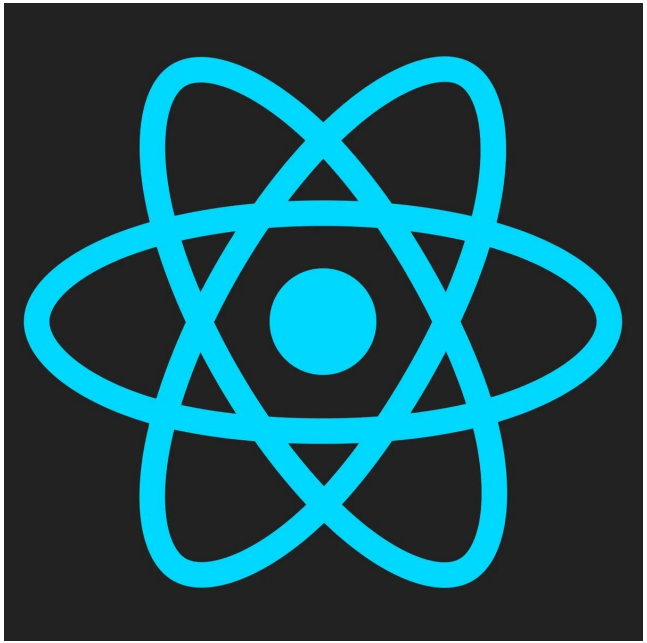


CSc 337
Dev stacks, MERN, and React

Benjamin Dicken

Project Check-ins

Project Check-in meetings on Thursday
Projects should be approximately 50% complete
We will ask questions about your project such as:

- How has the project gone so far?
- How are you dividing up the work? What has each member of the group been contributing?
- What has been the biggest challenge so far?
- What percentage of the work do you think you have completed?

Web Development Stacks

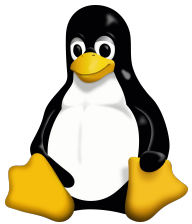
A set of tools often used together for creating web applications

Two well-known examples:

LAMP

MERN *(and variants such as MEAN, MEVN)*

LAMP

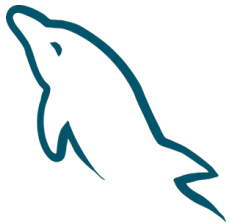


Linux

(or UNIX) Operating system family kernel commonly used for servers

Apache

Web server software



MySQL

A popular relational, SQL-based DBMS

PHP

A programming language, (previously) frequently used for web dev

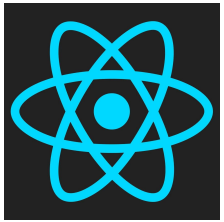
MERN

MongoDB

A NoSQL, document / object based DBMS

Express

Framework for handling requests to a server, via routes



React

UI and templating framework

Node

A JS interpreter / runtim, good for developing servers with

MEAN

MongoDB

A NoSQL, document / object based DBMS

Express JS

Express

Framework for handling requests to a server, via routes

Angular

Framework for creating web applications

Node

A JS interpreter / runtim, good for developing servers with

MEVN

MongoDB

A NoSQL, document / object based DBMS

Express

Framework for handling requests to a server, via routes

Vue

Web user interface framework

Node

A JS interpreter / runtim, good for developing servers with

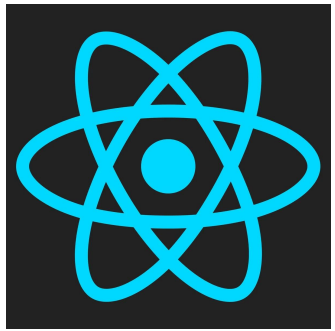
Why Web Stacks?

Why use a predetermined web development stack?

Why not just use whatever combination of tools you want?

MERN - A Useful Skill

https://www.indeed.com/jobs?q=mern+stack

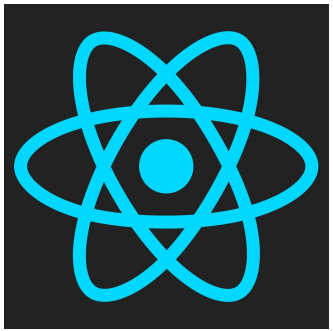


React- What is it?

- Widely used web development framework
- Maintained by Meta
- Can be used to generate UIs for mobile devices as well

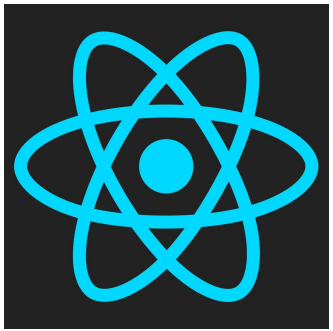
  (not just for web!)
- Provides mechanisms to keep UI code more organized and improve component reusability
- Usage flexibility

https://react.dev/



React - Why use vs plain HTML+CSS+Js?

- By Design, React allows us to organize our user interface into chunks (called ***Components***)
- Keeps things organizes, allows for more re-usability
- Less copy-paste!
- Also provides ***Hooks*** for data



Instead of This

```
<html>
<head> <!-- .... --> </head>
<body>
  <div class="segment" id="responsibilities">
    <h2>RESPONSIBILITIES LOGIN</h2>
  </div>
  <div>
    <div>LOG IN</div>
    <label for="usernameLogin">Username</label>
    <input id="usernameLogin" type="text"> </input>
    <br/>
    <label for="passwordLogin">Password</label>
    <input id="passwordLogin" type="text"> </input>
    <br/>
    <input type="button" onclick="login();" value="Log in"/>
  </div>
  <br/>
</body>
</html>
```

Create a React Component

```
import React from 'react';
function Login() {
  const [user, updateUser] = React.useState('?');
  const [pass, updatePass] = React.useState('?');
  // ....
  return (
    <div className="Login">
     <h1>Responsibilities Login</h1>
      <label>Username</label>
      <input onInput={u => updateUser(u.target.value)}></input>
      <label>Password</label>
      <input onInput={p => updatePass(p.target.value)}></input>
      <button onClick={e => sendLoginRequest()}>Login</button>
      <button onClick={e => createAccount()}>Create Account with Credentials</button>
    </div>
  );
}
export default Login;
```

```
import './App.css';
import Login from './Components/Login'
function Landing() {
  return (
    <div className="LoginPage">
      <Login />
    </div>
  );
}
```

Then use like this

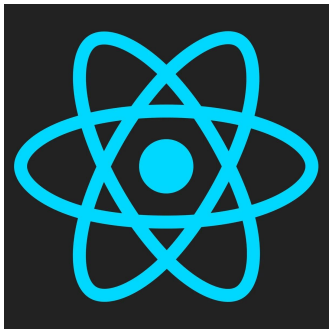
React - how to integrate

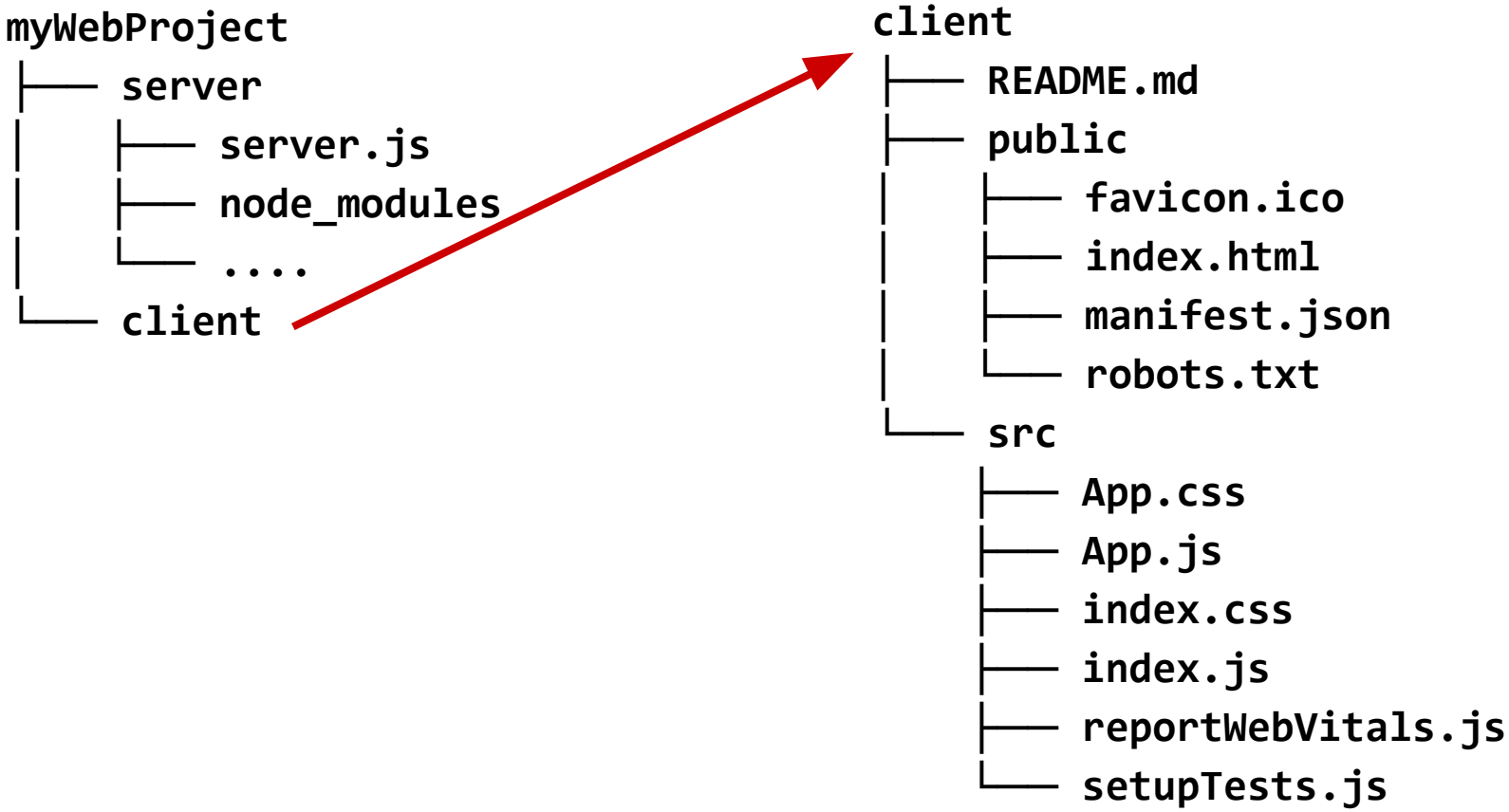
1. Develop Server-side as usual (Node, Express, MongoDB)
   EXCEPT: don't handle static files, no **`public_html`**
2. Initialize new react app directory for client design

```
$ npx create-react-app client
$ npm install react-router-dom   # will need later
```

3. Build your application
4. Run client and server on different ports

```
$ node server.js                 # server
$ npm start                      # client
```




myWebProject
server
server.js
node_modules
....
client
client
README.md
public
favicon.ico
index.html
manifest.json
robots.txt
src
App.css
App.js
index.css
index.js
reportWebVitals.js
setupTests.js

Building “Responsibilities” afresh

1. Review old Responsibilities
2. Create a new, simplified version using React
3. Will require components, hooks, and navigation

Demo Time!

(Can follow along with starter code on schedule)