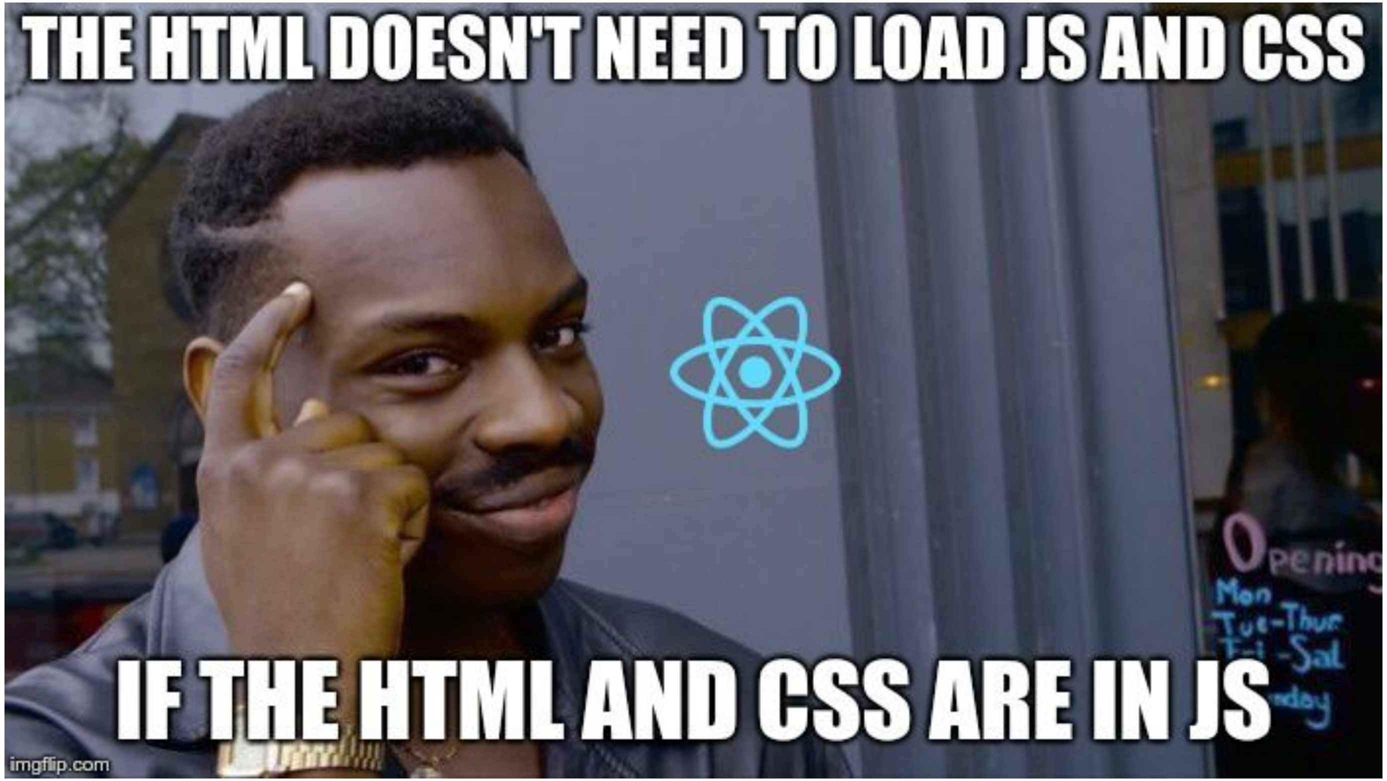
componentDidMount()

- Runs once, and only once, after the Component is rendered to the DOM

componentWillUnmount()

- Runs if the Component is removed from the DOM

These are just some of the lifecycle methods used in React to trigger a certain action dependent upon component updates and rendering. To learn more, see https://reactjs.org/docs/state-and-lifecycle.html


THE HTML DOESN'T NEED TO LOAD JS AND CSS
IF THE HTML AND CSS ARE IN JS
imgflip.com

Gatsby.js

Static site generator for React

Check out the Gatsby Docs

Making a static page in React

What is Gatsby?

Gatsby is a state-of-the-art modern site generator that pairs nicely with React frameworks. It is great for creating static sites, such as blogs, marketing, resumes, or portfolios ;)

We recommend Gatsby.js because of its great documentation and ease of use to get your website bundled and up and running on a server.

That's right, I use React

R	I can't
E	Setup
A	Node environment
C	Help
T	

Setting up the Development Environment

Windows

1. Download and install the latest Node.js version from the official Node.js website
2. Install Git

   https://www.atlassian.com/git/tutorials/install-git#windows

Mac

Install Homebrew

1. check if you have it: brew -v
2. Install using terminal
3. Type command

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

4. xcode-select --install
5. brew install node
6. Install Git

https://www.atlassian.com/git/tutorials/install-git#mac-os-x

Use the Gatsby CLI and Create a starter site

```
npm install -g gatsby-cli
```

Install the gatsby command line interface

```
gatsby --help
```

Show gatsby commands

Now use gatsby to create a “Hello World” project

```
gatsby new hello-world https://github.com/gatsbyjs/gatsby-starter-hello-world
cd hello-world
gatsby develop
```

Now it should be running on your local machine. Got to https://localhost:8000/

Try it yourself!

1. Fork our starter code and clone to your local machine
   https://github.com/carolynk/resume-site
2. `cd` into that directory (resume-site-master)
3. `npm install -g gatsby-cli`
4. `npm install react react-dom`
5. `gatsby develop`

Challenges

Change the name and icon to a character of your choice

(Use a square image for the avatar.)

Add a real github link to the footer

Add a project: Get an image from https://unsplash.com/

Change the css gradient background: https://cssgradient.io/

Add a skill using an svg from: https://konpa.github.io/devicon/

Surge

Once you have a hello world site on your local machine, you can deploy to surge.

Getting surge

```
npm install -g surge
```

Build your site (into a static site)

```
gatsby build
```

Now you have a static site in the public folder

To pick your own subdomain, go inside the public folder, and type:

Echo mysubdomain.

Some Developer Portfolio Examples for Inspiration

http://manparvesh.com

http://coryhughart.com

THANKS FOR COMING! WE'LL STICK AROUND IF YOU WANT TO ASK US QUESTIONS