**NOTE 3: STATE MANAGEMENT**

Q: how to rename folder in terminal?

A: mv old\_name new\_name

Node server.js : to start the backend server

**PROPS:**

Look at this custom attribute (prop)

<User username=’Bryan’ />

* This is the same as giving a function an argument to call it: var User = function username(Bryan)
* Props = argument

Now we can access that prop by using this.props

Ex:

class User extends React.Component {

render(){

return(

<p> Username: {this.props.username} </p>

<p> Is Friend?: {this.props.friend}</p>

)

}

}

<User **username=’Bryan’ friend={ true}**/>

NOTE: {} denotes Javascript expression

Highlighted in red is the props of that **class**

**CREATE-REACT-APP:**

1)

\_Sudo npm –g install Create-React-App

\_Then create-react-app project\_name

This will give you an src file with App.cs and App.js

2) But if you clone it from the Facebook repo here <https://github.com/facebookincubator/create-react-app>

Those won’t have that, but it’s unnecessary to clone since it’s just Facebook backend files.

create-react-app my-app

**TERMINAL:**

\_mkdir creates a folder/ directory

\_To remove empty folder, use rm –r name

\_To create an empty file, use touch. Ex: touch filename.js

\_Install Shell Command on Atom: Use cmd + shift + P

\_To open a file, just use atom filename

\_You can create an empty file with atom using atom newfilename. After putting content into that file, you can save it, then it’ll show up

**REACT:**

\_We can store state in a component

Q: Why do I need to do **import React, { Component} from 'react'**?

A: First, we import the React module from the external react module

We use {Component} so that we will have access to use Component as opposed to writing React.Component

Ex: Without the {Component} imported:

class ListContacts extends React.Component{}

But with it:

class ListContacts extends Component{}

Q: Why do we need to extend React.Component class?

A: \_Extending a Class is to make an instance of that class

But we do it so that we can initialize the constructor and change by using this.state

\_Within the render(){

return(

<UI>

)

}

}

Within the return{}, give a UI

\_ import ListContacts from './ListContacts'

* This means I import the ListContacts from that URL to use in the UI view
* In the ListContacts file, I have to export default ListContacts

\_ <ListContacts contacts={contacts}/>

* Pass data from 1 component/ file down to a component, I have to give it an attribute, and then you can access that via this.props
* The {contacts} refers to the variable contacts above (which contains all the name, avatar url etc.)
* With function it’s like setting a default value?

Q: Confirm this

A:

\_ So the console.log(‘Props’, this.props) from the ListContacts page means that you access your props by using this.props

(Like we can access arguments passed into a regular JS function, we can access components’ props with this.props)

Q: How to pass a prop?

A: <Component prop\_name={prop\_value} />

Ex:

<Clock currentTime={newDate().getTime()} />

Q: How to access the value of a prop from inside the component?

Ex: <Clock currentTime='3:41pm' />

A: **this.props.**currentTime

Q: how to map?

A: <ol>

this.props.contacts.map(contact) => (

<li **key={contact.id}>**

{contact.name}

</li>

))}

</ol>

}

}

}

1. Need to give a key
2. Use {} to indicate JS
3. Give className so that I can highlight the CSS class

Q: How to give an avatar image (background image)?

A: First, give a className to initialize the CSS class, then put in an inline style

For Background Image of that (for example, if it’s in a list, then the images will appear horizontally), use the **backgroundImage: `url(${backgroundURL})`**

Remember to use the ` symbol, not ‘. Otherwise, will have error. The ` symbol is below ~

Ex:

<li key ={contact.id} className='contact-list-item'>

<**div className= 'contact-avatar' style={{**

**backgroundImage: `url(${contact.avatarURL})`**

}}/>

This <li> is wrapped inside the <ol>. So unordered list, then “list” inside

Q: how to create a remove button> (appears as an X)?

A: <button> Remove </button>

Ex:

<button className= 'contact-remove'>

Remove

</button>

Q: how to pass multiple props?

A: Pass them in individually

Ex: <School name=’UPenn’ tuition=’60k’}>

^ ONLY DO THE FORMAT ABOVE IF I HAVE TO HARD CODE (like defining the value before hand, like functions in ES6)

OTHERWISE: <Component prop\_name={prop\_value} />

**SUMMARY ON PROP**

Q: How to pass and access a prop?

A: First, pass a prop:

<LogoutButton text=’Wanna log out?’ />

Text is the prop. The string is the value.

To access the text prop from inside the component, we use this.props.text

Ex:

render() {

return

<div>{this.props.text}</div>

}

* Here we don’t use () at return because we only need to return 1 thing

Q: Components toProps?

A: Like function to argument?

Q: What’s git commit –am?

A: Both add and commit together

**QUESTIONS:**

Q: Command to automatically refresh the page (npm)?

A: Nodemon

Q: What are states in React?

A:

Q: Is “Class” the same thing as “Component” in React?

A: Abstract-based class, meaning it contain one or more abstract functions (render())

Implemented by the subclass (child class)

<https://facebook.github.io/react/blog/2015/12/18/react-components-elements-and-instances.html>

Q: Another explanation for “Class” vs “Component”?

A:

A Tesla Model 3 is a type of car, so you can make TeslaModel3 a proper subclass of Car. Everything that a generic car can do, a Tesla Model 3 can do. So this is using inheritance properly.

On the other hand, a Toyota HiAce may share similarities to a car, but it might only have 2 seats instead of five, its doors open differently etc. So it should not extend from the Car class. Instead, we can either:

Abstract the Car class even further into a Vehicle class, and extend a Van class from the Vehicle class, or

Break the Car class into smaller components (like Wheels, Doors etc, and combine them in a different fashion to form the van.

If you find the subclasses are overriding the parent class' methods, then maybe think about choosing a different parent class (first example), or using composition instead of inheritance (second example).

Q: React.Component vs React.createClass?

A: More or less the same thing

Q: What does the render() method return?

A: An element of the DOM. Before REACT DOM’s applied, it’s not yet an element of the DOM. But after it is.

This is due to the Diffing Algorithm

Q: Argument to a Function is like \_\_\_\_\_\_ to a Component?

A: Props (and value)

\_Before importing a sub component, I need to export it first

ex: export default ListContacts

When you pass multiple props individually to a component, you don’t separate them with commas. Instead, just pass them in

Ex:

<Clock time={Date.now()} zone =’PST’ />

Q: How does the child component know how to access props via this.props?

A: Because whenever you write “this”, it means reference

**FUNCTIONAL COMPONENTS:**

\_Meaning we can use Function in the same way as Component. But we don’t have this.props anymore

Ex:

1. class User extends React.Component {

render(){

return(

<p> Username: {this.props.username} </p>

)

}

}

* Conversion to function:

1. function User(props)[

return(

<p> Username: {props.username} </p>

)

}

* So in the 1st example, props isn’t passed as an argument, but in the second instance, it is

**STATELESS FUNCTIONAL COMPONENT**

If all what your class has is a render() method, then instead of creating that class, you can just create a function that takes in an argument props. And we don’t have to return this.props (we can just use “props”)

\_It returns a JSX UI:

ex:

function ListContacts(props){

return(

<ol className= 'contact-list' >

{props.contacts.map((contact) =>(

<li key ={contact.id} className='contact-list-item'>

<div className= 'contact-avatar' style={{

backgroundImage: `url(${contact.avatarURL})`

}}/>

<div className='contact-details'>

<p>{contact.name}</p>

<p>{contact.email}</p>

</div>

<button className= 'contact-remove'>

Remove

</button>

</li>

))}

</ol>

)

}

* Return a chunk
* No this.props => props. Don’t have to worr about what context “this” is in
* Returns a UI description

Q: How would you access this

<IngredientList items={ingredient.items} />

A:

1. If it’s a Component: this.props.items (not “this.ingredient.items”)
2. If it’s a stateless functional component: props.items

More examples:

These are equivalent:

1)

class Email extends React.Component {

render() {

return (

<div>

{this.props.text}

</div>

);

}

};

2)

const Email = (props) => (

<div>

{props.text}

</div>

);

//Use arrow syntax when there’s no “this” so no confusion about its context