React Notes

Npm init -y, initialize a node server package.json. package.json gives the anchor to install ndoe modules. We create an express app after initializing express. And app.use allows us to use everything in the folder to serve it up statically. \_dirname+’/

Can use ReactDOM.render to render item to root. Render takes the html and the js to get element. It looks for index.html.

<script crossorigin src="https://unpkg.com/react@16/umd/react.production.min.js"></script>

This is React

<script crossorigin src="https://unpkg.com/react-dom@16/umd/react-dom.production.min.js"></script>

This is the react-dom. React dom renders the information. React dom Mingling of html and JS is JSX. Babel is a js compiler. React create component takes type, props and what is inside of the element.

Type is the element to create. An id needs to be a property of the element so that is the second element. Any class or ids are props. Cannot do images in jsx with img tag as every tag must be closed. Each jsx creates a react element, not a dom elemnt.

{saleOn() ? 9.99 : 59.99}

Ternary operator so if saleon then 9.99 else 59.99. React elements are essentially just regular javascript objects. React Dom. Render takes to args, what to render and where to render. The where is the actual dom element. The what is a react element.

React Virtual Dom, when react renders it is sent a bunch of react elements. It creates a structure. First time renders takes eeverything and puts it into the actual dom. If react needs to update the virtual dom, it only updates the particular tree. It compares the copy to the original and updates what needs to be updated.

Props are immutable, they are managed by parent not by the component itself. All props are meant to be pure variables.

Any time we have a react component we can add attributes.

Map is the same as forEach, but we make an array each time we loop.

React works a lot better when we give the elements we map from an array a key which can essentially be an index or some other variable. We map though the array, build a new array of components and we can update props that way.

New Date().getFullyear(), javascript Date Object. Must use `` if using the ${syntax}

Object literals

Const name = ‘’

Const age =

Const job = ‘’

Const prop = ‘aPropertyName’

Const x = {

Name,

Age,

Job,

[prop]:’Hello’

}

This bracket notation is now given the name of the prop. This is reusing a variable property name.

Const abilities = [Speed, Smart, Crazy]

Const datafromapi = {

Name: Thanos,

Age: 1000000,

Job: Villian,

Abilities:[Strength, …abilities, telepathy, immortal, destruction]

}

Destructuring

Const {name, age, job} = dataFromApi

With spread syntax, it unpacks in place without having to do the looping.

If we use

Const x = {

Name: Francois

}

Const a = {…x}

A is a copy of abilities, rather than a pointer to the abilities which changes the variable as well.

So by changing a.name it does not change x.name. … loops through array or key value pairs.

setInterval can help run a code x amount of times, given the argument and first function is a function or callback function.

Prototypes are added to every class.

Wigth classes we use class Shape{

Constructor(h, w, tyope){

This.h,

This.w,

etc

}

}

React.Component has a lot of stuff that our react classes can inherit. Classes in react must also have the render method in order to return the JSX. Constructor runs when the object is initialized. We must call this with the super class. In a class base component we need to use this. Props. We usually use the map function to build an array of components.

Webpack creates the development server for us. Eslint are built in to alert us of errors in our code.

We do not need StrictMode. Gives us helpful errors however.

State:

The value of variables at any given time. We can initialize state using this.state in the constructor.

setTimeout(()=>{

            this.setState({

                text:"State has been updated."

            })

        }, 2000)

Set timeout allows us to update the dom within a given interval. Can only use rocket functions in settimeout and react. This is because the rocket function does not cerate a new this.

In the onClicks, we don’t run code we pass the code that we want to run. Every event comes with the event object that allows us to prevent default.

Everytime state changes render is called and is able to update the Dom with ne state. State gives us a single source of truth.

Remember spread syntax can be used to create a copy.

Component LifeCycles.

React is its own thing. Server will be node/ express, We make an http request, node packages js, html and css. Each http request returns json. We can use axios to make http requests.

Nasdaq, openweathermap: Whalburger891991%%

Mcafee: shbad83u2@^#%JJAB@

Yodlee: 611&&T@haghs0xksa2\*2nd

Http request are asynchronous, javascriipt will not wait for response it keeps on moving. State is meant for the very thing where data need sto change. SetState should never be in render as it gets caught in an infinite loop, as setstate is called each time it renders which causes re render. Component DidMount

This is the payload. Componentdidmount is called autpmatically after the first render. The constructor runs sfirst, render runs second. As soon as first render is done, once componentdidmount is there, it will run it. This can allow the url apis to be called. Update state also causes a rerender, Every React component has to mount. ComponentdidMount is good to initiate wherw you will have external requests. Initializations that require dom nodes should also go here. Any js that needs to run after the component mounted, after first render that’s when we need component did mount.

Includes is an es6 method and checks if a string is in another string. It returns a Bool.

Remmeber we can do react fontaweosme.

React Router is a collection of navigational components. Can help create bookmarable URLS or deep links. NavLink just gives us a lot more styling power. We can call render in Route and pass a callback. We can render a component and use this in the Route ehwne we need to pass props into the component.

Fifa pw: Whalburger89()

Better to put the Navbar in the Route component. History is the stack. The length attribute represents how many places are on the stack. Pushing a path means we can move the user somewhere else. If we render a component in App we take control and say how and what we want to render. If we want the history object to be available we can enter it. We use props.history or props .matchs. This way we can get the location and match properties. Route using component only decides how to render the component. We can make a landing page for example using .push with a settimeout function callback. This is how the stack works, we end up whatever is at the top of the stack. Block gives a quick obstacle and throws a prompt before use leaves the page.

We can put wildcard routes for params

When using the match we use match.params and whatever param is sent into the param after the slash.

Redux

Redux is a state management tool. We previously did component based state management. Redux creates a store that is available outside the application. The component updates the store, the other component subscribes to the store to know what happens. We keep application state, not component state.

Redux can be used with any framework. There are actions, action creators, reducers, dispatcher. React uses react-redux. Redux and react each talks to react-redux. This has provider and connect. We essentially want to make our components talk to the store. The provider knows about the store we wrap the entire react app in the provider.

Objects in the store make up the application store. The store is informed by a bunch of little functions. These functions are called reducers which return a piece of state. They return an object and it collectively makes up the store. We never update the store directly but we use the reducers. All pieces of state make up the store. To update the reducers we have action creators. These are also little function that create actions. Actions are objects with at least a property of type. Action creators create the actions. The dispatch is built into the redux. This dispatches an action. Action goes to dispatch, dispath informs all the reducers about the action. The reducer return their piece of state, which create an object in the store, reducers get notified from actions and can update.

To set up a redux/react app we need react-redux. We need the Provider component around everything. A provider has to have a store. This allows redux to exist. We need reducers to populate the store. We always start with the root reducer. To make the root reducer we need a method from redux called combine reducer which combine reducers into one big store. We use CombineReducer and then get each indivuidak reducer. All reducers have 2 params. The current state, usually provide a default state and info that came from an action. We call combine reducers with an object, each key is a piece of state in the redux store. Each value will be a reducer. We want a component to know about redux. We need react-redux. We need the connect function. This is at the component level. We also change how we export. We export connect class. Map state to props, maps dispatch to props and merge props. Map state to props maps a piece of redux state to the component props. MapStateToProps this takes state. Which is the store or rootreducer. This function returns an object with property is the local prop name to the component. Value is the property in the root reducer. That means a piece of the store. The dispatcher helps us update redux store.

Action creators are just functions. They return actions. An action is an object with at least a property of type. Action creators are handed to the dispatch. Remmeber the reducer informs the component. The action calls the function to let the store know that the state has changed. We can also add another property called the payload. Payload contains data that comes along with the type. When we call the action it goes to the dispatch hich goes to the reducer which updates the store. We want our actions to run through the dispatch so that the reducers can be notified and update if needed. The mapdispatchtoprops maps to the dispatcher that notifies each reducer. It ties the component to the dispatch. Takes an argument of the dispatch which is part of redux. It returns bind action creators. We hand bindactioncreators an object, the property will eb a local prop, each value is a function that is dispatched when run. Second arg for bind actioncreators is the dispatch. The reducer gets notified everytime the dispatch is called and sends an action over. The redux store knows once something has change I need to update the store.

Reduce loops through an array and uses and accumulator with the item to accumulate and the second argument is the initial value.

const frozenQuantity = this.props.frozenData.reduce((accum,frozenItem)=>

            accum+frozenItem.quantity,0)

Redux Middleware.

Remember redux is just the store and the store informed by functions that returna piece of state call reducers. The connect function gives us access to the store. This new mapping is through mapstate to props. Action creators are not set up until we bind action creators. If we want to do something before a dispatched action gets to the reducer we can do something via redux middleware. Redux runs synchronously so it immediately sends action to dispatch and then to reducer. We can use redux promise to deal with this. Redux promise dispatches the resolved value of the promise. Middle ware is applied to the store in the proivider.

import { createStore, applyMiddleware } from 'redux';

import rootReducer from './reducers/rootReducer';

import reduxPromise from 'redux-promise';

const theStore = applyMiddleware(reduxPromise) createStore(rootReducer);

Redux thunk gives us more control than redux promise. Allows us to write action creators that return a function instead of an action. It returns a function that runs. When we pass a function in redux thunk , first function returned is the dispatch and the second is get State. We may want to dispatch actions or some other manipulation if some data is already present.

Remmeber reducers always get a piece of state and an action.