**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)

\_Favor composition over inheritance