npm install –g create-react-app

create-react-app my-react-app

cd my-react-app

atom .

npm start

npm test

package.json

* This is where the manager happens
* Name of app
* Version
* Dependencies
  + React package
  + React-dom package
  + React-script package
* Scripts (short cut)
  + Start
  + Build
  + Test
  + Eject

Index.js

* ReactDOM.render(<App />, document.getElementById('root'));
* Meaning whenever I find this root, I want render <App /> component
* <App /> this App.js

In order to build a component we need:

* import React, { Component } from 'react';
* you need a react, and you need a component
* the component will import something and export something
* the class of the component get exported
* render function return jsx (not template)

JSX

* <div name="techsith">Hello</div>
* <div name>Hello</div>
* <div name={false}>Hello</div>
* <div name={1+1}>Hello</div>
* <div ClassName={1+1}>Hello</div>
* <div htmlFor={1+1}>Hello</div>
* <div id={1+1}>Hello</div>

Create a component using function

function SayHello(){

return <h1>hello</h1>

}

class App extends Component {

render() {

return (

<SayHello></SayHello>

)

}

}

React Class Component vs Functional Component

Class based component we are going to extend it using the react components

* We need to import component library
  + import React, { Component } from 'react';
* class based component
  + class Users extends Component {}
    - render() method
    - only one parent tag in return ()
* export
  + export default Users;
* import
  + import Users from './users/Users'
* pass props as attribute in tag
  + <Users title="Users List"></Users>
  + Use the props as {this.props.title}

Function based component

* we need to import react
  + import React from ‘react’;
* we need to write
  + const User = (props) = > {}
* export
  + export default User;
* import
  + import User from './User';
* we can pass props to children
  + Ex. <User>JAMES</User>
  + Receive as props.children
  + Use props as {props.children}
  + we can pass attribute props as <User age=”30”>JAMES</User>
  + as use props {props.age}
* Difference: no render method, no component extends, no import component, props not available yet, need to pass argument.
* Use ES6 as much as possible
* When we want to maintain the STATE you cannot use the functional components
* Container component usually class component

State (DOES IN SIDE) and Props (DOES OUTSIDE) (make react dynamic)

* Set state => Virtual Dom
* State
  + Can only use state inside components created using class
  + Functional component are stateless component
  + // state is a reserved word
  + state = {
  + //property
  + users: [
  + {name: "John", age:20},
  + {name: "Jill", age:30},
  + {name: "Peter", age:40},
  + ],
  + title: "User List",
  + key: 1
  + };
  + We can iterate the state using map function
    - const newState = this.state.users.map((user)=>{})
    - when we return the map, we better give component an unique key
* <button onClick={this.functionName}> </button>
* Declare functions ES6
  + makeMeYounger = () => {
  + const newState = this.state.users.map((user)=>{
  + const tempUser = user;
  + if(tempUser.age>0){
  + tempUser.age -= 10;
  + }
  + return tempUser;
  + });
  + this.setState({newState});
  + }
* Change state
  + this.setState({});
  + the static change in virtual dom, react will compare the difference between the old state and new state, update information in the dom.

Two way binding

* Normal click event
  + <button onClick={()=>this.functionName(‘val’)}>Anonymous Function </button>
* Click that take with parameter
  + <button onClick={this.functionName.bind(this,’val’)> Bind Function </button>
* Take Input change
  + <input **onChange**={this.functionName} **value**={this.state.xxx} placeholder={this.state.xxx}>
  + functionName = (event) =>{
  + this.setState({
  + name: event.target.value
  + })
  + }

How component communicate with each other

* Grandparent
  + <div className="App">
  + <Parent changeTheWorldEvent={this.changeTheWorld.bind(this,'new World')}
  + keepTheWorldEvent={this.changeTheWorld.bind(this,'same world')}
  + title={this.state.title}> </Parent>
  + </div>
* Parent
  + <div>
  + <Child doWhatever={props.changeTheWorldEvent} title={props.title}/>
  + <Child doWhatever={props.keepTheWorldEvent} title={props.title}/>
  + </div>
* Child
  + <div>
  + <button onClick={props.doWhatever}>{props.title}</button>
  + </div>

UniqueId

* npm install react-html-id –save
* import UniqueId from ‘react-html-id’
* constructor(){
* super();
* UniqueId.enableUniqueIds(this);
* this.state = {
* users: [
* {key: this.nextUniqueId(), name: 'john', age:20},
* {key: this.nextUniqueId(), name: 'peter', age:30},
* {key: this.nextUniqueId(), name: 'jill', age:25}
* ]
* };
* console.log(this.state);
* }

List add delete

* removeItemEvent = (index,e) =>{
* const users = Object.assign([],this.state.users);
* users.splice(index,1);
* this.setState({users:users})
* }
* changeMe = (id,e) =>{
* const index = this.state.users.findIndex((user)=>{
* return user.key === id;
* console.log("sss")
* })
* const user = Object.assign({},this.state.users[index]);
* user.name = e.target.value;
* const users = Object.assign([], this.state.users);
* users[index] = user;
* this.setState({users:users});
* }
* render() {
* return (
* <div className="App">
* <ul>
* {
* this.state.users.map((user, index)=>{
* return (<User2
* changeEvent={this.changeMe.bind(this, user.key)}
* deleteItem={this.removeItemEvent.bind(this, index)}
* key={user.key}
* age={user.age}>{user.name}</User2>)
* })
* }
* </ul>
* </div>
* )
* }

Fragments

* const Temp = () => {
* return (
* <div>
* <div>Hi</div>
* <div>Hello</div>
* </div>
* )
* }

Bad syntax

* const Temp = () => {
* return (
* [
* <div key="1">Hi</div>,
* <div key="2">Hello</div>
* ]
* )
* }

Use Fragment

* import React, { Component, Fragment} from 'react';
* const Temp = () => {
* return (
* <Fragment>
* <div>Hi</div>
* <div>Hello</div>
* </Fragment>
* )
* }
* const Temp = (props) => {
* return (
* <Fragment>
* <div>
* {
* props.greeting ==="hi"
* ?<Fragment> &lt;div&gt;{props.greeting}&lt;/div&gt; </Fragment>
* : props.greeting
* }
* </div>
* </Fragment>
* )
* }

Life cycle