**JSX:**

Q: What is it for?

A: It’s used JS syntax to help us write our code more like HTML

Useful in React because it’s natural to render elements on HTML

Example:

Const people =[s

{name: ‘Michael’},

{name: ‘Ryan}.

{name: ‘Tyler}

]

const element = <ol>

{people.map(person => (

<li key = {person.name}>{person.name}</li>

))

</ol>

* Wrap javascript code in curly braces

Q: Translate this sample HTML code into JSX

<div class='greeting'><h2>Hello world!</h2></div>

A: React.createElement(‘div’,

{className: ‘greeting’}.

React.createElement(‘h2’, {}, ‘Hello World’)

);

* Always have to wrap it in quote ‘div’

**JSX RETURNS ONE SINGLE ELEMENT:**

\_That means that the JSX elements must be wrapped in <div> or <span>

**COMPONENTS:**

Q: What is it?

A: *Reusuable* pieces of code ultimately responsible for returning HTML to be rendered onto the page

\_In React, it’s under the name Class

Ex:

class ContactList extends React.Component {

render(){

const people = [

{name: ‘Michael’},

{name: ‘Ryan}.

{name: ‘Tyler}

]

return <ol>

{people.map(person => (

<li key = {person.name}>{person.name}</li>

))

</ol>

}

}

ReactDOM.render(

**<ContactList/>**

document.getElementById(‘root)

}

//Renders ContactList—render whatever class/ element you want to have on the page

\_One component = do one thing

Q: What does Webpack do?

A: Convert JSX code to regular JS code before reaching the browser

**ADMIN NPM INSTALL**

Sudo npm install

Q: What’s Create-React-App?

A: Facebook React’s setup without any configuration, but it comes with React-dom and React-scripts

Q: React-scripts?

A: Installs Babel, webpack, webpack-dev-server

Q: Yarn?

A: Package manager similar and is like an improved version of npm

**COMPOSITION:**

React encourages us to build application using composition, not inheritance

Ex: Instead of writing

Class ContactList extends React.Component {

render(){

const people = [

{name: Michael’},

{name: ‘Bryan’},

{name: ‘Jennifer’}

]

return <ol>

{people.map(person =>

<li key={person.name}>{person.name}</li>

))}

</ol>

}

}

I can write, using 2 components:

Class ContactList extends React.Component {

render() {

const people = **this.props.contacts**

return <ol>

{people.map(person => (

<li key = {person.name}> {person.name} </li>

))}

</ol>

}

}

class App extends Component {

render(){

return(

<div className= “App”>

<ContactList **contacts ={[**

**{name: ‘Bryan’},**

**{name: ‘Jennifer’},**

**{name: ‘Jonathan’}**

**]}**

</div>

);

}

(Red highlight is the this.props.contacts property)

(const people just refers to it so that it can be map out after)

Q: So what did the above mean?

A: Ever composition should only do one thing

\_Favor composition over inheritance

**QUESTIONS**

Q: What’s composition vs inheritance?

A:

**Inheritance:** Does typeB want to expose the complete interface of typeA such that typeB can be used where typeA is expected?

Ex: a Cessna biplane will expose the complete interface of an airplane. So that makes it fit to derive from an airplane

**Composition:** Does TyepB only want some/ part of the behavior of typeA?

Ex: A bird may only need the flying behavior of an Airplane. In this case, this is composition cuz you only need to extract that flying behavior

* People usually prefer composition> inheritance because it’s more malleable/ easy to modify. You don’t have to bring all the characteristics of something if you just want to take a certain part of it out

Q: Composition vs inheritance **in React**? Composition in React (how is it a composition)

A: For example,

<Page />

<Article />

<Sidebar />

Article and Sidebar is part of the Page. It’s an item on the page. However, I don’t want all the Article and Sidebar to display all the characteristics of the page. I only want part of it. So below:

<Page>

<Article />

<Sidebar />

</Page>

Here, I can manipulate Article and Sidebar to my liking

Q: Reusable codes for components?

A: +) you can write components that can be used multiple times for different things

theyre reusable in that the arguments they take, props, can differ the behavior or display of the component like the example i provided

<Button color = “red” />

<Button color =”green” />

\_Also in React, Component == Function (approximately the same)

**QUESTIONS ABOUT THE PROJECT:**

Q: How do I know which js file will be displayed? (Where to go to find it?)

A: For the Create\_React\_App template, it’s in “index.js”, where the ReactDOM renders <App /> at the ‘root’ id

ReactDOM is to prepare the DOM and then compare with the real DOM to see which item (node) should be replaced

Q: How do I import a Component from a separate page to be displayed on the main page?

A: First, import the component I want into the main component. The main component job’s mainly to display and manipulate smaller, sub components

Ex: import ListContacts from './ListContacts'

Second, put that new Component into the div

class App extends Component {

render() {

return (

<div>

**<ListContacts contacts={contacts}/>**

</div>

)

}

}

(The {contacts} refers to the value of an array defined above in the file)

Third, in the component I want to manipulate (not the main component), I can refer to the value of the props as: this.props.name\_of\_value

Ex: this.props.contacts.map((contact =>